Industrial Robotics

Theory, Modelling and Control

Industrial Robotics

Theory, Modelling and Control

Edited by

Sam Cubero

Contents

Preface

Robotics is the applied science of motion control for multi-axis manipulators and is a large subset of the field of "mechatronics" (Mechanical, Electronic and Software engineering for product or systems development, particularly for motion control applications). Mechatronics is a more general term that includes robotic arms, positioning systems, sensors and machines that are controlled by electronics and/or software, such as automated machinery, mobile robots and even your computer controlled washing machine and DVD movie player. Most of the information taught in mechatronic engineering courses around the world stems from industrial robotics research, since most of the earliest actuator and sensor technologies were first developed and designed for indoor factory applications.

Robotics, sensors, actuators and controller technologies continue to improve and evolve at an amazing rate. Automation systems and robots today are performing motion control and real-time decision making tasks that were considered impossible just 40 years ago. It can truly be said that we are now living in a time where almost any form of physical work that a human being can do can be replicated or performed faster, more accurately, cheaper and more consistently using computer controlled robots and mechanisms. Many highly skilled jobs are now completely automated. Manufacturing jobs such as metal milling, lathe turning, pattern making and welding are now being performed more easily, cheaper and faster using CNC machines and industrial robots controlled by easy-to-use 3D CAD/CAM software. Designs for mechanical components can be quickly created on a computer screen and converted to real-world solid material prototypes in under one hour, thus saving a great deal of time and costly material that would normally be wasted due to human error. Industrial robots and machines are being used to assemble, manufacture or paint most of the products we take for granted and use on a daily basis, such as computer motherboards and peripheral hardware, automobiles, household appliances and all kinds of useful whitegoods found in a modern home. In the 20th century, engineers have mastered almost all forms of motion control and have proven that robots and machines can perform almost any job that is considered too heavy, too tiring, too boring or too dangerous and harmful for human beings.

Human decision making tasks are now being automated using advanced sensor technologies such as machine vision, 3D scanning and a large variety of non-contact proximity sensors. The areas of technology relating to sensors and control are still at a fairly primitive stage of development and a great deal of work is required to get sensors to perform as well as human sensors (vision, hearing, touch/tactile, pressure and temperature) and make quick visual and auditory recognitions and decisions like the human brain. Almost all machine controllers are very limited in their capabilities and still need to be programmed or taught what to do using an esoteric programming language or a limited set of commands that are only understood by highly trained and experienced technicians or engineers with years of experience. Most machines and robots today are still relatively "dumb" copiers of human intelligence, unable to learn and think for themselves due to the procedural nature of most software control code.

In essence, almost all robots today require a great deal of human guidance in the form of software code that is played back over and over again. The majority of machine vision and object recognition applications today apply some form of mechanistic or deterministic property-matching, edge detection or colour scanning approach for identifying and distinguishing different objects in a field of view. In reality, machine vision systems today can mimic human vision, perception and identification to a rather crude degree of complexity depending on the human instructions provided in the software code, however, almost all vision systems today are slow and are quite poor at identification, recognition, learning and adapting to bad images and errors, compared to the human brain. Also, most vision systems require objects to have a colour that provides a strong contrast with a background colour, in order to detect edges reliably. In summary, today's procedural-software-driven computer controllers are limited by the amount of programming and decision-making "intelligence" passed onto it by a human programmer or engineer, usually in the form of a single-threaded application or a complex list of step-by-step instructions executed in a continuous loop or triggered by sensor or communication "interrupts". This method of control is suitable for most repetitive applications, however, new types of computer architecture based on how the human brain works and operates is unchartered research area that needs exploration, modelling and experimentation in order to speed up shape or object recognition times and try to minimize the large amount of human effort currently required to program, set up and commission "intelligent" machines that are capable of learning new tasks and responding to errors or emergencies as competently as a human being.

The biggest challenge for the 21st century is to make robots and machines "intelligent" enough to learn how to perform tasks automatically and adapt to unforeseen operating conditions or errors in a robust and predictable manner, without the need for human guidance, instructions or programming. In other words: "Create robot controllers that are fast learners, able to learn and perform new tasks as easily and competently as a human being just by showing it how to do something only once. It should also learn from its own experiences, just like a young child learning and trying new skills." Note that a new-born baby knows practically nothing but is able to learn so many new things automatically, such as sounds, language, objects and names. This is a "tall order" and sounds very much like what you would expect to see in a "Star Wars" or "Star Trek" science fiction film, but who would have thought, 40 years ago, that most people could be instantly contacted from almost anywhere with portable mobile phones, or that you could send photos and letters to friends and family members instantly to almost anywhere in the world, or that programmable computers would be smaller than your fingernails? Who ever thought that a robot can automatically perform Cochlear surgery and detect miniscule force and torque changes as a robotic drill makes contact with a thin soft tissue membrane which must not be penetrated? (A task that even the best human surgeons cannot achieve consistently with manual drilling tools) Who would have imagined that robots would be assembling and creating most of the products we use every day, 40 years ago? At the current accelerating rate of knowledge growth in the areas of robotics and mechatronics, it is not unreasonable to believe that "the best is yet to come" and that robotics technology will keep on improving to the point where almost all physical jobs will be completely automated and at very low cost. Mobile or "field" robotics is also a rapidly growing field of research, as more

applications for robotic and mechatronic engineering technology are found outside the well-structured and carefully controlled environments of indoor factories and production lines.

Technological development is now at the stage where robots can be programmed to automatically plant and harvest food at low cost to end world hunger, engage in cooperative construction work to erect buildings and low-cost modular homes to house the poor, perform remote surveying and video surveillance (land, sea, air & on other planets), automatically build space stations or bases on the Moon or on Mars, perform fully automated mining operations deep underground, safely transport people in flying aerial vehicles to avoid slow road traffic, mow your lawn and recharge itself, guide blind people to their destinations using GPS or machine vision and save human beings from the strain and boredom of highly repetitive production work in factories. In fact, there is no limit to where practical robotic technologies may be used to improve how people work and live. Rather than destroying factory and production jobs, robots are providing more opportunities for people to upgrade their skills to become technicians or robot operators who are spared the difficulties of strenuous, repetitive and often boring manual labour. We are not yet at the level of robotic automation depicted in films like "iRobot" or cartoons like "The Jetsons", where humanoid robots roam the streets freely, however, modern society appears to be headed in that direction and robots of all types could play an increasingly important role in our daily lives, perhaps improving the way we work, shop and play.

The one truth that faces us all is that life is short and it is important to do as much "good" as possible in the limited time that we are alive. It is important to leave behind a better world for future generations to inherit and enjoy so that they do not suffer unnecessary burdens, physical hardships, expensive education, poor employment opportunities or very high costs of living that leave them with little or no savings or financial incentives to work. Robotic and mechatronic engineers, researchers and educators are in an excellent position to help leaders in education, business and politics to understand and realize the benefits of promoting robotic applications. All it takes is the desire to do good for others and the kind of burning enthusiasm and zeal that makes it difficult to sleep at night! Unfortunately, most Universities do not teach engineers how to be effective at developing, selling, promoting and commercializing new technologies, good ideas and useful inventions that could change the world. Many education systems today still value "rote learning" and memorization skills over "Problem Based Learning" projects or design-and-build activities that promote creativity. It is this kind of "inventor's mindset" and "entrepreneurial spirit" which motivated the great inventors and scientists of the past to keep tinkering, exploring and experimenting with new ideas and concepts which showed good potential for being useful and practical in the real world. In the "spirit of discovery", robotic and mechatronic engineers and researchers around the world are working hard, relentlessly pursuing their research goals in order to discover, develop and test a new great idea or a new technological breakthrough that could make a significant impact or improvement to the world of robotics and mechatronics. Sometimes this work is arduous and difficult, requiring a great deal of patience and perseverance, especially when dealing with many failures. In fact, good results cannot always be guaranteed in new "cutting edge" research work.

Despite much frustration, the veteran researcher becomes adept at learning from past mistakes, viewing each failure as a necessary, vital "learning experience" and an opportunity to make progress towards different goals which may present more interesting questions. This kind of research and investigative work brings great joy when things are going well as planned. I have laughed many times when very conservative research engineers jump and even yell with joy when their experiments finally work for the first time after many failures. The truth is, robotics and mechatronic engineering is very addictive and enjoyable because continuous learning and solving challenging problems with a variety of intelligent people makes every day different, unpredictable and fun. Is technological change happening too fast? Advances in tools and devices are now happening at such a rapid pace that often, by the time students learn a particular type of software or piece of hardware, it is probably already obsolete and something new and better has replaced it already. Today, it is now virtually impossible for an engineer to be an expert in all areas of robotics and mechatronics engineering, however, it is possible to grasp the fundamentals and become an effective system integrator, able to bring together many different forms of technology to solve problems, and you will see plenty of evidence of this type of problem solving in this book. Mechatronic and robotic automation engineers are becoming increasingly dependent on using "off the shelf" devices, components and controllers. Using such commercially available components saves a great deal of development time and cost, allowing system developers to focus on accomplishing the tasks of designing, building and testing complete automation systems or manipulators customized for specific applications. Perhaps the most important learning skill for a mechatronic or robotics engineer is the ability to ask the right questions which could lead to the right answers.

This book covers a wide range of topics relating to advanced industrial robotics, sensors and automation technologies. Although being highly technical and complex in nature, the papers presented in this book represent some of the latest "cutting edge" technologies and advancements in industrial robotics technology. This book covers topics such as networking, properties of manipulators, forward and inverse robot arm kinematics, motion path-planning, machine vision and many other practical topics too numerous to list here. The authors and editors of this book wish to inspire people, especially young ones, to get involved with robotic and mechatronic engineering technology and to develop new and exciting practical applications, perhaps using the ideas and concepts presented herein. On behalf of all the authors and editors who have displayed a great deal of passion, tenacity and unyielding diligence to have this book completed on time, I wish you all the best in your endeavours and hope that you find this book helpful and useful in your research and development activities.

Please feel free to contact the publishers and let us know your thoughts.

Editor
Dr. Sam Cubero
Head & Course Coordinator
Mechatronic Engineering
Curtin University of Technology
Australia

Robotic Body-Mind Integration: Next Grand Challenge in Robotics

K. Kawamura, S. M. Gordon and P. Ratanaswasd

1. Introduction

During the last thirty years, the fields of robotics, cognitive science and neuroscience made steady progress fairly independently with each other. However, in a quest to understand human cognition and to develop embedded cognitive artifacts like humanoid robots, we now realize that all three fields will benefit immensely by collaboration. For example, recent efforts to develop so-called intelligent robots by integrating robotic body, sensors and AI software led to many robots exhibiting sensorimotor skills in routine task execution. However, most robots still lack robustness. What, then, would be the next challenge for the robotics community? In order to shed light on this question, let's briefly review the recent history of robotic development from design philosophy point of view.

In recent years, design philosophies in the field of robotics have followed the classic dialectic. Initial efforts to build machines capable of perceiving and interacting with the world around them involved explicit knowledge representation schemes and formal techniques for manipulating internal representations. Tractability issues gave rise to antithetical approaches, in which deliberation was eschewed in favor of dynamic interactions between primitive reactive processes and the world [Arkin, 1998] [Brooks, 1991].

Many studies have shown the need for both, motivating work towards hybrid architectures [Gat, 1998]. The success of hybrid architecture-based robot control led to wide-ranging commercial applications of robotics technologies. In 1996, a panel discussion was held at the IEEE International Conference on Robotic and Automation (ICRA) Conference to identify the grand research challenges for The Robotics and Automation Society for the next decade.

Figure 1 shows three grand challenges identified by the panel and the progress made in the last decade in each area.

Such an integration of robotic body, sensor and AI software led to a wide variety of robotic systems. For example, Sony's QRIO (see Figure 1) can dance and play a trumpet. The da Vinci robotic surgical system by Intuitive Surgical Inc. (www.intuitivesurgical.com) can assist surgeon in laparoscopic (abdominal) surgery.

- **The 1996 ICRA panel discussion**

  (IEEE Robotics and Automation Magazine, 3(4), Dec 10-16,1996)

    - Human-Robot Interface (HRI)
    - Modularity
    - System Integration

      Sony's QRIO

- **Much progress has been made since then**

    - Human-Robot Interface ➔ Vision, Voice, Gesture, Haptic, EMG, etc.
    - Modular / Evolutionary ➔ Multi-Agent Systems, BBDs
    - System Integration ➔ Integration of Body and Sensor

BBDs - Brain-Based Devices

Figure 1. Grand Challenges for Robotics and Automation.

Such robots are fluent in routine operations and capable of adjusting behavior in similar situations. We hypothesize, however, that robustness and flexibly responding to the full range of contingencies often present in complex task environments will require something more than the combination of these design approaches. Specifically, we see human's perception and cognitive flexibility and adaptability should be incorporated in the next generation of intelligent robots. We call this "robotic body-mind integration" in this paper. Thus, a fully cognitive robot should be able to recognize situations in which its reactive and reasoning abilities fall short of meeting task demands, and it should be able to make modifications to those abilities in hopes of improving the situation. These robots can be classified as *cognitive robots*.

Recently several national and international research programs were initiated to focus on "cognitive agents" [EU, 2004; DARPA, 2005; Asada, et al., 2006]. At ICAR2003 in Coimbra, Portugal, we proposed a cognitive robotic system framework (Figure 2) [Kawamura, et al, 2003a].

In this chapter, we will give details of our cognitive robot architecture with three distinctive memory systems: short-term and long-term memories and an adaptive working memory system will be described. Short-term and long-term memories are used primarily for routine task execution. A working memory system (MWS) allows the robot to focus attention on the most relevant features of the current task and provide robust operation in the presence of distracting or irrelevant events.
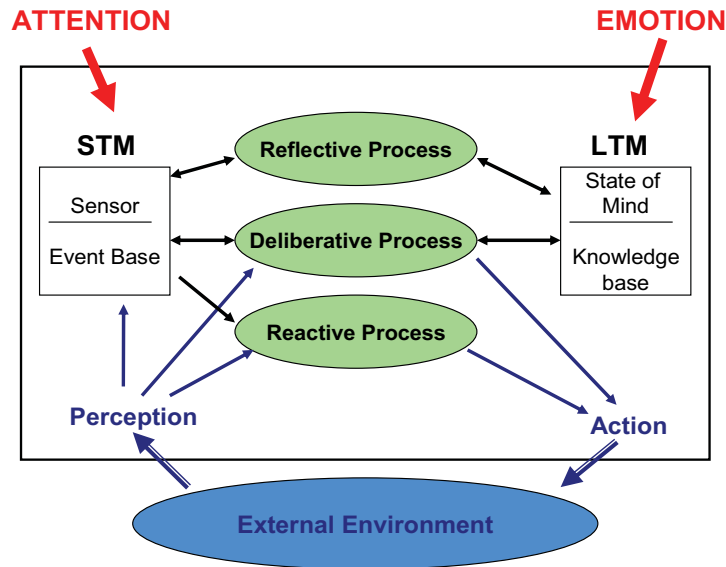
Figure 2. Framework for a cognitive robotic system.

2. Representative Cognitive Architectures in the US

Field of cognitive science has been interested in modeling human cognition for some time. Cognitive scientists study human cognition by building models that help explain brain functions in psychological and neuroscience studies. Over the last decades, many different cognitive architectures and systems have been developed by US cognitive scientists to better understand human cognition. In the following, we will briefly describe three of them. The first two were chosen for their popularity in the US and their generality. The third was chosen as an exemplary system to incorporate human perceptual and motor aspects in more specific ways to analyze complex cognitive tasks such as aircraft cockpit operation. All three have inspired our work.

2.1 ACT-R

ACT-R (Adaptive Character of Thought-Rational) [Anderson and Liebiere, 1998] is a cognitive architecture using production rules to be applied to problems of human cognitive and behavior modeling. It is based on The ACT-R theory of cognition. Within this architecture, one can develop ACT-R models for different cognitive tasks [Lovett, et al, 1999]. It includes multiple modules that correspond to different human cognitive functions, i.e. perception, motor and memory. Figure 3 shows (a) the functional structure of ACT-R and (b) how it works. "One important feature of ACT-R that distinguishes it from

other theories in the field is that it allows researchers to collect quantitative measures that can be directly compared with the quantitative measures obtained from human participants." [ACT-R, 2006] Successive versions of ACT-R have seen wide-spread applications to problems of cognitive and behavioral modeling. Anderson's group is extending the ACT-R architecture to show how visual imagery, language, emotion, and meta-cognition affect learning, memory and reasoning under the DARPA BICA (Biologically Inspired Cognitive Architecture) Program [DARPA, 2005].

(a)   (b)
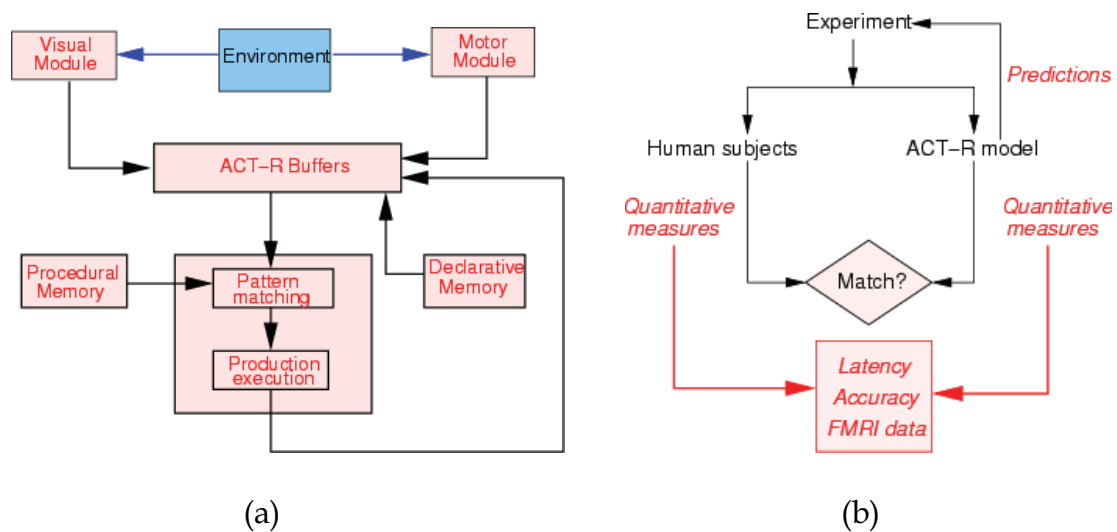
Figure 3(a). ACT-R architecture (b) How ACT-R works [ACT-R, 2006].

2.2 SOAR

Soar is a general purpose architecture designed as an unified theory of cognition by John Laird, et al [Laird, et al, 1987]. It is a production rule-based system based on the simple decision cycle – elaboration of state, selection of operator, and actions. Soar represents all cognitive activity by states. It has been applied commercially by Soar Technology Inc. Like the working memory system in our robot architecture, Soar's functional account of working memory emphasizes the important role of learning. Figure 4 shows the high-level description of the Soar Cognitive Architecture. Laird's group is now enhancing the Soar architecture by incorporating a comprehensive memory and learning system that includes the three types of human memory: procedural, semantic and episodic and emotion under the DARPA BICA (Biologically inspired Cognitive Architecture) Program [SOAR, 2006].

Learning in Soar is a by-product of impasse resolution. When an impasse is encountered, Soar creates a state space in which the goal is to resolve the impasse. Once the impasse is resolved, information about the resolution is trans-

formed into a new production rule. This new rule can then be applied whenever Soar encounters the situation again. The process of encoding and storing the newly learned rules is called "chunking". However, Soar's chunking is different from the term "chunk" used by cognitive neuroscientists when referring to human working memory. Soar's chunking is a learning method used to process information already present in the working memory for storage in the long-term memory. On the other hand in our architecture, as in human working memory, chunks refer to the arbitrary pieces of information stored in the long-term memory. (See Section 5.3.2 for details)
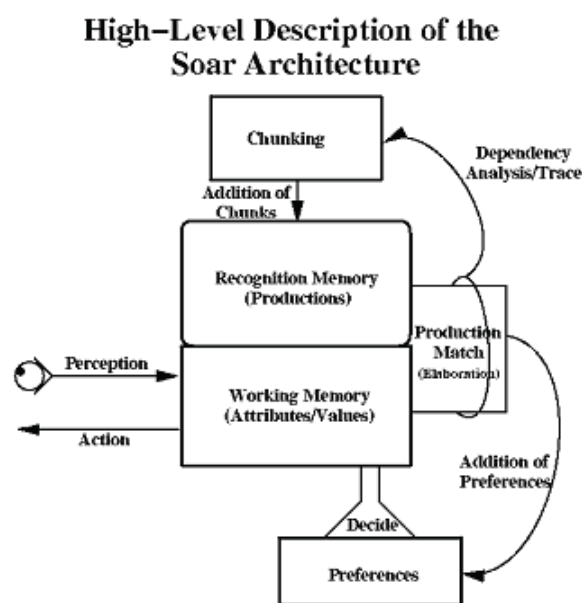
Figure 4. SOAR architecture adopted from [Wray, 2005].

2.3 EPIC

EPIC (Executive-Process/Interactive-Control) is a cognitive architecture designed to address the perceptual and motor aspects of human cognition [Kieras and Meyer, 1995]. It is designed to model human cognitive information processing and motor-perceptual capabilities. EPIC also uses a production system. EPIC has three types of simulated sensory organs: visual, auditory and tactile. Long-term memory consists of declarative and procedural memories. The cognitive processor populates working memory with procedural memory and actions are executed according to the production rules whose conditions are met. EPIC (Figure 5) was especially constructed for modeling complex cognitive activities associated with skilled perceptual-motor performance in task situations such as aircraft-cockpit operation and air-traffic control [Kieras, et al, 1999].
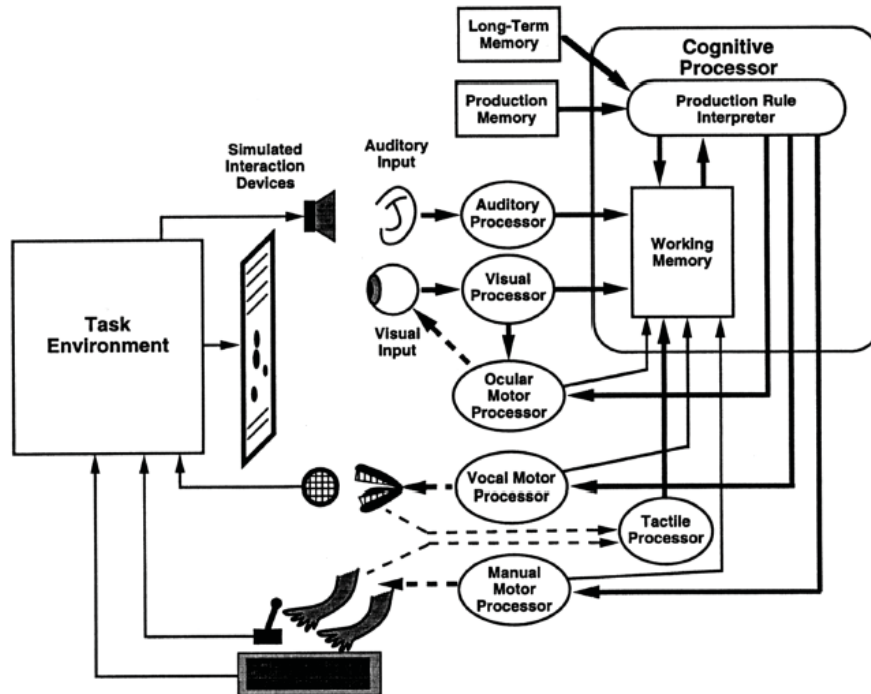
Figure 5. EPIC architecture [Meyer & Kieras, 1997].

3. Multiagent Systems

3.1 Multiagent Systems

In robotics, the term 'agent' is commonly used to mean an autonomous entity that is capable of acting in an environment and with other agents. It can be a robot, a human or even a software module. Since Minsky used the term 'agent' in *Society of Mind* [Minsky, 1985], the term 'multi-agent system' (MAS) – a system with many agents - is becoming more and more popular in artificial intelligence (where is better known as distributed artificial intelligence) [Ferber, 1999] and mobile robot communities (where it is often called multi-robot system). We adopted a multi-agent based system for our humanoid in the 1990s for its ease of modular development as we added more sensors and actuators and the need to integrate both the human and the robot in a unified human-robot interaction framework [Kawamura, et al, 2000].

3.2 Holons and Holonic Manufacturing Systems

In 1989, Japanese Government proposed a global collaborative program called the Intelligent Manufacturing Systems (IMS) [IMS, 1996] IMS was designed to advance a technical and organizational agenda in manufacturing to meet the challenges of global manufacturing in the 21st century. In 1996, we joined the Holonic Manufacturing System (HMS) project as a member of the US team within IMS. A holonic manufacturing system is a system having autonomous but cooperative components called holons [Koestler, 1967]. A holon can comprise other holons while, at the same time, being part of another holon. This gives rise to a holarchy where all holons automatically manage their component holons and simultaneously allow themselves to be managed within the holarchy [van Leeuwen, 1998]. The concept of holon and holarchy is similar to that of agent and agency [Minsky 1985]. Our goals within the HMS project were to develop a holonic system for batch manufacturing tasks [Saad, 1996] and to develop a control architecture for an prototype assembly holon (Figure 6), i.e. a humanoid robot [Shu, et al, 2000] using the Intelligent Machine Architecture described below. Unfortunately due to the lack of support from the US Government, we withdrew from IMS in 1999.
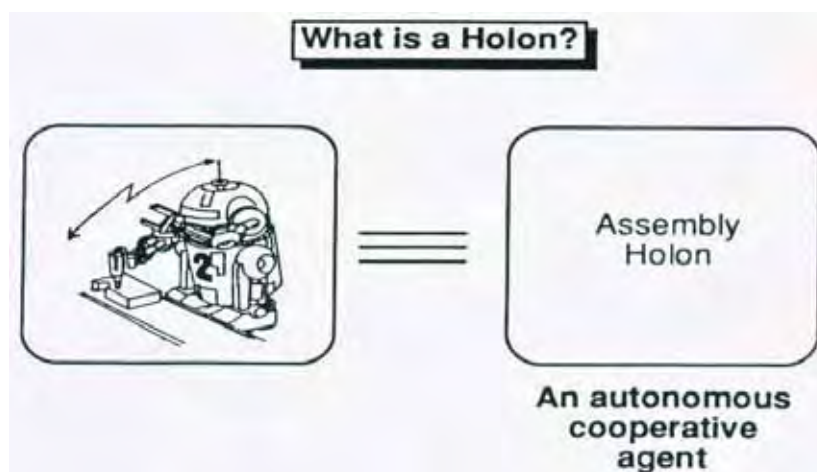
Figure 6. An assembly holon [Christensen, 1996]

3.3 Intelligent Machine Architecture

A humanoid robot is an example of a machine that requires intelligent behavior to act with generality in its environment. Especially in interactions with humans, the robot must be able to adapt its behaviors to accomplish goals safely. As grows the complexity of interaction, so grows the complexity of the software necessary to process sensory information and to control action pur-

posefully. The development and maintenance of complex or large-scale software systems can benefit from domain-specific guidelines that promote code reuse and integration. The Intelligent Machine Architecture (IMA) was designed to provide such guidelines in the domain of robot control [Kawamura, et al, 1986; Pack, 1998]. It is currently used to control ISAC. [Olivares, 2004; Olivares, 2003; Kawamura, et al, 2002].

IMA consists of a set of design criteria and software tools that supports the development of software objects that we call "agents". An agent is designed to encapsulate all aspects of a single element (logical or physical) of a robot control system. A single hardware component, computational task, or data set is represented by an agent if that resource is to be shared or if access to the resource requires arbitration. Agents communicate through message passing. IMA facilitates coarse-grained parallel processing. The resulting asynchronous, parallel operation of decision-making agents simplifies the system model at a high level. IMA has sufficient generality to permit the simultaneous deployment of multiple control architectures. behavior can be designed using any control strategy that most simplifies its implementation. For example, a simple pick and place operation may be most easily implemented using a standard Sense-Plan-Act approach, whereas visual saccade is more suited to subsumption, and object avoidance to motion schema.

IMA works very well to promote software reuse and dynamic reconfiguration. However, the large systems built with it have experienced scalability problems on two fronts. First, as the system exceeds a certain level of complexity it is difficult for any programmer to predict the interactions that could occur between agents during actual operation. This level seems to be higher than for a direct, sequential program. But that level has been reached in the development of ISAC. The other scalability problem may or may not be a problem with IMA itself but may be an inevitable consequence of increasing complexity in a system based on message passing. The asynchronous nature of message passing over communications channels with finite bandwidth leads to system "lock-ups". These occur with a frequency that apparently depends on the number of agents in the system. It may be possible to minimize this problem through the use of system-self monitoring or through a process of automatic macro formation. For example, the system could, through a statistical analysis, recognize the logical hierarchies of agents that form repeatedly within certain tasks or under certain environmental conditions. A structure so discerned could be used to "spin off" copies of the participating agents. These could be encapsulated into a macro, a compound agent that optimizes the execution and inter-process communications of the agents involved. For such an approach to be most useful, it should be automatic and subject to modification over time frames that encompass several executions of a macro.

4. ISAC Cognitive Architecture

IMA encapsulates the functions of hardware, low-level controllers, and basic sensory processing into independent, reusable units. This abstraction of details away from control loops, image operators, signal processing algorithms, and the like, enables programming to occur at the level of purposeful actions and environmental features. Actuators are supplanted by actions. Raw sensory data are replaced by features. These abstractions are the keys of ISAC's abilities and are implemented using IMA agents. The functions of actuators are encapsulated within control agents. Each agent interfaces to its corresponding hardware resource and provides control interface to other agents. In the current system, there are two arm agents, two hand agents, and a head agent. ISAC's perceptual system includes a number of sensors. Each sensor is assigned an IMA agent that processes the sensory inputs and stores the information based on the type of perception. For visual inputs, there are visual agents that perform perception encoding, such as color segmentation, object localization and recognition, motion detection, or face recognition. Other inputs include sound localizations and sound recognition agents. Each of the individual tasks is encapsulated by an atomic agent, such as find-colored-object, reach-to-point, and grasp-object agents. At the higher level, ISAC's cognitive abilities are implemented using two compound agents: the Self Agent which represents ISAC's sense of self, and is responsible mostly for task execution, and the Human Agent which represents the human who ISAC is currently interacting.

Memory structures are utilized to help maintain the information necessary for immediate tasks and to store experiences that can be used during decision making processes. Sensory processing agents write data to the Sensory EgoSphere (SES) which acts as a short-term memory (STM) and interface to the high-level agents [Peters, et al., 2001]. The long-term memory (LTM) stores information such as learned skills, semantic knowledge, and past experience (episodes) for retrieval in the future. As a part of LTM, Procedural Memory (PM) holds motion primitives and behaviors needed for actions, such as how to *reach to a point* [Erol et al, 2003]. Behaviors are derived using the Spatio-Temporal Isomap method proposed by Jenkins and Matarić [Jenkins & Mataric, 2003]. Semantic Memory (SM) is a data structure about objects in the environment. Episodic Memory (EM) stores past experience including goals, percepts, and actions that ISAC has performed in the past. The Working Memory System (WMS) is modeled after the working memory in humans, which holds a limited number of "chunks" of information needed to perform a task, such as a phone number during a phone- dialing task. It allows the robot to focus attention on the most relevant features of the current task, which is closely tied to the learning and execution of tasks. Figure 7 depicts the key IMA agents and the memory structure within the ISAC cognitive architecture.
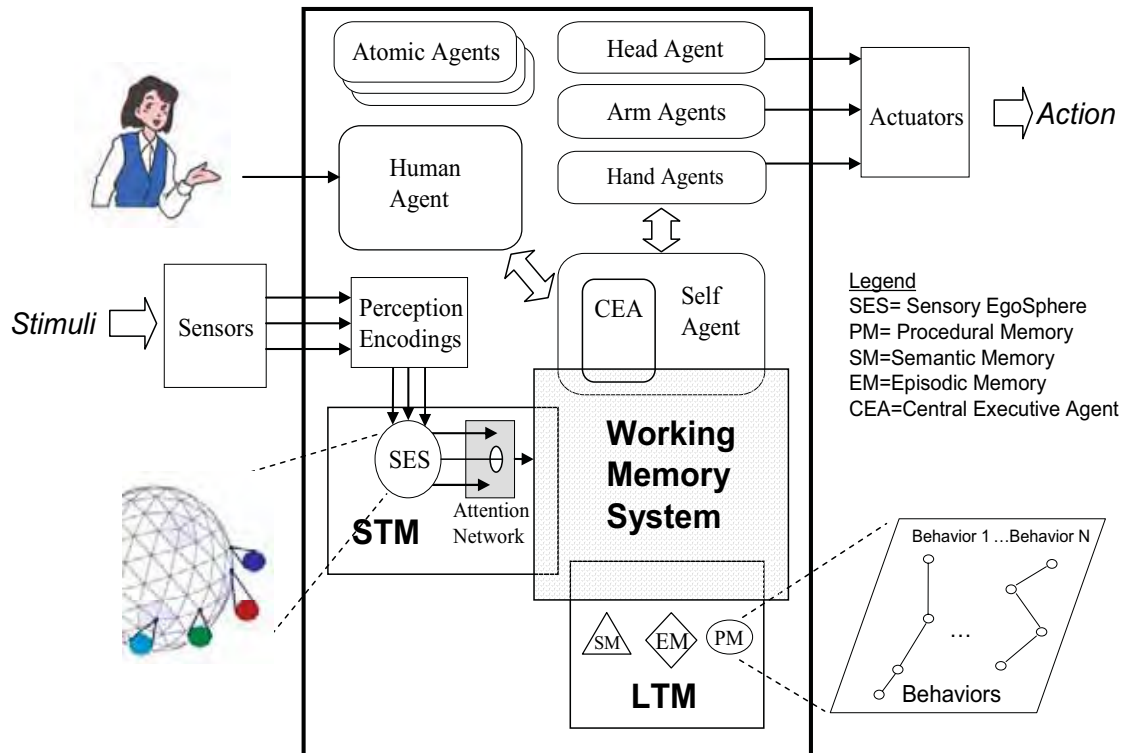
Figure 7. Multiagent-based cognitive robot architecture.

4.1 Self agent

According to Hollnagel and Woods, a cognitive system is defines as "*an adaptive system which functions using knowledge about itself and the environment in the planning and modification of actions*" [Hollnagel, 1999]. Key words here are <u>knowledge about itself</u>. In our architecture, the Self Agent (SA) represents robot itself. It is responsible for ISAC's cognitive activities ranging from sensor signal monitoring to cognitive or executive control (see Section 6.1 for detail discussions on cognitive control) and self reflection. "Cognitive control is needed in tasks that require the active maintenance and updating of context representations and relations to guide the flow of information processing and bias actions." [Braver, et al, 2002] Figure 8 is a diagram of the Self Agent and the associated memory structure. The Description Agent provides the description of atomic agents available in the system in terms of what it can or cannot do and what is it doing. The First-order Response Agent (FRA) selects the humanoid's actions according to (1) the percepts in the environment and (2) the commands/intentions of the person with whom the robot is currently interacting. The intentions are supplied by the Human Agent (see Section 4.2 for details) and interpreted by the Intention Agent. The Emotion Agent keeps

track of robot internal variables that will be used during action selection, attention and learning. The Activator Agent invokes atomic agents to handle temporal integration for the selected actions. The Central Executive Agent (CEA) working closely with the Working Memory System and the other SA agents provides cognitive control functions for ISAC. CEA is described in detail in Section 6.2.
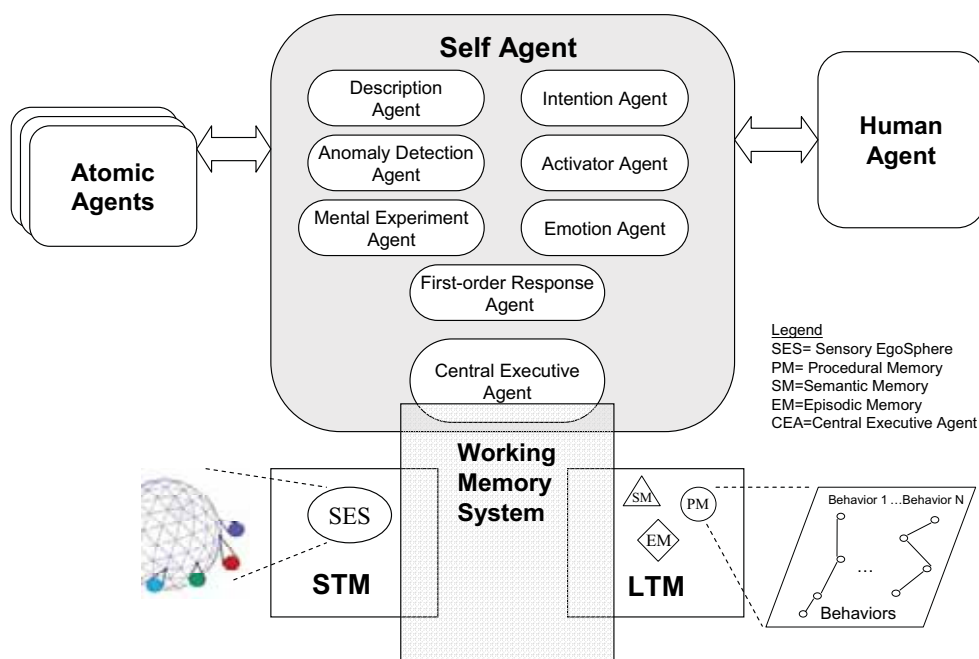
Figure 8. Self Agent and associated memory structure.

A key function of any cognitive robot must be is *self-reflection*. Self reflection will allow the robot to reason its own abilities, cognitive processes, and knowledge [Kawamura, et al, 2003b]. As part of an initial effort to incorporate self-reflective process into ISAC, we are proposing two agents: the Anomaly Detection Agent (ADA) and the Mental Experimentation Agent (MEA) within the Self Agent. ADA will monitor the inputs and outputs of the atomic agents in the system for fault detection. And when an impasse is raised and if the CEA fails to find an alternative solution, MEA will conduct a search through the space of control parameters to accomplish the task in "simulated mode" The concept of self reflection is closely related to that of self awareness (Fig. 9) and machine consciousness [Holland, 2003].
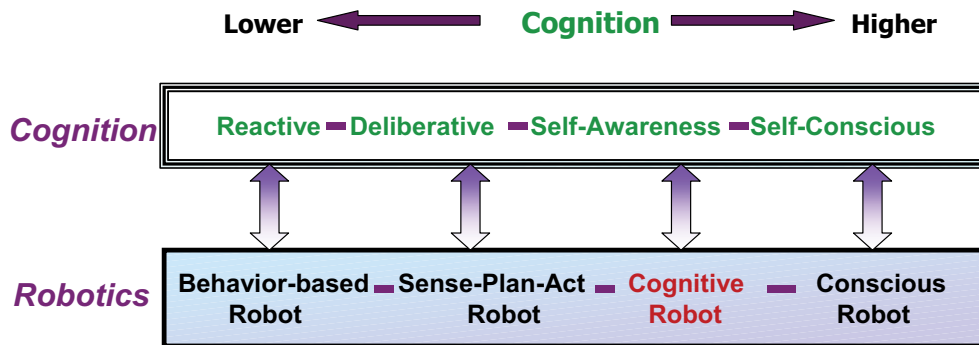
Figure 9. Spectrum of cognition in robotics.

4.2 Human agent

The Human Agent (HA) comprises a set of agents that detect and keep track of human features and estimate the intentions of a person within the current task context. It estimates the current state of people interacting with the robot based on observations and from explicit interactions (Figure 10 a and b) [Rogers, 2004]. The HA receives input from various atomic agents that detects physical aspects of a human (e.g., the location and identity of a face). The HA receives procedural information about interactions from the SA that employs a rule set for social interaction. The HA integrates the physical and social information with certain inferred aspects of the cognitive states of interacting humans, such as a person's current intention.

The HA processes two types of human intentions. An expressed intention is derived from speech directed toward ISAC, e.g., greetings and requests from a human. Inferred intentions are derived through reasoning about the actions of a person. For example, if a person leaves the room, ISAC assumes it means that the person no longer intends to interact, therefore, it can reset its internal expectations.

The Human Agent's assessment of how to interact is passed on to the SA. The SA interprets the context of its own current state, e.g. current intention, status, tasks, etc. This processing guides ISAC in the selection of socially appropriate behaviors that lead towards the ultimate goal of completing tasks with (or for) humans.

(a)
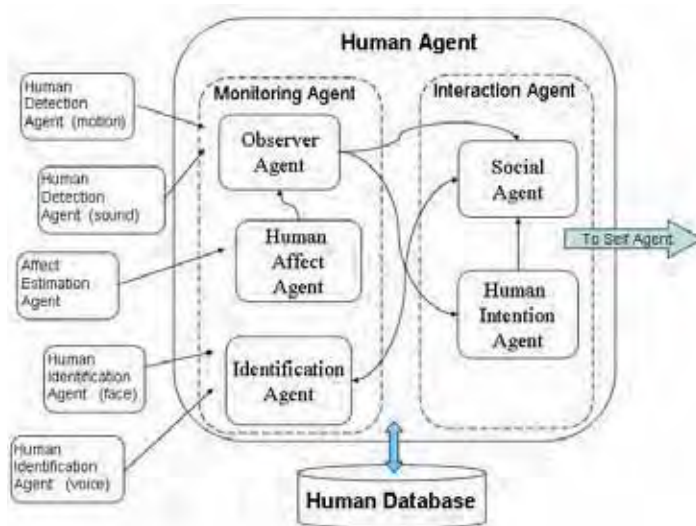
(b)

Figure 10. (a) ISAC interacting with humans and (b) Human Agent and associated atomic agents.

5. Memory Structure

ISAC's memory structure is divided into three classes: Short-Term Memory (STM), Long-Term Memory (LTM), and the Working Memory System (WMS). The STM holds information about the current environment while the LTM holds learned behaviors, semantic knowledge, and past experience, i.e., episodes. The WMS holds task-specific STM and LTM information and streamlines the information flow to the cognitive processes during the task.

5.1 Short-term memory: The Sensory EgoSphere

Currently, we are using a structure called the Sensory EgoSphere (SES) to hold STM data. The SES is a data structure inspired by the egosphere concept as defined by Albus [Albus, 1991] and serves as a spatio-temporal short-term memory for a robot [Peters, et al, 2001; Hambuchen, 2004]. The SES is structured as a geodesic sphere that is centered at a robot's origin and is indexed by azimuth and elevation.

The objective of the SES is to temporarily store exteroceptive sensory information produced by the sensory processing modules operating on the robot. Each vertex of the geodesic sphere can contain a database node detailing a detected stimulus at the corresponding angle (Figure 11).
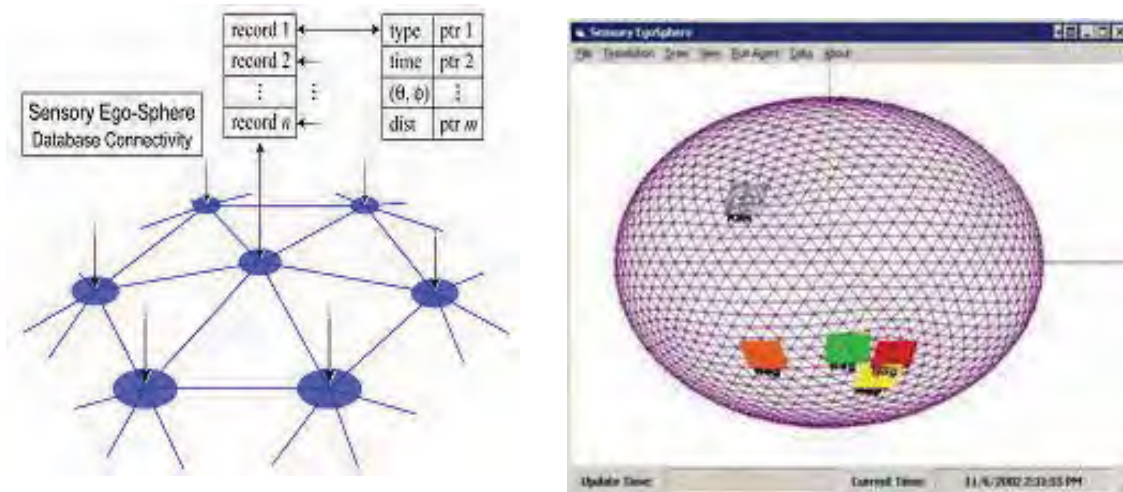
Figure 11. Mapping of the Sensory EgoSphere and topological mapping of object locations.

Memories in the SES can be retrieved by angle, stimulus content, or time of posting. This flexibility in searching allows for easy memory management, posting, and retrieval.

The SES is currently being used on ISAC (Figure 12a), and was installed on Robonaut (Figure 12b) at NASA's Johnson Space Center in Houston several years ago by members of our research group.

(a)   (b)
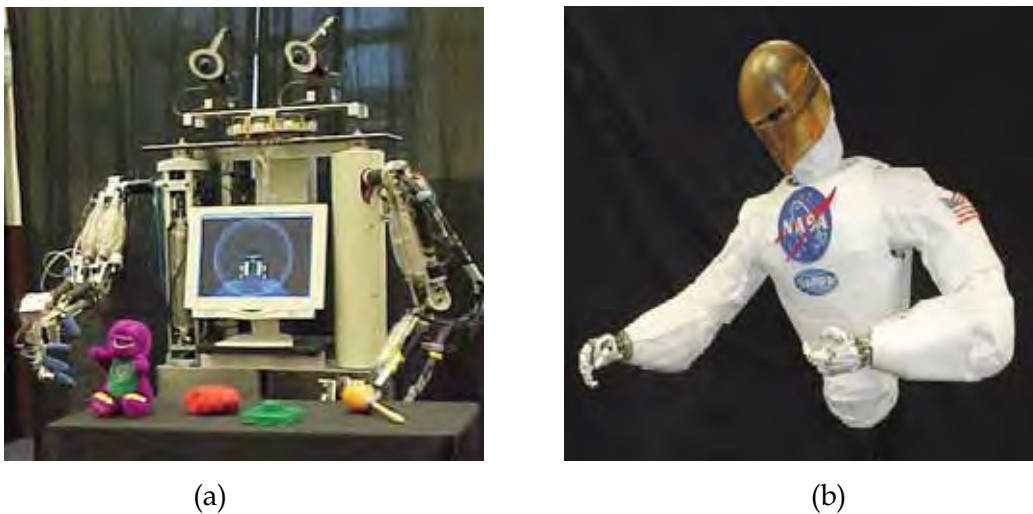
Figure 12(a). ISAC showing SES screen, (b) NASA's Robonaut.

5.2 Long-Term Memory: Procedural, Semantic and Episodic Memories

LTM is divided into three types: Procedural Memory, Semantic Memory, and Episodic Memory. Representing information such as *skills, facts learned* as well as *experiences gained* (i.e. *episodes*) for future retrieval.

The part of the LTM called the Procedural Memory (PM) holds behavior information . Behaviors are stored in one of two ways: as motion primitives used to construct behaviors or as full behavior exemplars used to derive variant motions.

Using the first method, stored behaviors are derived using the spatio-temporal Isomap method proposed by Jenkins and Mataric [Jenkins, et al, 2003]. With this technique motion data are collected from the teleoperation of ISAC. The motion streams collected are then segmented into a set of motion primitives. The central idea in the derivation of behaviors from motion segments is to discover the spatio-temporal structure of a motion stream. This structure can be estimated by extending a nonlinear dimension reduction method called Isomap [Tenenbaum, 2000] to handle motion data. Spatio-temporal Isomap dimension reduction, clustering and interpolation methods are applied to the motion segments to produce Motion Primitives (Figure 13a). Behaviors are formed by further application of the spatio-temporal Isomap method and linking Motion Primitives with transition probabilities [Erol, et al, 2003].

Motions recorded using spatio-temporal Isomap are stored in a separate manner as shown in Figure 13(b). At the top of this structure, behavior descriptions will be stored which will allow us to identify what each behavior can contribute to solving a given motor task. Each entry in the behavior table will contain pointers to the underlying motion primitives.

(a)

(b)
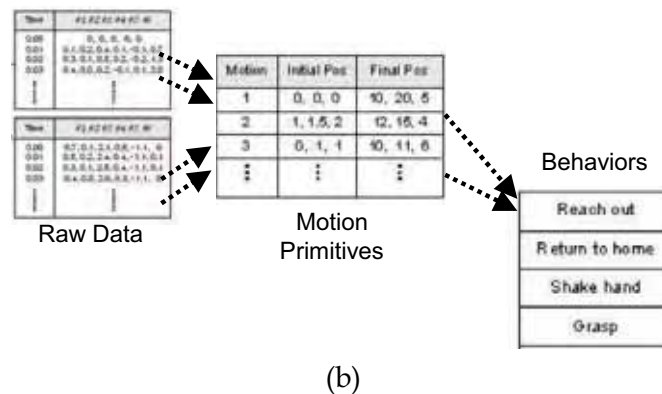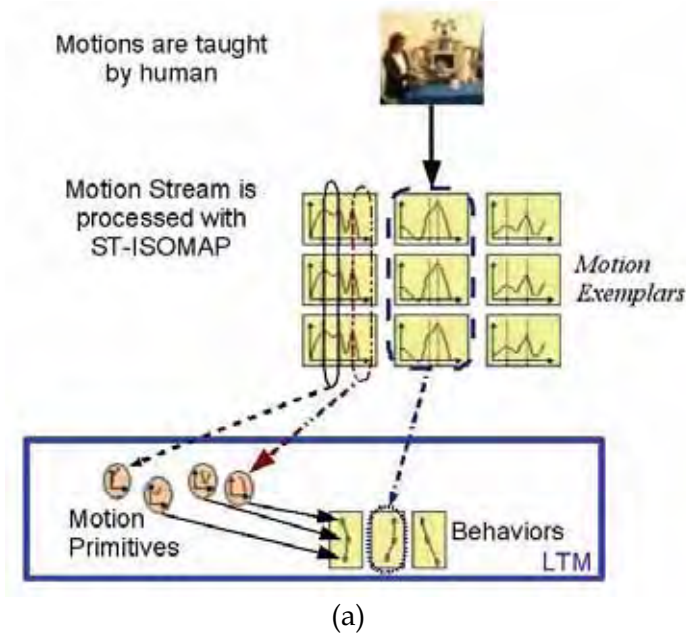
Figure 13(a). Derivation of Procedural Memory through human-guided motion stream and (b) Structure of Procedural Memory data unit.

The latter method stores behaviors using the Verbs and Adverbs technique developed in [Rose, et al, 1998]. In this technique, exemplar behavior motions are used to construct *verbs* while parameters of the motions are termed *adverbs*. An important aspect in storing and re-using a motion for a *verb* is the identification of the *keytimes* [Spratley, 2006; Rose, et al, 1998] of the motion. The keytimes represent significant structural breaks in the particular motion. For the Verbs and Adverbs technique to function properly individual motions for the same verb must have the same number of keytimes and each keytime must have the same significance across each motion. Figure 14(a) shows keytimes for three example motions. The example motions are recording of the same motion, three different times. This information is used to create the verb, *hand-*

shake. The keytimes in this example are derived by analyzing the motions using a technique called Kinematic Centroid [Jenkins, et al, 2003]. The x-axis represents the normalized point index for each motion. The y-axis represents the Euclidian distance of the kinematic centroid of the arm from the base of the arm.
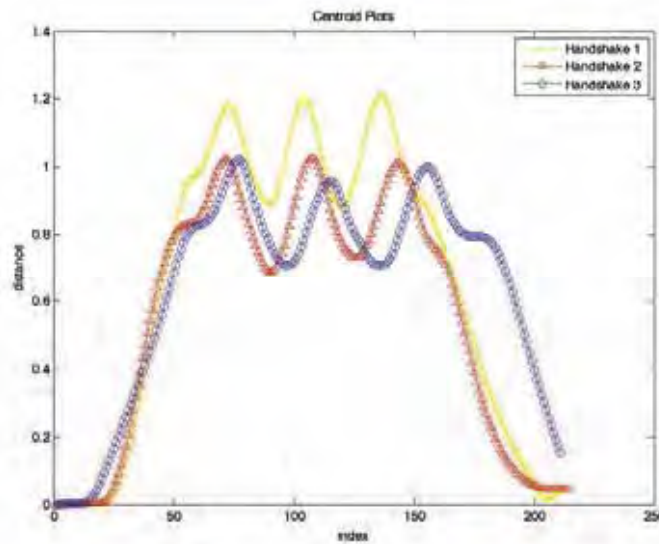
(a)
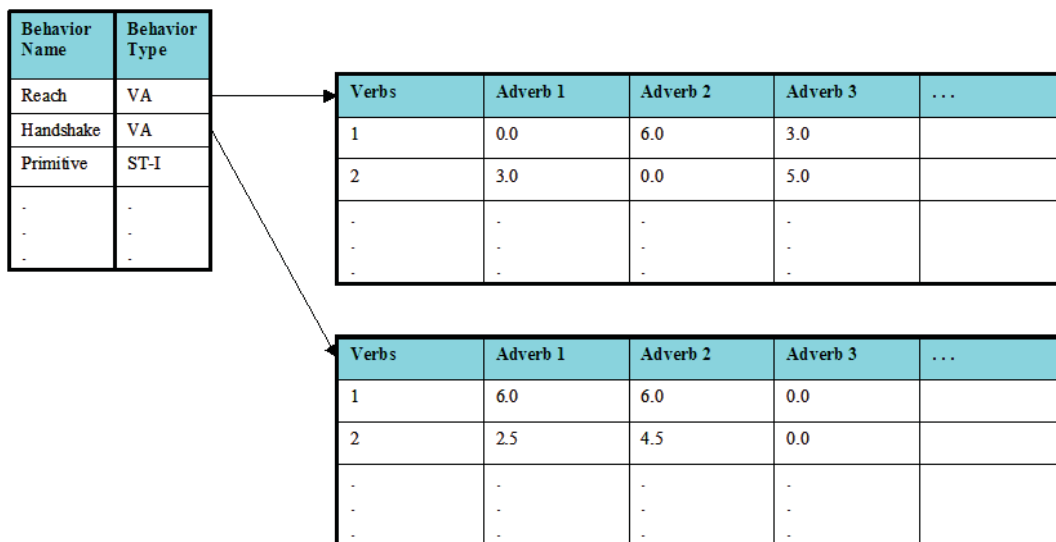
(b)

Figure 14 (a). Example motions and their keytimes [Spratley, 2006], (b) Structure of PM data representation for Verbs and Adverbs.

Each verb can have any number of adverbs, each of which relate to a particular space of the motion. For example, the verb *reach* could have two adverbs: the first related to the direction of the *reach* and the second related to the distance from ISAC's origin that the particular motion is to extend. Extending this example, adverbs could be added to include features from any other conceivable space of the motion, such as the strength of the motion or the speed of the motion. Stored in the LTM are the verb exemplars and the adverb parameters for each verb. New motions such as *reaching*, or *handshaking* are interpolated by ISAC at run time using the new (desired) adverb values.

Figure 14(b) depicts the manner in which behaviors are stored in LTM using Verbs and Adverbs. For each entry in PM, the motion and storage types are recorded. The next entry holds pointers to the verb information and the final entries hold the adverb values.

5.3 Attention and the Working Memory System

5.3.1 Attention

Attention is a sensory/cognitive mechanism to limit the amount of information needed to be manipulated by the brain for task execution. It "allows the brain to concentrate only on particular information by filtering out distracters from a desired target object or spatial location by amplification of the target representations." [Taylor and Fragopanagos, 2004] Attention can be goal-oriented during task execution such as searching for an object or it can be reactive in salience events such as when hearing a loud sound.

Attentional function in ISAC is implemented using the Attention Network which monitors both task relevant sensory data and unexpected yet salient sensory data on the Sensory EgoSphere (SES) [Hambuchen, 2004]. As sensory processors report all exteroceptive events to the SES, the direction of attention to external sensory events are also available through SES nodes (Figure 15). As multiple events are registered in a common area, activation increases around a central node. Nodes that receive registration from task- or context-related events have their activations increased by the Attention Network. The Attention Network selects the node with the highest activation as the focus of attention. Sensory events that contributed to this activation are selected and those that fall within a specified time range of each other are passed into the working memory.

Besides level of activation, the Attention Network also pays attention to percepts on SES with high emotional salience. When a percept is assigned high emotional salience, the Attention Network selects the percept as the focus of attention. Emotional salience is provided by the Emotion Agent, a part of the Self Agent. Its implementation, including attention based on emotional salience is described in Section 7.2.
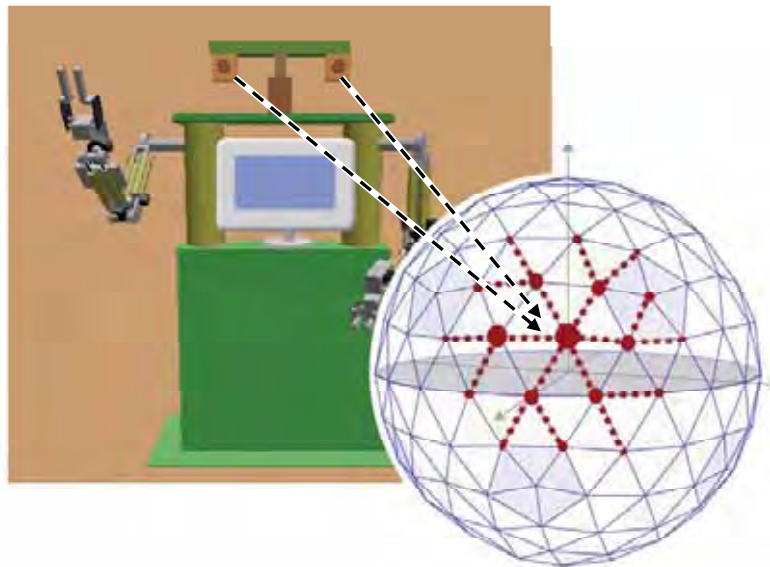
Figure 15. The attention network's assignment of FOA at the center node among events registered in a common area on SES [Hambuchen, 2004].

5.3.2 Working memory system

There is much evidence for the existence of working memory in primates [Funahashi, et al, 1994; Miller, et al, 1996]. Such a memory system is closely tied to the learning and execution of tasks, as it contributes to attention, learning and decision-making capabilities by focusing on task-related information and by discarding distractions [O'Reilly, et al, 1999; Baddeley, 1986; Baddeley, 1990]. The working memory in humans is considered to hold a small number of "chunks" of information needed to perform a task such as retaining a phone number during dialing.

Inspired by the working memory models developed in cognitive science and neuroscience, the Working Memory System (WMS) in robots was designed to provide the embodiment necessary for robust task learning and execution by allowing ISAC to focus attention on the most relevant features of the current task [Gordon & Hall, 2006].

WMS in our cognitive architecture was implemented using the Working Memory Toolkit (WMtk) based on the computational neuroscience model of working memory [Phillips, 2005]. This toolkit models the function of dopamine cells in human brains using a neural net-based temporal difference (TD) learning algorithm [Sutton, 1988]. The toolkit has a function to learn to select and hold on to "chunks" of information that are relevant to the current task based on future expected reward from processing these chunks. These chunks include behaviors, current percepts, and past episodes. Figure 16 illustrates the current WMS structure and associated system components. A simulated delayed saccade task using WMtk was reported by Philips and Noelle [Philips,

2006]. Section 7.1 in this chapter details working memory training and task learning conducted on ISAC.
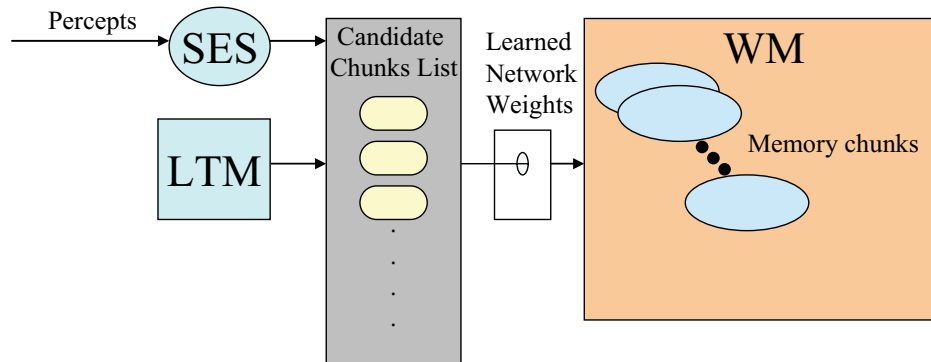
Figure 16. Structure of the working memory system.

6. Cognitive Control and Central Executive Agent

6.1 Cognitive Control

Cognitive control in humans is a part of executive functions (such as planning and abstract thinking) within the frontal lobes in the human brain [Stuss, 2002]. Cognitive control is "the ability to consciously manipulate thoughts and behaviors using attention to deal with conflicting goals and demands" [O'Reilly, et al, 1999] [MacLeod and Sheehan, 2003]. As levels of human activities range from reactive to full deliberation, cognitive control allows humans to inhibit distractions and focus on the task at hand including task switching. According to researchers in neuroscience, human cognitive control is performed through the working memory in the pre-frontal cortex (PFC) [O'Reilly, et al, 1999; Braver and Cohen, 2000; MacDonald et al., 2000]. Cognitive control during task execution/switching requires the brain to perform executive functions including:

- Focus attention on task-related information
- Maintain and update goal information
- Inhibit distractions
- Shift between different level of cognition ranging from routine actions to complex deliberation
- Learn new responses in novel situations

Cognitive robots, then, should have the ability to handle unexpected situations and learn to perform new tasks. Also, cognitive control is expected to give

flexibility to the robot to reason and act according to stimuli under conflicting goals in dynamic environment. Realization of cognitive control functions for ISAC is currently pursued through the maintenance of task-related information in the working memory system through gating of information that could be passed into the decision-making mechanism called the Central Executive Agent discussed in Section 6.2.

Attention and emotions are known to play an important role in human' decision and task execution[Arbib, 2004]. Therefore, we are now adding attention and emotion networks to realize cognitive control for robots as shown in Figure 17 modified from [Miller, 2003].
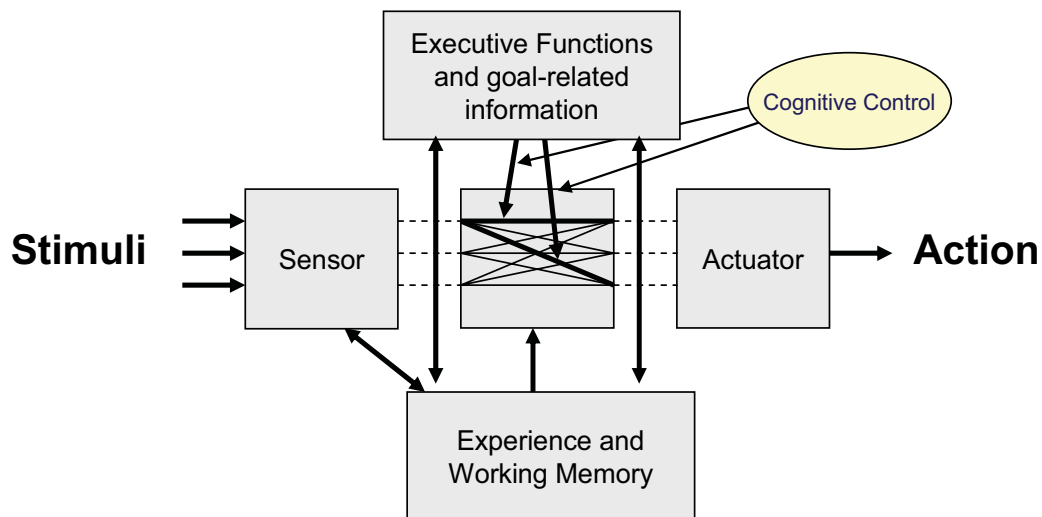
Figure 17. Concept of cognitive control modified from [Miller, 2003].

6.2 Central Executive Agent

ISAC's cognitive control function is modeled and implemented based on Baddeley and Hitch's psychological human working memory model [Baddeley, 1986]. Their model consists of the "central executive" which controls two working memory systems, i.e., phonological loop and visuo-spatial sketch pad (Figure 18).
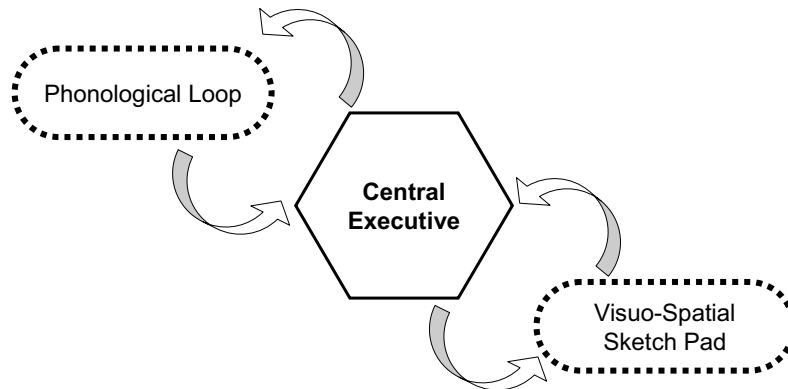
Figure18. A schematic Diagram of a multi-component model of working memory [Baddeley & Hitch, 1974].

In our cognitive architecture, an IMA agent called the Central Executive Agent (CEA) is responsible for providing cognitive control function to the rest of the system. It interfaces to the Working Memory System (WMS) to maintain task-related information (or "chunks") during task execution. Under the current design, CEA will have the four key functions: 1) situation-based action selection, 2) episode-based action selection, 3) control of task execution, and 4) learning sensory-motor actions. Each function will be realized through interaction between CEA, other IMA agents, and various memory systems as shown in Figure 19.
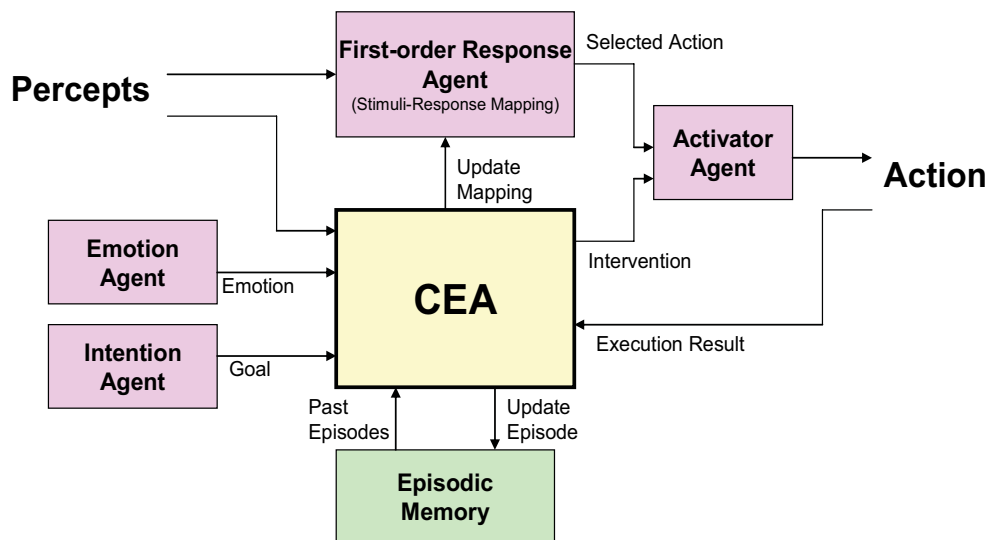
Figure 19. CEA's interactions with other processes.

Sensory inputs, stimuli and/or task commands, are encoded into percepts and posted on the SES. Only those percepts that have high *emotional salience* will

receive attention from the Attention Network and will be passed to WMS. These percepts, if not intervened, will cause corresponding actions to be selected according to embedded stimuli-response mapping. On the other hand, CEA selects actions using the combination of two paradigms. CEA will retrieve past episodes that are relevant to these percepts from the Episodic Memory. These past episodes contain actions used in past execution and results. The results of invoked actions from stimuli-response mapping could be a part of these episodes. CEA determines if the action to be executed is likely to produce a negative result. If so, CEA will intervene by suggesting a different action based on the current state of ISAC, current percepts, and action. Once the action is executed, CEA will update the stimulus-response mapping according to the execution result and the current task is then stored as a new episode in the Episodic Memory.

7. Experimental Results

7.1 Working Memory Training Experiment for Percept-Behavior Learning Tasks

The working memory system (WMS) is used to manage ISAC's memory focus during task execution. For simple tasks, WMS holds a small number of chunks of information related to the task. Typically on ISAC, the number of chunks loaded into WMS ranges from 2 to 4. For example, if ISAC were to be asked to "*reach* to the red bean bag", WMS would be responsible for loading two chunks of information: one chunk for the *reach* behavior and another chunk for the red bean bag percept. For more complex tasks (i.e. those that require more than 4 chunks of information) the tasks must be broken into simpler tasks and WMS is responsible for handling each simple task in turn as well as maintaining ISAC's focus on the long-term goal, the completion of the complex task.
WMS is not the tool that decomposes complex tasks into simple tasks. In the future, another agent such as CEA (section 6.2) must do this job. WMS, given the current state of ISAC, solely determines which chunks from LTM and STM to load into the system, in essence focusing ISAC on those pieces of information. Experiments utilizing WMS in this manner have already been conducted [Gordon, et al, 2006].
Current work with ISAC's WMS is centered on training a variety of different WMS for different types of tasks, such as::

- *Object Interaction* – Simple object interactions such as *reaching, pointing, tracking*, etc.
- *Human Interaction* – Performing such behaviors as *face tracking, greeting, handshaking, waiting for commands*, etc.

Figure 20 shows the architecture being used to train each these WMS.
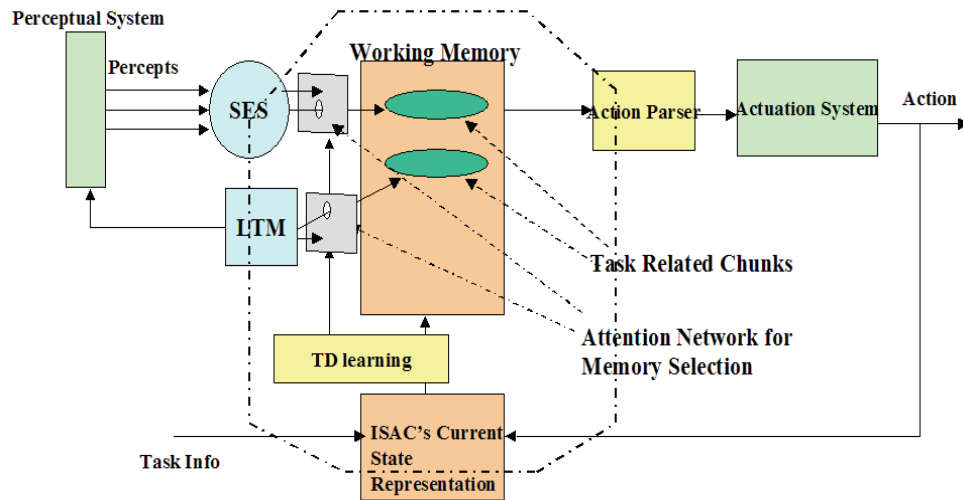
Figure 20. Control architecture used during working memory training.

During training, a reward rule is used to inform WMS how well it is performing. The reward rule captures whether or not the current chunks could be used to accomplish the task and how well the task has been accomplished.

7.1.1 Experimentation and Trials

Using the architecture shown in Figure 20, an initial experiment was designed for to test object interaction using working memory. Steps for this experiment are as follows:

1. ISAC is given certain initial knowledge (i.e. embedded ability and/or information)

   a) ISAC's perceptual system is trained to recognize specific objects. The information is stored in the semantic memory section of the LTM.
   b) Using the Verbs and Adverbs algorithm, ISAC is taught a small set of motion behaviors including how to *reach*, *wave*, and *handshake*.
   c) Figure 21 demonstrates ISAC performing these behaviors. This information is stored in the procedural memory section of the LTM.

Reach   Wave   Handshake
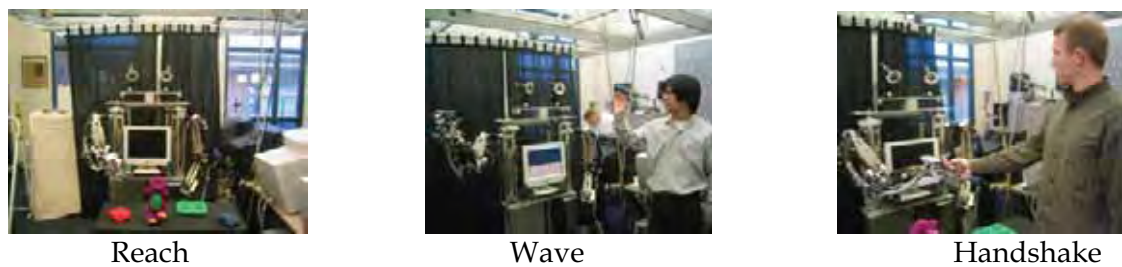
Figure 21. Sample configurations for reach, wave, and handshake.

2. Two bean bags are placed on a table as shown next in Figure 22.a.
3. ISAC is asked to "reach to the bean bag", although a specific bean bag is not specified.
4. ISAC's perceptual system will recognize the bean bags and post the information to SES.
5. WMS will focus on "chunks" of information necessary for accomplishing the task.
6. A reward is given based upon how well the action is completed.
7. Over time, ISAC learns the appropriate chunks to focus on from the SES and LTM.
8. Once ISAC has demonstrated that it has learned the most appropriate chunks to load into WMS (Figure 22.a), bean bags are rearranged (Figure 22.b) and ISAC is given the command "reach to the bean bag".
9. Real-time experiments were conducted after initial simulation trials (Figure 22.c).

When the bean bags are rearranged, ISAC should not necessarily reach to the same bean bag as before but should choose the *bean bag percept* from the SES that is the most appropriate. For this task the most appropriate bean bag is the nearest one. The combination of percept and behavior, or "chunks", will be loaded into the working memory and used to execute the action.

(a)   (b)   (c)
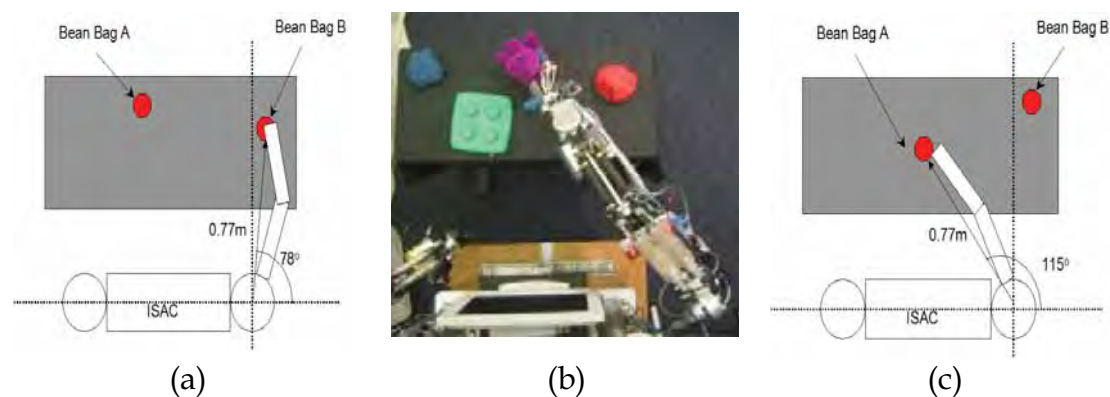
Figure 22 (a, b). Sample configurations for reaching and (c) actual experiment view.

The reward rule for this experiment is based on three criteria:

1. What is the degree of success for the behavior WMS chose to load?
2. How well did the object chosen by WMS meet the task criteria? e.g., focusing on any bean bag *vs.* focusing on another object.
3. How well is SAC able to act upon the object? e.g., in this experiment, could ISAC reach the bean bag?

In order to measure Reward Criterion #3, the reward was given based on the inverse proportion of the distance from ISAC's hand to the object. Reward Criteria #1 and #2 gave a discrete positive valued reward if the system chose appropriately. No preference (i.e., reward of 0) was the result if the system did not choose correctly. The values for the overall reward typically fell in the range of 0 – 400. Since it was desired to give negative reward to the system when it did not act appropriately, a negative weighting factor of –200 was added to the final reward to "tilt" the low values into the negative range.

Note that when using these reward criteria, it is possible to incorrectly reward the system for performing the task in less than an optimal manner. For example, if the system performs the behavior *handshake* or *wave* while focusing on the appropriate bean bag and if this action happens to bring the hand very close to the bean bag, then the system would receive a positive reward. In order to avoid this undesirable situation, more rules or knowledge are needed.

Initial trials for this experiment were performed off-line, in simulation, to speed-up the initial testing phase of the system. This simulation was set-up to remove the time-bottleneck of generating and performing motions. For the simulation, when ISAC needed to act on an object within the workspace, the motion was assumed to have been performed properly (Reward Criterion 3).

The action taken by ISAC was determined by what WMS currently believed was the best choice. In other words the action that WMS believed would yield the greatest reward. This system also contained an exploration percentage, specified as a part of initial knowledge that determined the percentage of trials that WMS chose a new or different action. This enabled WMS to always continue learning and exploring.

During initial research trials, simulation was not allowed to choose the same action more than twice. This constraint enabled a much quicker simulation time. Once the system finished exploration, the system was restarted with the learned information and given the task to "*reach* to the bean bag". For each arrangement (Figures 22a,b) the system chose appropriately to reach towards the correct bean bag, i.e. the nearest one. Table 1 shows the contents of ISAC's short-term and long-term memory systems during the training portion of the simulation.

SES	LTM
1. bean bag: location = (Figure 22.b), type = A	1. *reach*
2. bean bag: location = (Figure 22.a), type = B	2. *handshake*
3. blank	3. *wave*

Table 1. Memory contents during simulation training.

	Working Memory Contents			
Trial #	1	2	3	4
Chunk 1	bean bag A	bean bag B	*wave*	*handshake*
Chunk 2	*reach*	bean bag A	bean bag B	bean bag A
Random	NA	*handshake*	NA	NA
Reward	203.4	-20.5	-197.7	2.3

Table 2. Working memory contents during simulation training.

In these trials, WMS was allowed to choose two "chunks" from the short- and long-term memory systems to accomplish the task. However, the working memory was not restricted to choosing exactly one object and one behavior. If the working memory chose to focus on two objects or two behaviors, then respectively a behavior or object was chosen at random. This ensured that an action was still performed. The reasoning behind this was so that the system did not learn to simply choose combinations that lead to no reward, a situation that could be preferred if WMS was consistently getting negative reward for its choices. Table 2 shows samples of the contents in the working memory in these trials.

To evaluate system performance further, a third task was developed. For this task ISAC was again given the command to "*reach* to the red bag", however this time the *reach* behavior was deleted from the initial knowledge limiting the behavior choices to *handshake* and *wave*. The working memory had to choose the *next best* behavior. For each of the arrangements shown previously (Figures 22a,.b), WMS chose to perform the *handshake* behavior. This behavior was chosen because it allowed the arm to get closest (Reward Criterion 3) to the bean bag (Reward Criterion 2) and thus, best accomplished the goal task.

7.1.2 Trials on ISAC

After the initial training and experimentation, ISAC was allowed to perform the generated motions (Figure 22.c). Two new objects (a green Lego toy, and a purple Barney doll) were added to the table, at random positions. ISAC's vision system was trained (Step 1) to recognize each new object and recorded the type of object as well as some simple descriptive information (color=green,

purple; toy type=Barney doll, Lego). ISAC was given tasks (Step 3) such as "*reach* to the bean bag" or "*reach* to the toy". Each of these tasks did not specify to which bean bag or toy ISAC was to *reach*. ISAC recognized the objects (Step 4). WMS focused on "chunks" of information from the SES and LTM in order to accomplish the task (Step 5). ISAC was allowed to explore the space of possible actions receiving reward each time (Steps 6 and 7). After this was accomplished, the objects were rearranged in a variety of different positions (Step 8) and ISAC was given a command. The results (set of 20 commands) were that ISAC successfully performed the correct action on the nearest (easiest to reach) requested object.

For this system to properly choose the correct set of chunks to focus on, the system currently has to explore all the possibilities during training. Figure 23, shows an example learning curve for this system for the *reach* command. The graph shows the number of times each of the trained behaviors (see Figure 23) was chosen during each ten trial segment. When the system first begins training, it is required to explore each of the possible behaviors as well as try different percept/behavior combinations. As can be seen from this graph, it took approximately 20 trials to learn *reach* before the system determined that the *reach* behavior was the definite best.

Attempting to explore all possibilities in the future will lead to a combinatorial explosion if a large number of behaviors or percepts are added to the system. In order for this system to continue to operate properly in the future, improvements need to be made to the representational structures for behaviors and percepts used by the system. One method of improving this representational structure that we are considering is to store intentionality along with percepts (i.e. chairs are for *sitting*, tables are for *placing*, and bean bags are for *reaching* and *grabbing*). This, along with a method discussed in section 7.1.3 of pre-filtering chunks using Episodic Memory, will aid WMS to perform quick and accurate chunk selection and retrieval.
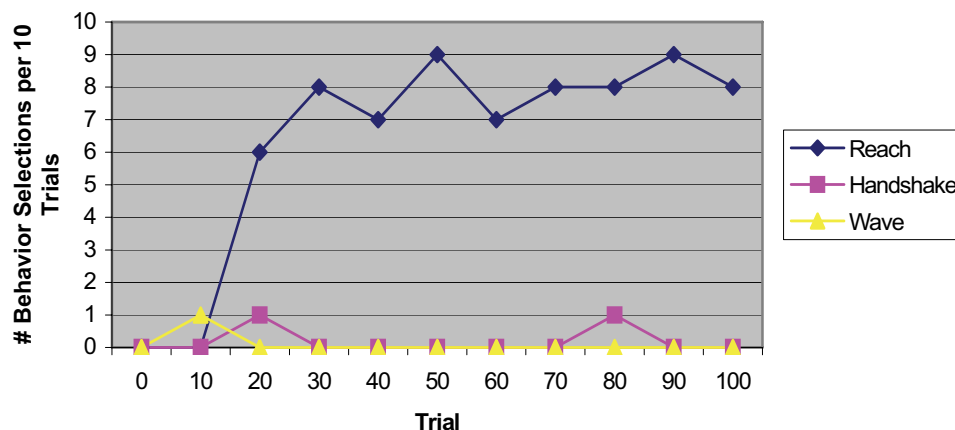
Figure 23. Learning Curve for *Reaching* Action.

7.1.3 Learning New Tasks Using Multiple WMS

A single WMS, if it were large enough and if it were trained extensively enough, could theoretically handle most, if not all, of the simple situations ISAC could encounter. However, due to the size of the state and chunk representations the computation time to select appropriate chunks and the training time to train a single WMS over all possibilities would be enormous. For this reason, separate WMS are being trained to handle different types of situations that ISAC may encounter. As stated earlier in this section, two differently WMS are currently in development: Object Interaction working memory (WM1) and Human Interaction (WM2).

When training WM1, the "Task Info" is set to the current command, such as "*reach* to the bean bag". When training WM2, however, the "Task Info" is kept blank. WMS in each case is responsible for learning which behavior chunks from LTM and which percept chunks from STM are appropriate for each situation. WMS is also responsible for learning "how well" certain chunks accomplish particular tasks. It is important that WMS learn which memory chunks best accomplish tasks and which other chunks could be used when, for some reason, the "best" ones are not available.

Using multiple WMS to accomplish the task of one monolithic WMS speeds up training time and decreases computation time. The idea behind training these separate systems is to enable ISAC the ability to continuously, smoothly, and appropriately interact with its environment. Each of these WMS, once trained, will be stored in the LTM and linked with the particular episode (see Episodic Memory, section 5.2 and 5.3).

Upon entering a new state, ISAC will pull from the Episodic Memory an episode that most closely matches the current state. Along with this episodic information will be the trained WMS that enabled ISAC to act appropriately in that situation. This WMS will be loaded into the system and used throughout the duration of the current state.

Episodic information also helps filter the list of candidate chunks presented to WMS. Figure 24 shows how Episodic Memory can be used to filter the candidate chunks list.

Pre-filtering the candidate chunks list also speeds up computation and selection time for WMS. This feature is especially important as ISAC's knowledge base grows. When no appropriately matching episode can be retrieved, ISAC can rely on the current state information (such as the presence of a task command, people to interact with, etc.) to determine which trained WMS is likely the most appropriate.

No appropriate feature is in place to filter the candidate chunk list for ISAC for this scenario.

$$\sum (LTM + STM)$$

Chunk List1

Beh. 1
Beh. 2
.
.
.
Beh. N
Percept 1
.
.
.
Percept M

Beh. x
Beh. y
Beh. z
Percept l
Percept m

WM1

Chunk List 2

Beh. t
Beh. u
Percept a
Percept b
Percept c

WM2

Episodic Memory
influence on chunk
list creation
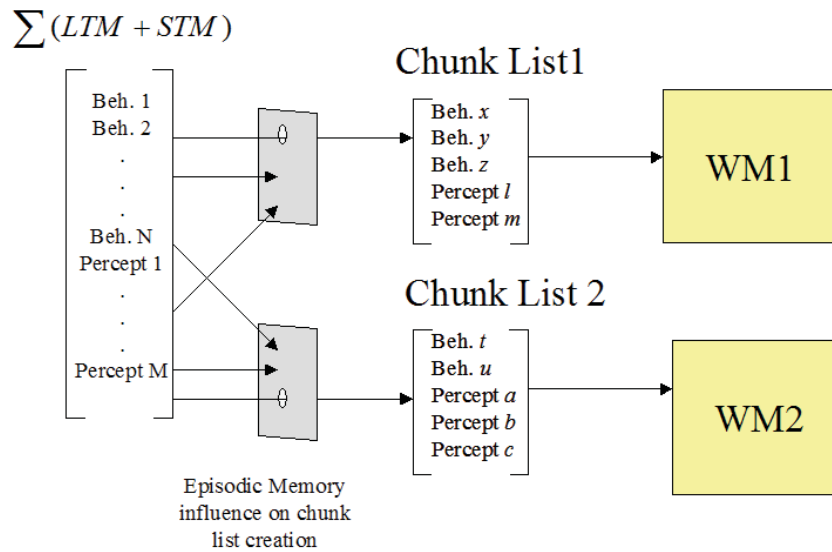
Figure 24. Pre-selection of candidate chunks based on past experience.

ISAC will be responsible for pouring through all stored, memorable information to complete the task. As ISAC's experience grows, however, the chunk list will begin to shrink as ISAC learns what types of information are most relevant. Once ISAC learns the information, an episode can be created and used to filter chunks for future scenarios.

7.2 Situation-based Stimuli Response Experiment

In order to test ISAC's decision-making functions under conflicting goals, a simulation experiment was conducted [Ratanaswasd, et. al., 2006]. In this experiment, ISAC first selects a set of percepts to pay attention to based on the emotional salience. ISAC then decides how to respond to each percept according to a situational change.

7.2.1 Experiment setup

System components use are Central Executive Agent, Attention Network, and Emotion Agent. Sound stimuli (i.e., music and alarm) are captured through a microphone and processed in Matlab. Mozart's Symphony No. 40 is used for "music," and a recorded sound of an actual fire alarm is used for "alarm." The initial state of ISAC's emotional level is to *dislike the alarm sound while liking the music*. This is accomplished through the emotion vectors shown in Table 3. ISAC is also trained to perform three actions, i.e., performing the requested task to fixate on the Barney doll, yelling "Alarm!", and performing a free-style dance.

Two types of situations were tested as shown in Figure 25:

Situation 1: (Salience-based Reactive Action Experiment))
Various sounds (a short piece of music, an alarm sound, human voices, and background noises) were presented to the system at different times while ISAC was idle, i.e. no task was conducted.

Situation 2: (Situation-Based Task Switching Experiment)
A task to fixate on the Barney doll was first given to the model. Then, the same sounds were presented during the task execution.
A feedback on the action selected was given by a human teacher as a part of supervisory learning..
The process was repeated until the model learned the proper response.
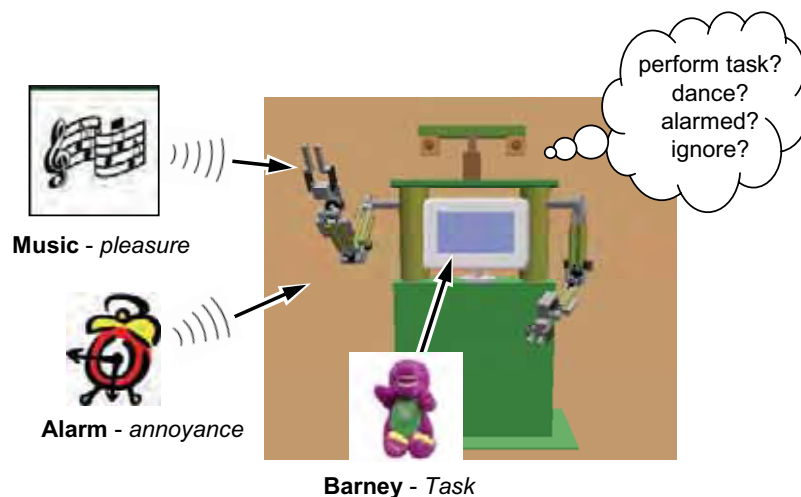
Figure 25. Overview of the experiment

7.2.2 Attention and Emotion

In our cognitive architecture, emotions are handled by the Emotion Agent (EA) [Dodd, 2005]. EA provides the emotional responses to the percepts in the environment. This response is currently represented in a pre-defined form of a vector called the *Emotion Vector*. Each element of the vector holds the level of a basis emotion that ISAC possesses toward the percept. Table 3 shows the Emotion Vector used in the experiment. The magnitude of this vector is sent to the Attention Network as the level of emotional salience for the given stimulus. The Attention Network then acts as a gating by allowing only the percepts with high emotional salience to go through and become candidate chunks for WM as shown in Figure 26.

Stimulus/Task-response	Emotion Vector			Emotional salience
	happy-sad	love-hate	alert-uninterested	
Alarm sound	0.1	-0.8	-0.6	1.01
Musical piece	0.7	0.5	0.1	0.86
Task command	0.1	0	0.6	0.61
Other human words	0.1	0.1	0.1	0.17

Table 3. Stimuli *vs.* Corresponding emotion vectors.
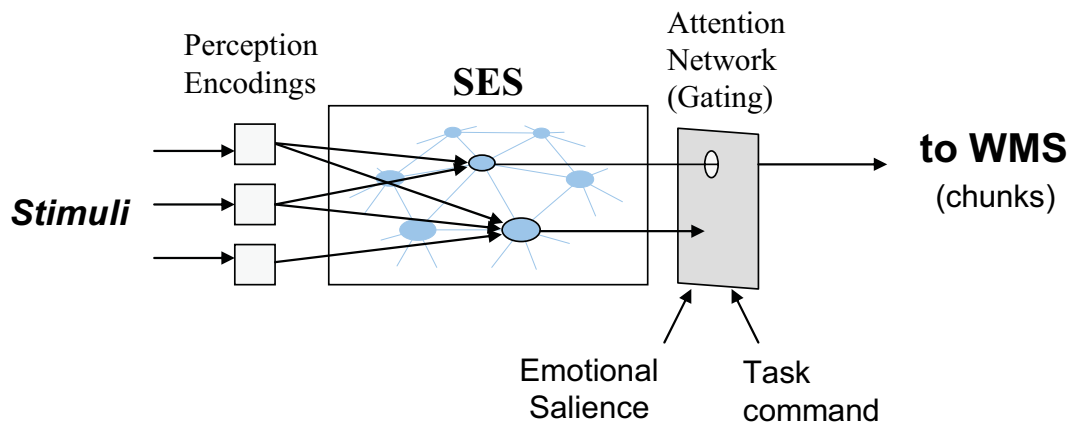
Figure 26. Focus of attention using the Attention Network.

7.2.3 Situation-Based Decision Making

If two or more percepts and/or commands are given to ISAC at the same time, ISAC must resolve the conflict. The Central Executive Agent (CEA) described in Section 6.2 is responsible for conflict resolution. For example, if a percept with a high emotional salience is detected while a task is being executed, CEA must make a decision how to respond to the newly acquired percept. The current situation is used by CEA for decision making. For this experiment, "a situation" can be translated from the change in perceptual information as follows: Let the set of all percepts in the Focus of Attention (FOA) at a given time be denoted by X. Members of X then comprise a combination of some known percepts from LTM. In a perceptual event, either a percept disappears or a new percept attracts the robot's attention, and the original set of percepts in FOA will change. For this experiment, "a situation" was considered to be *any change of specific members of X* as illustrated in Figure 24.
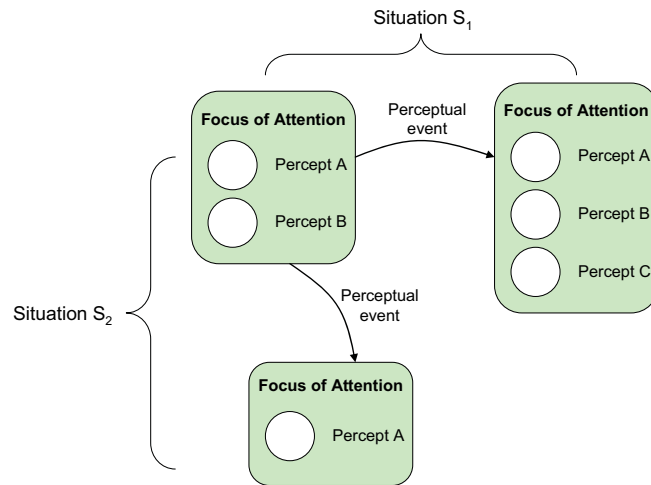
Figure 24. Relationship between changes in percepts in FOA and situations.

The probabilities $P[A_j^{(i)}]$, $j=1,2,\ldots,N$, associated with the actions $A_1^{(i)}$, $A_2^{(i)}$ ..., $A_N^{(i)}$ and subjected to the constraint $\sum_{i=1}^{N} P[A_j^{(i)}]=1$, were computed by CEA using past history of the number of times the appropriate action was provided through supervised learning. That is, during the teaching phase, the human teacher provided the appropriate action $A_j^{(i)}$. CEA then kept track of the frequency that $A_j^{(i)}$ had been provided for S_i, and used it to update $P[A_j^{(i)}]$ accordingly. During the execution phase, when Situation S_i occurred, CEA selected an action as follows:

The unit interval [0,1] is partitioned into N regions, each with a width of $P[A_j^{(i)}]$. A uniform random number on the unit interval is generated, and the region j, $1 \le j \le N$, in which it falls is determined. The associated action $A_j^{(i)}$ is then selected.
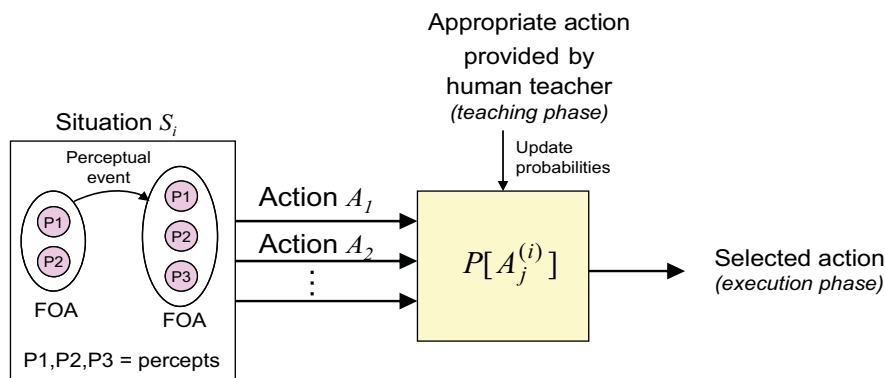
Figure 25. Decision making and learning.

By selecting an action probabilistically, the actions having higher probabilities are more likely to be selected. This enables continual exploration so that the

robot may respond to dynamic situations. The decision-making process is illustrated in Figure 25.

7.2.4 System Evaluation

In the first situation, only the musical piece and alarm sound caused the Emotion Agent to create the emotion vectors with a high emotional salience. Because no task (goal) was given, CEA selected the action based on the initial emotional state. This situation demonstrated the system's ability to focus attention to those percepts that cause a high emotional salience

In the second situation, a task to fixate on the Barney doll was given to the system prior to the presence of other stimuli. The changes in FOA then created two situations, i.e. "Music was heard during the task execution" and "Alarm was heard during the task execution". Using the probabilistic model of the situation as discussed above, CEA decided if it should pay attention to the stimuli or keep focusing on the task based on prior knowledge of the stimuli and situation. "Situation 2 (before learning)" in Table 4 summarizes the system responses.

FOA	Situation 1	Situation 2 (before learning)	Situation 2 (after learning)
Music	"Dancing"	Ignored the music	Ignored the music
Alarm	Yelled "Alarm!"	Yelled "Alarm!"	Ignored the alarm

Table 4. Stimuli Response Results.

Finally, the model was later taught to respond to Situation 2 differently from the initial knowledge. That is, the model entered the teaching phase again to learn a new appropriate response, which in this case was to ignore the alarm for Situation 2. 100 trials of teaching were performed and the results from learning are shown in Figure 26. This figure shows the number of times the model chose to ignore the alarm for every ten trials. In the beginning, the model did not ignore the alarm right away because of the strong association between the percepts and actions initially embedded in the model. After about 20 trials, the supervised learning changed the associated probabilities in the model enough so the model started to learn to ignore the alarm. With increasing trials, the system learned to select the correct response. However, as the selection was performed using a probabilistic method, it was still possible that the system selected incorrect action occasionally as seen in the graph. This allows the system to explore other possible actions in dynamic situations. Because the probabilistic model was updated for every teaching trial, the system was more likely to select the correct action as the number of trials increased. If

this number reached infinity, the system would then select the correct action 100% of the time.
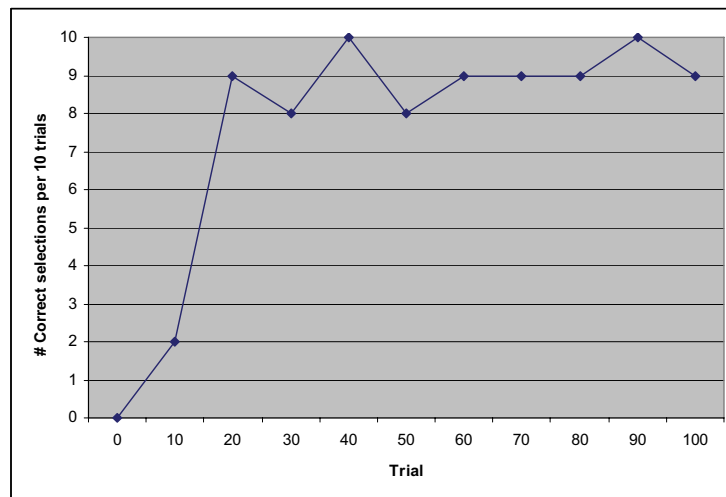
Figure 26. Learning curve for the response to the alarm in Situation 2.

This simple experiment was conducted to verify that the system did learn to select the appropriate action under supervisory learning [Mitchell, 1997] using attention and a set of "snapshot" state of emotions. As the next step, we are now working to develop a more realistic, dynamic model of emotion which will reflect the change in ISAC's internal states over time. The details of how this time-varying event-based model of emotion will influence action-selection process will be described in Section 8.

8. Future Integrated Experiment

Any cognitive robot should be able to use both external and internal stimuli to consciously organize their behaviors such as action selection, attention and learning. According to this, emotion could be one of main factors to mediate decision-making process. In order to make the attention- and emotion-based action selection process more realistic, we are now working to develop a time-varying event-based model of emotion reflecting the change in ISAC's internal states over time. In this type of action-selection process, the system does not necessarily perform the same action every time for the same set of external stimuli.

In ISAC's cognitive architecture, the Self Agent is responsible for meta-management of its internal states similar to that proposed by Sloman [Sloman, et al., 2005] as shown in Figure 30. We have used the fixed, embedded emotion level as a part of the Self Agent in the experiment. The Emotion Agent will

be modified to be more dynamic to better keep track of ISAC's internal state. The details of this work are described now.
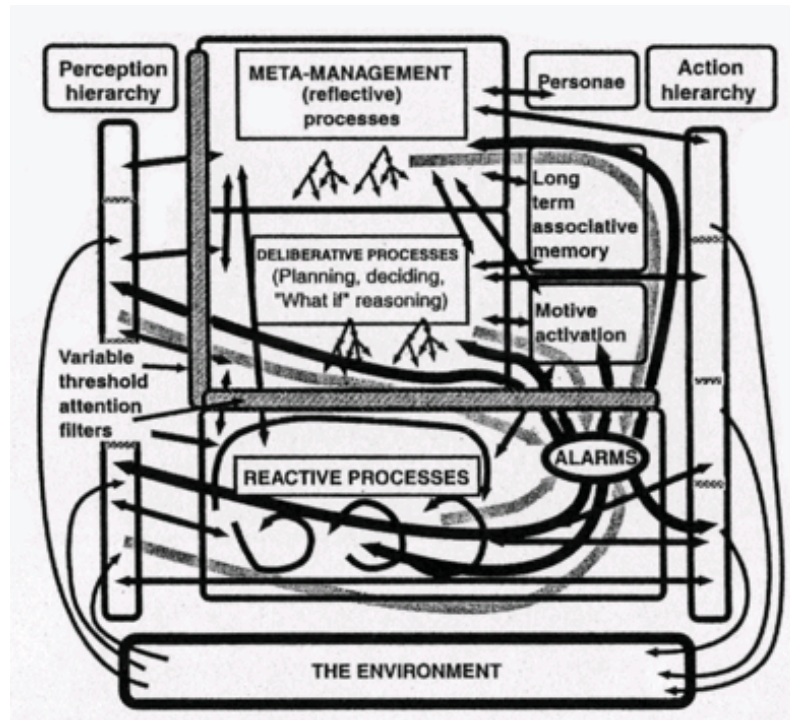
Figure 30. H-CogAff Architecture [Sloman, et al, p. 227, 2005]

8.1 System Integration

The incorporation of both internal and external stimuli in the architecture enables the system to be as dynamic as possible, gearing responses so that they are not a function of the external inputs alone. This creates a robot that can respond differently to the same situation based solely on the internal state of the robot. The internal stimulus that will be used for this experiment is the level of excitement of the robot. The excitement level will be a product of both ISAC's external environment and ISAC's other internal states (such as presence of command, joint activity, etc.)

It is important that ISAC's excitement or arousal to a given situation not be a static function, but rather a dynamic function of time. For the time being, ISAC's level of excitement is calculated using a first-order exponential decay function:

$$Excitement \ = \alpha(S) \cdot e^{-\beta(S) \cdot t}$$

The terms α(S) and β(S) are functions of the state, S, of ISAC and are designed in such a way that they can be learned or modified over time using standard reinforcement learning techniques. Therefore, a particular situation (or a change in state) S_i which may initially be embedded in ISAC as "very exciting" (i.e. α(S_i) returns a high value and β(S_i) returns a low value) can, over time, adjust to reflect ISAC's experience with that particular state. Conversely, states initially embedded as "not exciting" can, based on experience, become exciting states. One final point to add is that the decay nature of the excitement function ensures that no state continues to excite ISAC indefinitely (i.e. ISAC will eventually get bored with even the most exciting event).

As ISAC's other cognitive processes learn, these processes in turn will utilize the current state of excitement when making decisions. This utilization will be a function of the excitement level as well as the internal and external states that have caused the current excitement level. As the stimuli that excite ISAC change over time, ISAC's decision-making process should reflect this change and summarily, ISAC should make different choices. The experiment is designed to teach ISAC this ability and then put ISAC in a situation in which multiple possibilities exist forcing the robot to make a decision. It is hoped that ISAC's cognitive architecture will allow it to make this decision.

8.2 Experimental Design

To demonstrate the use and effectiveness of utilizing both internal and external stimuli during action selection and task switching, an experiment has been designed that requires the principles of cognitive robotics discussed in this chapter. During this experiment, ISAC will be presented with a range of different scenarios and be forced to decide whether to continue with the present task or switch to another task. Close monitoring of ISAC's internal level of excitement or arousal will be the mechanism that that aids in making this decision.

Through habituation and learning, ISAC will develop an association between excitement levels and different percepts or tasks. In other words, based on experience, certain percepts will excite ISAC more than other percepts, and certain tasks will excite ISAC more than other tasks. These associations will begin as embedded knowledge, based on novelty, within ISAC. Over time and through experience and habituation, these correlations will change and ISAC will begin to develop its own sense of excitement/boredom.

he experiment steps are as follows:

1. Embed ISAC with knowledge that certain percepts and tasks are more exciting than others (i.e. faces are more exciting than bean bags, *dancing* is more exciting than *reaching*, etc.)

2. Train a small set of WMS to react to certain situations (see WM1 and WM2 from section 7.1)

   a) WM1 is trained to enable ISAC to interact with simple objects.
   b) WM2 is trained for interaction with people.
   c) WM3 is trained to enable ISAC to respond appropriately to sound stimuli.

3. Have a person enter the room and give ISAC a task.

4. Repeat step 3 several times in order to cause a change in ISAC's embedded excitement function (Section 8.1)

5. Have a person enter the room and give ISAC a task. During the task execution have music begin playing in the room.

6. Continue playing the music for several minutes.

Steps 1 and 2 of this experiment are the initial embedding of knowledge into the system. When a person enters the room and gives ISAC a command, this interaction should excite ISAC causing it to desire to engage with the person and complete the task. Through repetition of Step 3, this excitement level should continue to decrease with each repeated command. Over time, the excitement level associated with Step 3 should degrade to such an extent that ISAC essentially becomes unmotivated to perform the task. At this point, when ISAC hears music during the execution of the task (Step 5), the robot should choose to ignore the person and pay attention to the music instead. After the music plays for several minutes (Step 6), ISAC should eventually become bored with this as well (as discussed in section 8.1.). Once bored with the music, ISAC should transition back to the commanded task.

9. Conclusions

In the last forty years, industrial robots have progressed from the Plan-Sense-Act paradigm to more robust, adaptive/intelligent control paradigm [Kawamura, 2006]. In particular, the integration of body, sensor and AI-based software has produced not only advanced industrial robots, but non-industrial robots ranging from entertainment and home to a variety of health-related robots, we expect this trend to continue. This chapter introduced the next grand challenge in robotics, i.e. the integration of body and mind. In particular, the chapter described our efforts towards this challenge through the realization of a cognitive robot using cognitive control, attention, emotion, and an adaptive working memory system. In the last forty years, the field of industrial robotics and automation has also seen many innovations. As manufacturing becomes more distributed and sophisticated, realization of human-like robotic coworkers with cognitive skills will be a challenge not only to academia, but to manufacturing engineers as well.

10. References

ACT-R (2006). http://act-r.psy.cmu.edu/about/

Albus, J.S. (1991). Outline for a theory of intelligence, *IEEE Trans Systems, Man, and Cybernetics,* vol.21, no.3, pp. 473–509.

Anderson, J. & Lebiere, C. (1998). *The Atomic Components of Thought*, Lawrence Erlbaum Associates, Mahwah,, NJ.

Arbib, M.A. & Fellous, J-M. (2004). Emotions: From brain to robot, *Trends in Cognitive Science*s, vol. 8, no. 12, pp. 554-561, December 2004.

Arkin, R. (1998). *Behavior-Based Robotics*. MIT Press, Boston, MA.

Asada,M.(2002).*Japan'sSynergisticIntelligenceProject*, http://www.jeap.org/web/

Baddeley, A.D. (1990). *Human Memory: Theory and Practice,* Oxford University Press, Oxford, UK.

Baddeley, A.D. (1986). *Working Memory*, Clarendon Press, Oxford, UK.

Baddeley, A.D. & Hitch, G.J. (1974). *Working Memory*, In G.A. Bower (Ed.), *Recent Advances in Learning and Motivation*, Vol. 8, pp. 47-90, Academic Press, New York, NY.

Braver, T.S. & Barch, D.M. (2002). A theory of cognitive control, aging cognition, and neuromodulation, *Neuroscience and Behavioral Reviews,* vol. 26, pp. 809-817, Elsevier Science Ltd., St. Louis, MO.

Braver, T.S. & Cohen, J.D. (2000). On the control of control: The role of dopamine in regulating prefrontal function and working memory, In S. Monsell and J. Driver, eds., *Control of Cognitive Processes: Attention and Performance XVIII,* pp. 713-738, MIT Press, Cambridge, MA.

Brooks, R.A.(1991). Intelligence without representation, *Artificial Intelligence,* vol.47, no.1-3, pp.139-160.

Christensen, J.H. (1996). Rockwell Automation Advanced Technology. A Handout, *Presentation of Holonic Manufacturing Systems (HMS): Partnering Opportunities,* Harrisburg, PA, February 6, 1996.

DARPA IPTO. (2005). Biologically-Inspired Cognitive Architecture (BICA) Program; http://www.darpa.mil/ipto/programs/bica/index.htm

Dodd, W. & Gutierrez, R. (2005). Episodic Memory and Emotion for Cognitive Robot**,** *14th IEEE Int'l Workshop on Robot and Human Interactive Communication (RO-MAN2005)*, Nashville, TN, Aug 13-15, 2005.

Erol, D.; Park, J., Turkay, E., Kawamura, K., Jenkins, O.C. & Mataric, M.J. (2003). Motion Generation for Humanoid Robots with Automatically Derived Behaviors, *Proc. of IEEE Int'l. Conf. on Systems, Man, and Cybernetics,* Washington, DC, October 2003.

EU UnitE5 Cotion.(2004).http://cordis.europa.eu/ist/cognition/index.html

Ferber, J. (1999). *Multi-Agent Systems: An introduction to distributed artificial intelligence*, Addison-Wesley Professional.

Funahashi, S. & Kubota, K. (1994). Working memory and prefrontal cortex, *Neuroscience Research,* vol. 21, pp. 1-11.

Gat, E. (1998). *Three Level Architectures,* Chapter 8, in *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems,*(E. Kortenkamp, R.P. Barasso, & R. Murphy, eds.), AAAI Press, pp. 195-210.

Gordon, S. & Hall, J. (2006). System Integration with Working Memory Management for Robotic Behavior Learning, *ICDL*, Bloomington, IN.

Greeno, J.G. (1984). Gibson's Affordances, *Psychological Review*, Vol. 101, No. 2, pp.336-342.

Hambuchen, K.A. (2004). *Multi-Modal Attention and Binding using a Sensory EgoSphere*, Ph.D. Dissertation, Vanderbilt University, Nashville, TN, May 2004.

Holland, O., ed. (2003). *Machine Consciousness*, Imprint Academic, Charlottesville, VA.

Hollnagel, E. & Woods, D.D., (1999). Cognitive Systems Engineering, *Int'l Jo. of Human-Computer Studies*, Vol. 18, pp. 339-356.

IMS. (1996). http://ksi.cpsc.ucalgary.ca/IMS/IMS.html

Jenkins, O.C. & Matarić, M.J. (2003). Automated derivation of behavior vocabularies for autonomous humanoid motion," *2nd Int'l Joint Conf. on Autonomous Agents and Multiagent Systems*, **LOCATION.**

Kawamura, K.; Rogers, T.E., Hambuchen, K.A., & Erol, D. (2003a). Towards a Human-Robot Symbiotic System, pp.555 – 565, *Robotics and Computer Integrated Manufacturing*, vol. 19.

Kawamura, K.; Noelle, D.C., Hambuchen, K.A. & Rogers, T.E. (2003b). A Multi-agent Approach to Self-reflection for Cognitive Robots", *Proc. of 11th Int'l Conf. on Advanced Robotics*, pp.568-575, Coimbra, Portugal, Jun 30-Jul 3, 2003.

Kawamura, K.; Rogers, T., & Ao, X. (2002). Development of a cognitive model of humans in a multi-agent framework for human-robot interaction," *1st Int'l Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS),* pp. 1379-1386, Bologna, Italy, July 25-27, 2002.

Kawamura, K.; Peters II, R.A., Wilkes, D.M., Alford, W.A., & Rogers, T.E. (2000). ISAC: Foundations in human-humanoid interaction," *IEEE Intelligent Systems,* July/August, pp. 38-45.

Kawamura, K.; Bagchi, S., Iskarous, M., Bishay, M. & Peters II, R.A. (1995). Intelligent Robotic Systems in Service of the Disabled," *IEEE Trans. on Rehabilitation Engineering*, Vol. 3, No.1, pp.14-21.

Kawamura, K.; Pack, R.T., Bishay, M., & Iskarous, M. (1986). Design Philosophy for Service Robots, *Robotics and Autonomous Systems, Int'l Workshop on Biorobotics: Human-Robot Symbiosis,* (eds. K. Kawamura and T.E. Davis), Elsevier, pp.109-116.

Kieras, D.E.; Mueller, M.S., & Seymour, T. (1999). Insights into Working Memory from the Perspective of the EPIC Architecture for Modeling Skilled Perceptual-Motor and Cognitive Human Performance, in Models of Working Memory. Mechanisms of Active Maintenance and Executive

Control, A. Miyake and P. Shah, Eds., Cambridge University Press, Cambridge.

Kieras, D. & Meyer, D. (1995). An Overview of the EPIC Architecture for Cognition and Performance with Application to Human-Computer Interaction, *EPIC Report No. 5 (TR-95/ONR-EPIC-5)*, University of Michigan.

Koestler, A. (1967). *The Ghost on the Machines,* Arkana Books, London.

Laird, J.E.; Newell, A. & Rosenblum, P.S. (1987). SOAR: An architecture for general intelligence, *Artificial Intelligence*, Vol. 33, pp. 1-64.

Lovett, M.C.; Reder, L.M., & Lebiere, C. (1999). Modeling Working Memory in a Unified Architecture. An ACT-R Perspective, in Models of Working Memory. Mechanisms of Active Maintenance and Executive Control, (A. Miyake and P. Shah, Eds.), Cambridge University Press, Cambridge.

MacDonald, A.W.I.; Cohen, J., Stegner, V., & Carter, C.S. (2000). Dissociating the Role of Dorsolateral Prefrontal and Anterior Cingulated Cortex in Cognitive Control," *Science, Vol.* 288 (5472), pp.1935-1838.

MacLeod, C.M. & Sheehan, P.W. (2003). Hypnotic Control of Attention in the Stroop Task: A historic footnote, *Consciousness and Cognition*, vol. 12, pp. 347-353.

Meyer, D.E. & Kieras, D.E. (1997) A computational theory of executive cognitive processes and multiple-task performance: I. Basic mechanisms. *Psychological Review*, Vol. 10, No. 41, pp. 3-65.

Miller, E.K. (2003). *Cognitive Control: Understanding the brain's executive,* in *Fundamentals of the Brain and Mind*, Lecture 8, June 11-13, 2003, MIT Press, Cambridge, MA.

Miller, E.K.; Erickson, C.A. & Desimone, R. (1996). Neural Mechanisms of Visual Working Memory in Prefrontal Cortex of the Macaque," *Jo. of Neuroscience,* vol.16, pp.5154-6.

Minsky, M. (1985). *The Society of Mind,* Simon & Schuster, New York, NY, pp. 39.

Mitchell, Tom M. (1997), *Machine Learning*, McGraw-Hill, New York, NY.

Olivares, R.E. (2004). *The Intelligent Machine Architecture 2.5: A revised development environment and software architecture,* Master Thesis, Vanderbilt University, Nashville, TN, May 2004.

Olivares, R.E. (2003). *Intelligent Machine Architecture 1.0: Introduction and system overview,* CIS / Cognitive Robotics Laboratory Technical Report, Vanderbilt University, Nashville, TN, May 2003.

O'Reilly, R.; Braver, T. & Cohen, J. (1999). *A Biologically Based Computational Model of Working Memory,* In *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control*, (A. Miyake and P. Shah, Eds.), Cambridge University Press, Cambridge, UK.

Pack, R.T. (1998). *IMA: The Intelligent Machine Architecture,* PhD Dissertation, Vanderbilt University, Nashville, TN, May 1998.

Peters II, R.A.; Hambuchen, K.A., Kawamura, K. & Wilkes, D.M. (2001). The

sensory egosphere as a short-term memory for humanoids," *Proc. of the IEEE-RAS Int'l Conf. on Humanoid Robots*, Waseda University, Tokyo, Nov. 22-24, 2001, pp 451-459.

Phillips, J. & Noelle, D. (2006). Working Memory for Robots: Inspirations from Computational Neuroscience, *Proc. from 5th Int'l Conf on Development and Learning,* Bloomington, IN, June 2006.

Phillips, J. & Noelle, D. (2005). A Biologically Inspired Working Memory Framework for Robots, *14th IEEE Int'l Workshop on Robot and Human Interactive Communication*, Nashville, TN, Aug 13-15, 2005, pp 599-601.

Ratanaswasd, P., Garber, C. & Lauf, A. (2006). Situation-Based Stimuli Response in a Humanoid Robot, *Proc. from 5th Int'l Conf on Development and Learning,*, Bloomington, IN, June 2006.

Rogers, T.E. (2003). *The Human Agent: A model for human-robot interaction*, PhD Dissertation, Vanderbilt University, Nashville, TN, August 2003.

Rose, C.; Cohen, M.F. & Bodenheimer, B. (1998). Verbs and Adverbs: Multidimensional motion interpolation," *IEEE Computer Graphics and Applications,* vol. 18, no. 5, September– October, pp. 32-40.

Saad, A. (1996). *Agent-Based Dynamic Decentralized Scheduling for Flexible Manufacturing Systems*, Ph.D. Dissertation, Vanderbilt University, Nashville, TN, August 1996.

Shu, S., Wilkes, M. & Kawamura, K. (2000). Development of Reusable, Configurable, Extensible Holonic Manufacturing System, *Proceedings of IEEE Systems, Man & Cybernetics (SMC 2000),* Washington, D.C., Vol.3, pp. 1679-1684.

Sloman, A., Chrisley, R.I. & Scheutz, M.(2005). The Architectural Basis of Affective States & Processes. In M. Arbib and J. M. Fellous, eds, *Who Needs Emotions?: The Brain Meets the Machine.* Oxford Univ. Press, Oxford.

SOAR BICA Research Project. (2006). http://sitemaker.umich.edu/soar/soar_bica_research-project

Spratley II, A.W., *Verbs and Adverbs as the Basis for Motion Generation in Humanoid Robots,* M.S. Thesis, Vanderbilt University, Nashville, TN, August 2006.

Stuss, D.T. & Knight, R.T. (2002). *Principles of Frontal Lobe Function*, Oxford University Press, Oxford.

Sutton, R.S. (1988). Learning to predict by the method of temporal differences, *Machine Learning,* vol.3, pp. 9-44.

Taylor, J.G. & Fragopanagos, N. (2004). Modeling the Interaction of Attention and Emotion, *Brain-Inspired Cognitive Systems,* University of Stirling, UK.

Tenenbaum, J.B.; de Silva, V. & Langford, J.C. (2000). A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290 (5500), pp. 2319–2323.

van Leeuwen, E.H. & Norrie, D. (1998). Holonic Manufacturing Systems, Handout, HMS8 Plenary Meeting, Vancouver, BC, February 11-17, 1998.

Automatic Modeling for Modular Reconfigurable Robotic Systems – Theory and Practice

I-Ming Chen, Guilin Yang and Song Huat Yeo

1. Introduction

A modular reconfigurable robot consists of a collection of individual link and joint components that can be assembled into a number of different robot geometries. Compared to a conventional industrial robot with fixed geometry, such a system can provide flexibility to the user to cope with a wide spectrum of tasks through proper selection and reconfiguration of a large inventory of functional components. Several prototyping systems have been demonstrated in various research institutions (Cohen et al. 1992; Fukuda & Nakagawa 1988; Schmitz, et al. 1988; Wurst 1986). Applications of modular systems have been proposed in rapid deployable robot systems for hazardous material handling (Paredis et al. 1995), in space stationed autonomous systems (Ambrose 1995), and in manufacturing systems (Chen 2000; 2001).

In the control and simulation of a modular reconfigurable robot system, precise kinematic and dynamic models of the robot are necessary. However, classical kinematic and dynamic modelling techniques for robot manipulators are meant for robot with fixed geometry. These models have to be derived manually and individually stored in the robot controller prior to simulating and controlling the robot. Commercial robot simulation software usually provides end users with a library of predefined models of existing robots. The models of any new robot not in the library have to be derived exclusively from the given parameters and commands in the package. For a modular robot system built upon standard modular components, the possible robot geometries and degrees of freedom become huge. As shown by Chen (1994), the number of robot-assembly configurations grows exponentially when the module set becomes large and the module design becomes complicated. To derive all of these models and store them as library functions require not only tremendous effort but also very large amount of disk storage space. In such cases, it is impractical and almost impossible to obtain the kinematic and dynamic models of a robot based on the fixed-geometry approach. Hence, there is a need to develop an automatic model-generation technique for modular robot applications.

In this chapter, we introduce a framework to facilitate the model-generation procedure for the control and simulation of modular robots. The framework consists of three parts: a component database; a representation of modular robot geometry; and geometry-independent modelling techniques for kinematics, dynamics, and calibration. The component database maintains the description and specifications of standard robot components, such as actuators, rigid links, sensors, and end effectors. The robot representation indicates the types and orders of the robot components being connected. The geometry-independent modelling algorithms then generate the proper models based on the robot description.

A graph based technique, termed the *kinematic graph*, is introduced to represent the module-assembly sequence and robot geometry. In this graph, a node represents a connected joint module and an edge represents a connected link module. Modules attached to or detached from the robot can be indicated by adding or removing nodes or edges from the graph. The realization of this graph is through an *Assembly Incidence Matrix* (AIM) (Chen 1994; Chen & Burdick 1998). A modular robot can be conceived according to the given AIM without knowing the other parameters, such as joint angles and initial positions. Here, we assume the generic structure of a modular robot is branch-type. The serial type modular robot is a special case of the branch-type structure.

Previous attempt to deal with automatic model generation for modular robots employed Denavit-Hartenburg (D-H) parameterization of the robot (Kelmar & Khosla 1988; Benhabib et al. 1989). However, the D-H method does not provide a clear distinction between the arranging sequence of the modules in the robot chain and their spatial relationship. Also, it depends on the initial position of the robot: the same robot may have different sets of D-H parameters just because of the different initial or zero positions. When evaluating the task performance of a modular robot with respect to its corresponding geometry, complicated equivalence relationships must be defined on the sets of parameters to identify the uniqueness of the robot geometry (Chen & Burdick 1998).

The formulation of the kinematics and dynamics is based on the theory of Lie groups and Lie algebras. The robot kinematics follows a local representation of the product-of-exponential (POE) formula, in which the joints, regardless of the types, are defined as members of *se(3)*, the Lie algebra of the Euclidean group *SE(3)*. The associated Lie algebraic structure can simplify the construction of the differentials of the forward-kinematic function required for numerical inverse solutions. The POE representation can also avoid the singularity conditions that frequently occur in the kinematic calibration formulated by the D-H method (Chen & Yang 1997; Chen et al. 2001). Thus, it provides us with a uniform and well-behaved method for handling the inverse kinematics of both calibrated and uncalibrated robot systems. Since the joint axes are described in the local module (body) coordinate systems, it is convenient for progressive

construction of the kinematic models of a modular robot, as it resembles the assembling action of the physical modular robot components. The formulation of the dynamic model is started with a recursive Newton-Euler algorithm (Hollerbach 1980; Rodriguez et al. 1991). The generalized velocity, acceleration, and forces are expressed in terms of linear operations on se(3) (Murray et al. 1994). Based on the relationship between the recursive formulation and the closed-form Lagrangian formulation for serial-robot dynamics discussed in (Featherstone 1987; Park et al. 1995), we use an *accessibility* matrix (Deo 1974) to assist in the construction of the closed-form equation of motion of a branch-type modular robot, which we assume is the generic topology of a modular robot. Note that all the proposed modelling techniques can contend with redundant and nonredundant modular robot configurations.

This chapter is organized as follows. Section 2 introduces the basic features of the hardware and software of a newly conceived modular robotic workcell. Section 3 briefly reviews the definitions of the AIM presentation and the associated accessibility matrix and path matrix. Section 4 concerns the formulation and implementation of geometry-independent kinematic, dynamic, and calibration models for modular robots. In addition to automated model generation, identification of the optimal modular robot assembly geometry for a specific task from the vast candidate database is also important. The AIM representation facilitates the search/optimization process by using the genetic algorithms approach. Section 5 investigates the task-oriented optimal geometry issues in modular reconfigurable robots and the advantage of using AIM to solve this type of problem. The proposed automatic model-generation method implemented in a Windows Based application for modular robotic automation system, termed *SEMORS* (Simulation Environment for MOdular Robot System) is introduced in Section 6. Prototypes of the modular robotic automation systems configured in both serial and parallel geometries for positioning and machining purposes based on the operation of *SEMORS* are illustrated in Section 7. This chapter is concluded in Section 8.

2. System Architecture

Figure 1 illustrates the system layout of a reconfigurable robotic workcell proposed by Nanyang Technological University and Singapore Institute of Manufacturing Technology (Chen 2001). The objective of this project is to develop a reconfigurable modular robotic workcell which is capable of performing a variety of tasks, such as part assembly, material transfer, and light machining (grinding, polishing and deburring), through rapid change of reusable workcell components. In this system, workcells are made of standard interchangeable modular components, such as actuators, rigid links, end-of-arm tooling, fixtures, and sensors. These components can be rapidly assembled and config-

ured to form robots with various structures and degrees of freedom. The robots, together with other peripheral devices, will form a complete robotic workcell to execute a specific manufacturing task or process. The corresponding intelligent control and simulation software components are then reconfigured according to the change of the workcell configuration. The maintenance and upgrade of the system are simplified by replacing the malfunctioned or outdated components. Converting a manufacturing line from one product to another can be very fast in order to keep up with the rapidly changing marketplace.
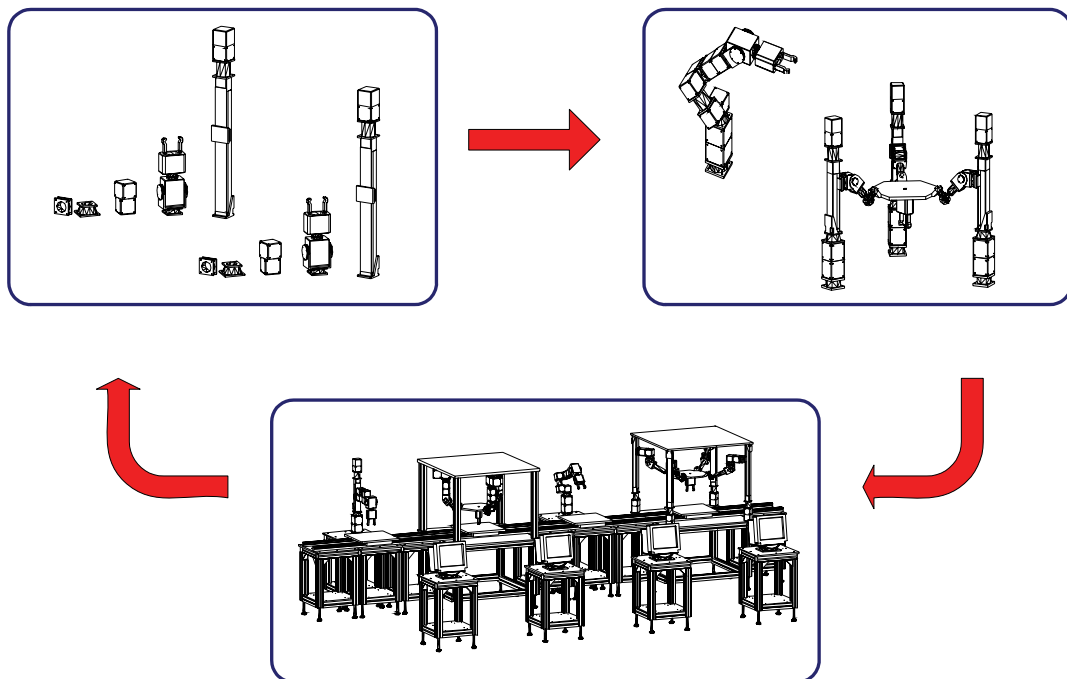
Figure 1. Deployment of a reconfigurable robotic workcell

In this system, the workcell software is designed in reusable- and reconfigurable-object fashion for ease of maintenance and development. Figure 2 illustrates the overall software architecture of the modular workcell. The user environment will provide all the necessary functions to facilitate the end user in controlling, monitoring and simulating the workcell. It consists of the following parts:

Component browser –- for viewing and editing the components available in the component database;

- *Simulator* --- for generating a computer-simulation model of a modular robot and the entire workcell; additionally, the simulator may be employed as the core function for future virtual manufacturing capabilities;

- *Task level planner* --- for determining the optimal geometry of a modular robot for a given task and the overall layout of the workcell for a particular manufacturing process;

- *Programming interface* --- for providing command and control of the system; and

- *Controller* --- for commanding the low-level individual controllers located in the components, and identifying the robot's geometry from the local component controllers.
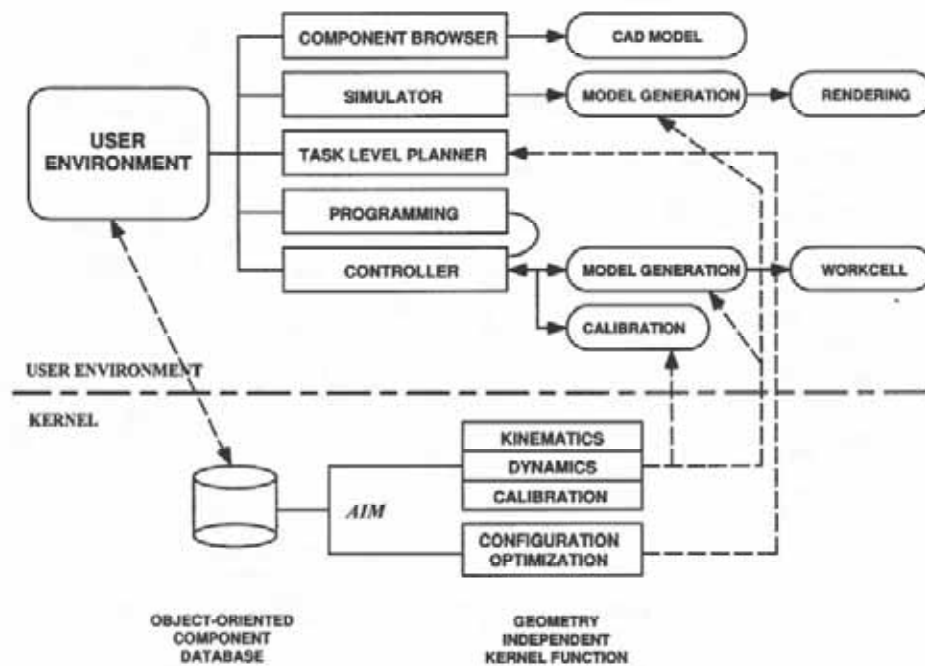
Figure 2. Software architecture for reconfigurable workcell

The system kernel, which is hidden from the user, provides automated model-generation functions and the configuration-optimization function (a component database is also associated with it):

- *Object-oriented component database*----manages the specification of all the components, such as the dimensions and weights of the links, maximum kinematic and dynamic performance of the actuators, etc. It can be accessed by the user for browsing and editing purposes.

- *Geometry-independent kernel functions*---generates kinematic and dynamic models of the robots shared by the simulators and the controller. Using identical models in the simulation and control of the workcell insures the reliability and integration of the system, and enables physically based simulations through the workcell controller. The configuration-optimization function can enumerate all possible robot geometry from an inventory of module components in the database, and select the most suitable one for a prescribed task. This information will pass back to the task-level planner to determine the optimal layout and locations of the robots in the workcell.

The information passing from the component database to the modeling functions is through the assembly incidence matrix. Robot geometries (serial, branch, or hybrid) and detailed connection information, such as the connecting orientation and the types of adjacent modules, are all indicated in the matrix. This matrix is then passed to the geometry-independent functions for model generation.

In such a system, the need to maintain a huge library of robot models is eliminated; instead, we maintain a small selection of the component-database and kernel functions for automated model generation, reducing the overall footprint of the system software.

3. Modular Robot Representation

3.1 Module Representation

To make the automatic model-generation algorithms work on a variety of module components, we introduce a conceptual set of modules whose features are extracted from those of real implementations. The modular systems developed to date have several common mechanical and structural features: (1) only 1-DOF revolute and 1-DOF prismatic joints; (2) symmetric link geometries for interchangeability; and (3) multiple connection ports on a link.
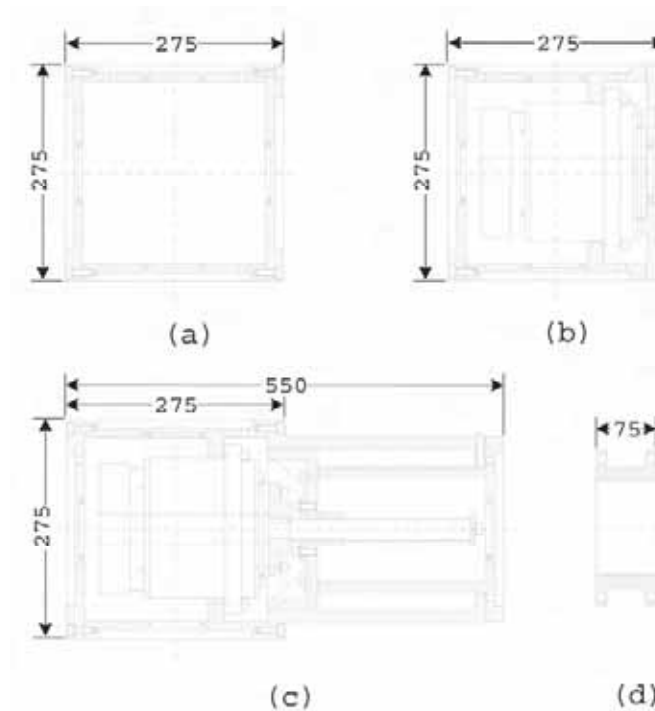
Figure 3. Modular robot components

3.1.1 Joint Modules

A modular robot joint module is an ``active'' joint, which allows the generation of a prescribed motion between connected links. Two types of joint modules, the revolute joints (rotary motion) and the prismatic joints (linear or translational motion), are considered. Rotary and linear actuators must reside in the modules to produce the required motions and maintain the modularity of the system. Multi-DOF motions can be synthesized with several 1-DOF joints. Joint modules are attached to link modules through standardized connecting interfaces for mechanical, power, and control connections.

3.1.2 Link Modules

The place on a link module where the joint is connected is called a connecting port. Without loss of generality, we assume that a link module is capable of multiple joint connections, and the link module has symmetrical geometry. Such a design allows modules to be attached in various orientations, and the robot geometry to be altered by simple reassembling. The modular robot components developed in our university are shown in Figure 3. This design follows the building-block principle whereby modules can be stacked together in various orientations through connecting points on all six faces of the cubes.

3.2 Assembly Incidence Matrix

Definition 1. (Graph)

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a vertex set, $\mathcal{V} = \{\mathbf{v}_0, \cdots, \mathbf{v}_n\}$, and an edge set, $\mathcal{E} = \{\mathbf{e}_0, \cdots, \mathbf{e}_m\}$, such that every edge in $\mathcal{E}$ is associated with a pair of vertices, i.e., $\mathbf{e}_i = (\mathbf{v}_j, \mathbf{v}_k)$.
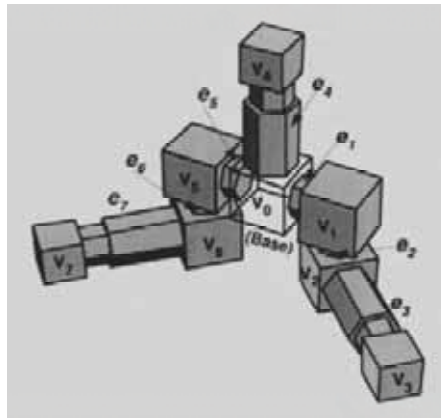
(a)
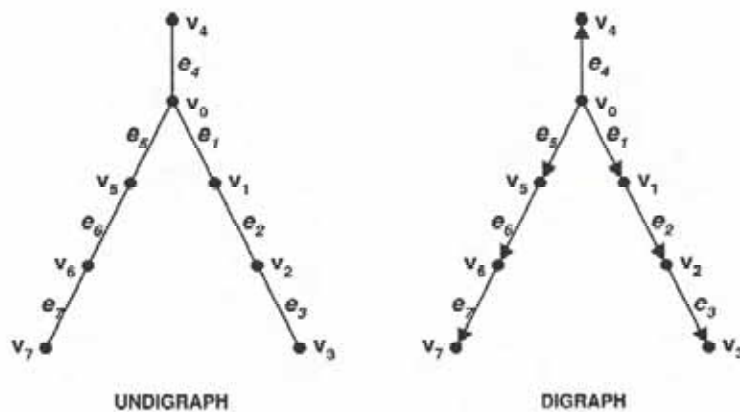
(b)

Figure 4. (a) A branching modular robot; (b) kinematic graphs of the robot

In mechanism design theory, a kinematic chain of links and joints is often represented by a graph, termed a *kinematic graph* (Dobrjanskyj & Freudenstein 1967), in which vertices represent the links and edges represent the joints. Using this graph representation, we are able to categorize the underlying structure (or geometry) of a linkage mechanism and apply the result from the graph theory to enumerate and classify linkage mechanisms. A robot

manipulator is also a kinematic chain, thus, admitting a kinematic graph representation. For example, an 8-module 7-DOF branch-type modular robot and its kinematic graphs are shown in Figure 4(a) and 4(b). It is also known that a graph can be represented numerically as a *vertex-edge incidence matrix* in which the entries contain only 0s and 1s (Deo 1974). Entry (i, j) is equal to 1 if edge $\mathbf{e}_j$ is incident on vertex $\mathbf{v}_i$, otherwise, it is equal to zero. This incidence relationship defines the connectivity of the link and joint modules. Because link modules may have multiple connecting points, we can assign labels to the connecting points to identify the module connection. To further identify those connections in the incidence matrix, we can replace those entries of 1 by the labels of the connected ports being identified on the link modules, and keep those entries of 0 unchanged. This modified matrix, termed an *assembly incidence matrix*, provides us the necessary connection information of the modules and also the basic geometry of the modular robot.

Definition 2. (Assembly incidence matrix)

Let $\mathcal{G}$ be a kinematic graph of a modular robot and $\mathcal{M}(\mathcal{G})$ be its incidence matrix. Let **port** *be the set of labels assigned to the connecting ports on the link modules. The assembly incidence matrix of the robot $\mathcal{A}(\mathcal{G})$ is formed by substituting the 1s in $\mathcal{M}(\mathcal{G})$ with labels in* **port** *on respective modules. One extra column and row are augmented to $\mathcal{A}(\mathcal{G})$ to show the types of link and joint modules.*

Note that the representation and assignment of the labels are nonunique. The labels of the connecting ports may be numerical values (Chen 1994) or may be derived from the module coordinates (Chen & Yang 1996). In this case, the module-component database should use consistent bookkeeping for this information. The AIM of the modular robot (8 link modules and 7 joint modules) shown in Fig. 4 is a 9×8 matrix:

$$\mathcal{A}(\mathcal{G}) = \begin{bmatrix} 1 & 3 & 5 & 0 & 0 & 0 & 0 & B \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & C1 \\ 0 & 1 & 0 & 6 & 0 & 0 & 0 & C1 \\ 0 & 0 & 2 & 0 & 6 & 0 & 0 & C1 \\ 0 & 0 & 0 & 5 & 0 & 2 & 0 & C2 \\ 0 & 0 & 0 & 0 & 5 & 0 & 3 & C2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & C2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & C2 \\ P & R & R & R & R & P & P & 0 \end{bmatrix}. \tag{1}$$

Note that there are three types of link modules in the robot: the base (B), the large cubic module ($C1$), and the small cubic module ($C2$). Cubic modules have six connecting interfaces labeled 1 – 6; i.e., **port** $= \{1, \cdots, 6\}$, which follows the labeling scheme on dice. The revolute joints and prismatic joints are denoted by R and P respectively.

3.3 Accessibility Matrix and Path Matrix

Two matrices, namely the accessibility matrix and the path matrix, derived from a given AIM are defined in this section to provide the accessibility information from the base module to every pendant module in a branch-type modular robot. The accessibility information enables us to formulate the kinematics and dynamics of a general branch-type robot in a uniform way.

3.3.1 Module traversing order

The links and joints of a serial-type robot can follow a natural order from the base to the tip. A branch-type robot has more than one tips, and no loops. Therefore, the order of the links of a branch-type robot depends on the graph traversing algorithms (Cormen et al. 1990). Let $\mathcal{G} = (V, \mathcal{E})$ represent the kinematic graph of a branch-type modular robot with $n+1$ link modules, where $V = \{v_0, v_1, ... v_n\}$ represents the set of modules. The fixed base module is denoted by $\mathbf{v}_0$ and is always the starting point for the traversing algorithm. The rest modules are labeled by their traversing orders i. The traversing orders of the links in the robot of Figure 4(a) are indicated by the numbers on the vertices of the graph of Figure 4(b). This order is obtained by the depth-first-search algorithm. Note that the farther the module is away from the base, the larger its traversing order.

3.2.2 Directed graphs

A branch-type robot with $n+1$ modules has n joints. Let $\mathcal{E} = \{\mathbf{e}_1, \cdots, \mathbf{e}_n\}$ represents the set of joints, where joint $\mathbf{e}_i$ is designated as the connector preceding link module $\mathbf{v}_i$. With a given traversing order, the robot graph $\mathcal{G}$ can be converted to a directed graph (or digraph) $\vec{\mathcal{G}}$, which is an outward tree for a branch-type manipulator in the following manner. Let $\mathbf{e}_j = (\mathbf{v}_i, \mathbf{v}_j)$ be an edge of the graph $\vec{\mathcal{G}}$ and $i < j$. An arrow is drawn from $\mathbf{v}_i$ to $\mathbf{v}_j$ as edge $\mathbf{e}_j$ leaves vertex $\mathbf{v}_i$ and enters vertex $\mathbf{v}_j$. Suffice to say, link $\mathbf{v}_i$ precedes link $\mathbf{v}_j$. An example of the directed graph is shown in Figure 4(b). From an outward tree with n vertices, an $n \times n$ accessibility matrix can be defined to show the accessibility among the vertices.

Definition 3. (Accessibility matrix) *The accessibility matrix of a directed kinematic graph $\vec{\mathcal{G}}$ of a modular robot with $n+1$ modules (vertices) is an $(n+1) \times (n+1)$ matrix, $\mathcal{R}(\vec{\mathcal{G}}) = [r_{ij}]$ $(i, j = 0, \cdots, n)$ such that $r_{ij} = 1$, if there is a directed path of length one or more from $\mathbf{v}_i$ to $\mathbf{v}_j$; $r_{ij} = 0$, otherwise.*

The accessibility matrix can be derived from the AIM once the traversing order on the link modules is determined. For example, the accessibility matrix of $\vec{\mathcal{G}}$ in Figure 4(b) is

$$
\mathcal{R}(\vec{\mathcal{G}}) = \begin{array}{c} \\ \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \\ \mathbf{v}_6 \\ \mathbf{v}_7 \end{array} \begin{array}{c} \mathbf{v}_0\ \mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3\ \mathbf{v}_4\ \mathbf{v}_5\ \mathbf{v}_6\ \mathbf{v}_7 \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}.
\tag{2}
$$

From $\mathcal{R}(\vec{\mathcal{G}})$, we can obtain the shortest route from the base to the pendant link. This route is called a *path*. The pendant links are the rows of $\mathcal{R}(\vec{\mathcal{G}})$ with all 0s. The number of paths in a branching robot is equal to the number of pendant links. Let link $\mathbf{v}_i$ be a pendant link. All link modules on the path from the base to $\mathbf{v}_i$ are shown in the nonzero entries of column i of $(\mathcal{R}(\vec{\mathcal{G}}) + I_{(n+1)\times(n+1)})^T$. Collecting all the paths, we obtain the path matrix:

Definition 4. (Path matrix)
The path matrix $\mathcal{P}(\tilde{\mathcal{G}})$ of a directed kinematic graph $\tilde{\mathcal{G}}$ of a branch-type robot with $n+1$ link modules (vertices) and m paths is an $m\times(n+1)$ matrix, $\mathcal{P}(\vec{\mathcal{G}}) = [p_{ij}]$, $(i = 1, 2, \ldots, m;\ j = 0, 1, \ldots, n)$ such that $p_{ij} = 1$, if path i contains vertex j, and $p_{ij} = 0$ otherwise.

For instance, the robot of Figure 4(a) contains three branches (paths). The three paths can be represented as a 3×8 path matrix:

$$
\mathcal{P}(\vec{\mathcal{G}}) = \begin{array}{c} \mathbf{v}_0\ \mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3\ \mathbf{v}_4\ \mathbf{v}_5\ \mathbf{v}_6\ \mathbf{v}_7 \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}.
\tag{3}
$$

Row 1 represents the branch of the robot containing link modules $\mathbf{v}_0$, $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$; Row 2 represents the branch of $\mathbf{v}_0$ and $\mathbf{v}_4$; Row 3 represents the branch of $\mathbf{v}_0$, $\mathbf{v}_5$, $\mathbf{v}_6$, and $\mathbf{v}_7$. It can be seen that the rows of $\mathcal{P}(\vec{\mathcal{G}})$ are identical to Columns 3, 4, and 7 of $(\mathcal{R}(\vec{\mathcal{G}}) + I_{(n+1)\times(n+1)})$ respectively.

4. Geometry-Independent Models

4.1 Forward Kinematics

The forward kinematics of a general branch-type modular robot starts with a given AIM and a dyad kinematic model that relates the motion of two connected modules under a joint displacement. A dyad is a pair of connected links in a kinematic chain. Using dyad kinematics recursively with a prescribed graph-traversing order assigned to the robot modules, we may obtain the forward transformation of every branch with respect to the base frame, having a prescribed set of joint displacements. Note that a branch-type robot is one without any closed loop geometry. The kinematics of a closed loop type robot mechanism requires additional constraints, and is not considered here.

4.1.1 Dyad kinematics

Let $\mathbf{v}_i$ and $\mathbf{v}_j$ be two adjacent links connected by a joint $\mathbf{e}_j$, as shown in Figure 5. Denote joint $\mathbf{e}_j$ and link $\mathbf{v}_j$ as link assembly j and the module-coordinate frame on link $\mathbf{v}_i$ as frame i. The relative position (including the orientation) of the dyad, $\mathbf{v}_i$ and $\mathbf{v}_j$, with respect to frame i with a joint angle q_j, can be described by a 4×4 homogeneous matrix,

$$\mathbf{T}_{ij}(q_j) = \mathbf{T}_{ij}(0)e^{\hat{s}_j q_j}, \tag{4}$$

where $\hat{s}_j \in se(3)$ is the twist of joint $\mathbf{e}_j$ expressed in frame j, $\mathbf{T}_{ij}(q_{ij})$ and $\mathbf{T}_{ij}(0) \in SE(3)$. $\mathbf{T}_{ij}(0)$ is the initial pose of frame j relative to frame i. Note that in the following context, the *pose* of a coordinate frame is referred to the 4×4 homogeneous matrix of the orientation and position of a coordinate frame:

$$\mathbf{T}_{ij}(0) = \begin{bmatrix} \mathbf{R}_{ij}(0) & \mathbf{d}_{ij}(0) \\ \mathbf{0} & 1 \end{bmatrix}, \tag{5}$$

where $\mathbf{R}_{ij}(0) \in SO(3)$ and $\mathbf{d}_{ij}(0) \in R^3$ are the initial orientation and position of link frame j relative to frame i respectively. The twist $\hat{s}_j$ of link assembly j is the skew-symmetric matrix representation of the 6-vector line coordinate of the joint axis, $s_j = (\mathbf{q}_j, \mathbf{p}_j)$; $\mathbf{p}_j, \mathbf{q}_j \in R^3$. $\mathbf{p}_j = (p_{jx}, p_{jy}, p_{jz})$ is the unit-directional vector of the joint axis relative to frame j, and $\mathbf{q}_j = (q_{jx}, q_{jy}, q_{jz}) = \mathbf{p}_j \times \mathbf{r}_j$, where $\mathbf{r}_j$ is the position vector of a point along the joint axis relative to frame j. For revolute joints, $\mathbf{s}_j = (0, \mathbf{p}_j)$, and for prismatic joints, $\mathbf{s}_j = (\mathbf{q}_j, 0)$.
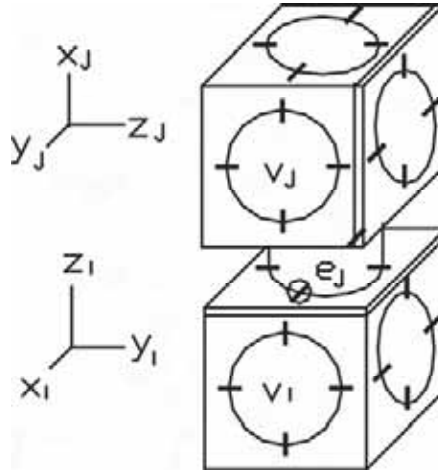
Figure 5. Link-assembly j connected to link i

4.1.2 Recursive forward kinematics

Based on eq. (4), we propose a recursive algorithm for a general branch-type modular robot, termed *TreeRobotKinematics*. This algorithm can derive the forward transformations of the base link to all pendant links based on graph-traversing algorithms. The procedure is illustrated in Figure 6. Implementation details can be found in an earlier work (Chen & Yang 1996). The algorithm takes three inputs: the AIM of the robot $\mathcal{A}(\mathcal{G})$, the base link location T_0, and a set of joint angles $\{q\}$. The forward-kinematics calculation follows the breath-first-search (BFS) traversing algorithm to travel on the connected robot modules.
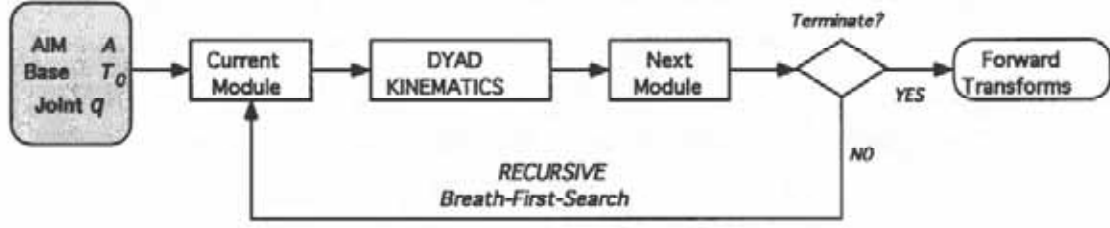
Figure 6. The TreeRobotKinematics algorithm

4.1.3 Path-by-path forward kinematics

A tree-type robot consists of several paths that give the shortest routes from the base to the respective pendant links. Each path can be considered as a serially connected *submanipulator* so that the forward transformation can be derived as conventional industrial manipulator. The sequence of the connected modules in a path is indicated in a row of the path matrix $\mathcal{P}(\vec{\mathcal{G}})$. Let $a = \{a_0, a_1, a_2, \cdots, a_n\}$ represent the links of path k. The base is $a_0 \equiv 0$ and the number of links in the path k is defined to be $|a| = n+1$. For instance, path 1 of the robot in Figure 4(a) is $a = \{0,1,2,3\}$. The forward kinematics from the base to the pendant link a_n of path k is given by

$$
\begin{aligned}
T_{a_0 a_n} &= T_{a_0 a_1}(q_{a_1}) T_{a_1 a_2}(q_{a_2}) \ldots T_{a_{n-1} a_n}(q_{a_n}) \\
&= \prod_{i=1}^{n} (T_{a_{i-1} a_i}(0) e^{\hat{s}_{a_i} q_{a_i}})
\end{aligned}
\tag{6}
$$

For a branch-type modular robot with several paths, the forward kinematics is

$$
\mathbf{T}(q_1, q_2, \ldots, q_n) =
\begin{bmatrix}
T_{a_0 a_n} \\
T_{b_0 b_m} \\
\vdots
\end{bmatrix}
=
\begin{bmatrix}
\prod_{i=1}^{n} (T_{a_{i-1} a_i}(0) e^{\hat{s}_{a_i} q_{a_i}}) \\
\prod_{i=1}^{m} (T_{b_{i-1} b_i}(0) e^{\hat{s}_{b_i} q_{b_i}}) \\
\vdots
\end{bmatrix},
\tag{7}
$$

where $\mathbf{T}(q_1, q_2, \ldots, q_n)$ represents the vector of 4×4 homogeneous matrices of the poses of all the pendant end-effectors. Since many paths in the branch-type robot share many common modules, there will be repetitive calculations using the model of eq. (7). In actual implementation, we prefer the recursive approach, which introduces no repetitive calculations.

4.2 Inverse Kinematics

The purpose of an inverse kinematics algorithm is to determine the joint angles that cause the end-effector of a manipulator to reach a desired pose. Current robot inverse kinematics algorithms can be categorized into two types: closed-form and numerical. Closed-form-type inverse kinematics requires a complete parameterization of the solution space, usually in terms of simultaneous polynomial equations. Solutions to such a set of simultaneous polynomial solutions exist for a few types of robots with revolute joints or simple geometry. It is very difficult to obtain the inverse kinematics for an arbitrary modular reconfigurable robot in this manner. Here we adopt the numerical approach to solve the inverse kinematics of modular robots. The inverse-kinematics algorithm will construct the differential kinematic model using the local POE formula. The differential kinematic equation of a single branch of a branch-type robot is considered first. Based on the AIM of the robot, one can extend this differential kinematics model to include multiple branch structures. Then the Newton-Raphson iteration method is used to obtain the numerical inverse kinematics solutions. The differential kinematic model can be easily modified to solve the pure position, pure orientation, and hybrid inverse kinematics problems (Chen & Yang 1999).

4.2.1 A Single branch

Let $T_{a_0 a_n}$ be the forward transformation of path k as indicated in eq. (6). The differential change in the position of the end-link a_n can be given by

$$
\begin{aligned}
dT_{a_0 a_n} &= \sum_{i=1}^{|a|-1} \frac{\partial T_{a_0 a_n}}{\partial q_{a_i}} dq_{a_i} \\
&= \sum_{i=1}^{|a|-1} \left[T_{a_0 a_{i-1}} \frac{\partial (T_{a_{i-1} a_i}(0) e^{\hat{s}_{a_i} q_{a_i}})}{\partial q_{a_i}} T_{a_i a_n} \right] dq_{a_i} \\
&= \sum_{i=1}^{|a|-1} \left[T_{a_0 a_i} \hat{s}_{a_i} T_{a_i a_n} \right] dq_{a_i}
\end{aligned}
\tag{8}
$$

Left-multiplying $T_{a_0 a_n}^{-1}$, eq. (8) becomes,

$$
T_{a_0 a_n}^{-1} dT_{a_0 a_n} = \sum_{i=1}^{|a|-1} T_{a_i a_n}^{-1} \hat{s}_{a_i} T_{a_i a_n} dq_{a_i} .
\tag{9}
$$

Equation (9) is the differential kinematic equation of a path. Let $T_{a_0 a_n}^d$ denote the desired position of the end-effector. When it is in the neighborhood of a nominal position of the end-effector $T_{a_0 a_n}$, we have

$$dT_{a_0 a_n} = T_{a_0 a_n}^d - T_{a_0 a_n} \,.$$
(10)

Left-multiplying $T_{a_0 a_n}^{-1}$ to eq. (9), and using the matrix logarithm,

$$\log(T_{a_0 a_n}^{-1} T_{a_0 a_n}^d) = (T_{a_0 a_n}^{-1} T_{a_0 a_n}^d - I) - \frac{(T_{a_0 a_n}^{-1} T_{a_0 a_n}^d - I)^2}{2} + \frac{(T_{a_0 a_n}^{-1} T_{a_0 a_n}^d - I)^3}{3} - \cdots$$
(11)

We can obtain the following equation by first order approximation:

$$T_{a_0 a_n}^{-1} dT_{a_0 a_n} = \log(T_{a_0 a_n}^{-1} T_{a_0 a_n}^d) \,.$$
(12)

Substituting eq. (12) into eq. (9), we obtain

$$\log(T_{a_0 a_n}^{-1} T_{a_0 a_n}^d) = \sum_{i=1}^{|a|-1} T_{a_0 a_n}^{-1} \hat{s}_{a_i} T_{a_0 a_n} dq_{a_i} \,.$$
(13)

Explicit formulae for calculating the logarithm of elements of $SO(3)$ and $SE(3)$ were derived by Park and Bobrow (1994). Definitely, $\log(T_{a_0 a_n}^{-1} T_{a_0 a_n}^d)$ is an element of $se(3)$ so that it can be identified by a 6×1 vector denoted by $\log(T_{a_0 a_n}^{-1} T_{a_0 a_n}^d)^\vee$ in which the first and later three elements represent the positional and orientational *differences* between $T_{a_0 a_n}$ and $T_{a_0 a_n}^d$. Converting eq. (13) into the adjoint representation, we get

$$\log(T_{a_0 a_n}^{-1} T_{a_0 a_n}^d)^\vee = Ad_{T_{a_0 a_n}^{-1}} \sum_{i=1}^{|a|-1} Ad_{T_{a_0 a_i}} s_{a_i} dq_{a_i} \,.$$
(14)

Conveniently, eq. (14) can also be expressed as the following form:

$$D_{T_k} = J_k dq_k \qquad\qquad\qquad ,$$
(15)

where

$D_{T_k} = \log(T_{a_0 a_n}^{-1} T_{a_0 a_n}^d)^\vee \in R^{6 \times 1}$ is referred as the *pose difference vector* for path k ;
$J_k = A_k B_k S_k \in R^{6 \times (|a|-1)}$, is termed as *body manipulator Jacobian matrix* (Murray et al. 1994);
$A_k = Ad_{T_{a_0 a_n}^{-1}} \in R^{6 \times 6}$;

$$B_k = \text{row}[Ad_{T_{a_0 a_1}}, Ad_{T_{a_0 a_2}}, \ldots, Ad_{T_{a_0 a_n}}] \in R^{6 \times 6(|a|-1)} \; ;$$

$$S_k = \text{diag}[s_{a_1}, s_{a_2}, \ldots, s_{a_n}] \in R^{6(|a|-1) \times (|a|-1)}] \; ; \text{and}$$

$$dq_k = \text{column}[dq_{a_1}, dq_{a_2}, \ldots, dq_{a_n}] \in R^{(|a|-1) \times 1}.$$

Equation (15) defines the differential kinematics for path k. It can be utilized in the Newton-Raphson iteration to obtain an inverse kinematics solution for a given pose.

4.2.2 Entire manipulator

The paths of a branch-type manipulator may not be independently driven, because of the common sharing modules. This forbids us to treat each path as independent serial-type manipulators. Hence, with a given set of the pendant end-effectors's poses for all branches, the inverse kinematics must be solved simultaneously. With the assistance of the path matrix, we are able to identify the connected and related modules in a path. Then, we can orderly combine the differential kinematic equations (eq. (15)) ofall constituting paths into a single matrix equation of the following form:

$$D_T = Jdq \qquad\qquad\qquad\qquad , \qquad\qquad\qquad (16)$$

where

$D_T = \text{column}[D_{T_1}, D_{T_2}, \ldots, D_{T_m}] \in R^{6m \times 1}$, is termedthe *generalized pose difference vector*;

$J = ABS \in R^{6m \times n}$, is termed the *generalized body manipulator Jacobian matrix*;

$A = \text{diag}[A_1, A_2, \ldots, A_m] \in R^{6m \times 6m}$; and

$$B = \begin{bmatrix} p_{11}Ad_{T_{01}} & p_{12}Ad_{T_{02}} & \ldots & p_{1n}Ad_{T_{0n}} \\ p_{21}Ad_{T_{01}} & p_{22}Ad_{T_{02}} & \ldots & p_{2n}Ad_{T_{0n}} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1}Ad_{T_{01}} & p_{m2}Ad_{T_{02}} & \ldots & p_{mn}Ad_{T_{0n}} \end{bmatrix} \in R^{6m \times 6n}$$

The coefficient, $p_{ij}(i=1,2,\ldots,m; j=0,1,2,\ldots,n)$ is entry (i,j) of the *path* matrix $\mathcal{P}$, and m is the total number of paths; $S = \text{diag}[s_1, s_2, \ldots, s_n] \in R^{6n \times n}$; $dq = \text{column}[dq_1, dq_2, \ldots, dq_n] \in R^{n \times 1}$.

Rewriting this equation in an iterative form, we get

$$dq^{i+1} = J^* D_T \qquad\qquad\qquad\qquad\qquad\qquad\qquad (17)$$

$$q^{i+1} = q^i + dq^{i+1}, \qquad\qquad\qquad\qquad\qquad\qquad (18)$$

where i represents the number of iterations and J^* is the Moore-Penrose pseudoinverse of J. Using the Newton-Raphson method, a close-loop iterative algorithm similar to that of Khosla, Newman and Prinz (1985) is employed (Fig. 7). The iterative algorithm determines the necessary changes in the joint angles to achieve a differential change in the position and orientation of the end-effector. Given a complete robot assembly (or the AIM) and a set of desired poses T^d, this algorithm starts from an initial guess, q^0, somewhere in the neighborhood of the desired solution. It is terminated when a prescribed termination criteria is reached. As one can see, the structure of J depends on the path matrix, which is implied in the kinematic graph of the robot. Therefore, once the assembly configuration of a modular robot is determined and all module parameters are obtained, the differential kinematic model (eq. (16)) can be generated automatically.

Computational examples of the inverse kinematics algorithms for branch-type and serial modular robots are given by Chen & Yang (1999) to illustrate the algorithm's applicability and effectiveness. When compared to the other numerical inverse kinematics algorithm using D-H parameters, our method always use less number of iterations and computing time for the same given pose. This is due to the use of the *pose difference vector* computed from the matrix logarithm in eq. (16), and not the difference of homogeneous transformation matrices. Actual implementation of the algorithm using C++ codes shows that the computation time for each solution can take less than 20 msec on a Pentium II 300MHz PC, which satisfies the basic requirement for real-time control and simulation.
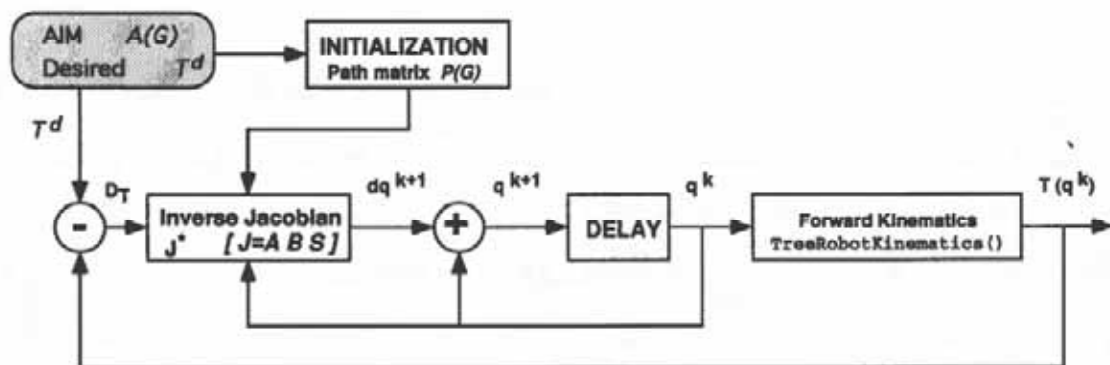
Figure 7. Inverse kinematics algorithm

4.3 Kinematic Calibration

The machining tolerance, compliance, and wear of the connected mechanism and mis-alignment of the connected module components may introduce errors in positioning the end-effector of a modular robot. Hence, calibrating the kinematic parameters of a modular robot to enhance is positioning accuracy is important, especially in high precision application such as hard-disk assembly. Current kinematic calibration algorithms for industrial robots that are designed for certain types of serial manipulators are not suitable for modular robots with arbitrary geometry. Here we propose a general singularity-free calibration-modeling method for modular reconfigurable robots, based on the forward kinematics discussed in previous section. This method follows local POE formulae. The robot errors are assumed to be in the initial positions of the consecutive modules. Based on linear superposition and differential transformation, a six-parameter model is derived. This model can be generated automatically once the AIM of the robot is given. An iterative least-square algorithm is then employed to find the error parameters to be corrected.

The calibration starts with a serial-type manipulator kinematics model:

$$
\begin{aligned}
T_{0n}(\mathbf{q}) &= T_{01}(q_1)\,T_{12}(q_2)\cdots T_{n-1,n}(q_n) & (19)\\
&= T_{01}(0)e^{\hat{s}_1 q_1}\,T_{12}(0)e^{\hat{s}_2 q_2}\cdots T_{n-1,n}(0)e^{\hat{s}_n q_n} & (20)
\end{aligned}
$$

Extension to a general branch-type modular robot is similar to the treatment of the inverse-kinematics model in previous section. Basically, eq. (20) can be treated as a function of the joint angles, $\mathbf{q} = (q_1, \cdots, q_n)$, locations of the joint axes, $\hat{\mathbf{s}} = (\hat{s}_1, \cdots, \hat{s}_n)$, and the relative initial positions of the dyads, $T_0 = (T_{01}(0), \cdots, T_{n-1,n}(0))$:

$$
T_{0n} = f(T_0, \hat{\mathbf{s}}, \mathbf{q}) . \tag{21}
$$

Written in differential form, we have

$$
dT_{0n} = \frac{\partial f}{\partial T_0} dT_0 + \frac{\partial f}{\partial \hat{\mathbf{s}}} d\hat{\mathbf{s}} + \frac{\partial f}{\partial \mathbf{q}} d\mathbf{q} . \tag{22}
$$

The differential dT_{0n} can be interpreted as the difference between the nominal position and the measured position.

4.3.1 Error model of a dyad

Our kinematic calibration is based on the local frame representation of a dyad

described in eq. (4). Two assumptions are made in the dyad of link $\mathbf{v}_{i-1}$ and $\mathbf{v}_i$ of a modular robot chain: first, small geometric errors only exist in the initial position $T_{i-1,i}(0)$; second, the twist and joint angle q_i assume the nominal values through out the calibration analysis. Hence, instead of identifying the module's actual initial positions, joint twists and angle offsets, we look for a new set of local initial positions (local frames, called calibrated initial positions), in the calibration model, so that the twist of the joint remains the nominal value. In other words, the errors in a dyad are lumped with the initial position. Therefore, $d\hat{\mathbf{s}}$ and $d\mathbf{q}$ can be set to 0. Because $SE(3)$ has the dimension of six---three for positions and three for orientations---there can be only six independent quantities in $T_{i-1,i}(0)$, and there will be six independent error parameters in a dyad. Denote the small error in the initial position of dyad $(\mathbf{v}_{i-1}, \mathbf{v}_i)$ as $dT_{i-1,i}(0)$, then

$$
\begin{aligned}
dT_{i-1,i}(0) &= T_{i-1,i}(0)\,\hat{\Delta}_i \\
\hat{\Delta}_i &= \begin{bmatrix}
0 & -\delta z_i & \delta y_i & dx_i \\
\delta z_i & 0 & -\delta x_i & dy_i \\
-\delta y_i & \delta x_i & 0 & dz_i \\
0 & 0 & 0 & 0
\end{bmatrix},
\end{aligned}
\tag{23}
$$

where dx_i, dy_i, and dz_i are infinitesimal displacements along x-, y-, and z-axes of link frame i respectively, and δx_i, δy_i and δz_i are infinitesimal rotations about x-, y-, and z-axes of link frame i respectively.

4.3.2 Gross error model of a robot

Similar to the error model of a dyad, the gross-geometric error, dT_{0n} between the actual end-effector position the nominal position can be described as:

$$
dT_{0n} = \hat{\Delta}_{0n} T_{0n}
\tag{24}
$$

and

$$
\begin{aligned}
\hat{\Delta}_{0n} &= dT_{0n} T_{0n}^{-1} \\
&= \begin{bmatrix}
0 & -\delta z_{0n} & \delta y_{0n} & dx_{0n} \\
\delta z_{0n} & 0 & -\delta x_{0n} & dy_{0n} \\
-\delta y_{0n} & \delta x_{0n} & 0 & dz_{0n} \\
0 & 0 & 0 & 0
\end{bmatrix},
\end{aligned}
\tag{25}
$$
$$
\tag{26}
$$

where δx_{0n}, δy_{0n}, δz_{0n} are the rotations about the axes of the base frame, and

dx_{0n}, dy_{0n}, and dz_{0n} are the displacements along the axes of base frame respectively. Note that the gross error, dT_{0n}, is expressed in the base frame. Equation (25) follows the left multiplicative differential transformation of T_{0n}. The calibrated position of the end-effector becomes

$$T_{0n}'(q) = T_{0n} + dT_{0n}. \tag{27}$$

4.3.3 Linear superposition

Based on the assumptions, the errors in the dyads will contribute to the gross error in the end-effector's position dT_{0n}. Since the geometric errors are all very small, the principle of linear superposition can be applied. We assume that the gross errors dT_{0n} are the linear combination of the errors in the dyads $dT_{i-1,i}(0)$, $(i = 1, 2, ..., n)$; then

$$dT_{0n} = \sum_{i=1}^{n} T_{0,i-1} dT_{i-1,i}(0) e^{\hat{s}_i q_i} T_{i,n}. \tag{28}$$

Equation (28) converts and sums the initial position errors of the dyads in the base-frame coordinates. The forward kinematics of link-frame j relative to link-frame i $(i \leq j)$ is represented by T_{ij}. Especially, $T_{ij} = I_{4\times4}$ when $i = j$. Substituting eq. (23) into eq. (28), and right-multiplying T_{0n}^{-1},

$$dT_{0n} T_{0n}^{-1} = \hat{\Delta}_{0n} \tag{29}$$

$$= \sum_{i=1}^{n} T_{0,i-1} T_{i-1,i}(0) \hat{\Delta}_i T_{i-1,i}^{-1} T_{0,i-1}^{-1} \tag{30}$$

From the first order approximation, we have

$$\hat{\Delta}_{0n} = dT_{0n} T_{0n}^{-1} \approx \log(T_{0n}'T_{0n}^{-1}). \tag{31}$$

Converting eq. (31) into the adjoint representation, we have

$$\log^{\vee}(T_{0n}'T_{0n}^{-1}) = \sum_{i=1}^{n} Ad_{T_{0,i-1}}(Ad_{T_{i-1,i}(0)}(\Delta_i)). \tag{32}$$

Equation (32) can also be expressed in the following matrix form

$$y = \mathbf{A}x, \tag{33}$$

where

$y = \log^{\vee}(T_{0n}'T_{0n}^{-1}) \in R^{6\times1}$;

$x = \text{column}[\Delta_1, \Delta_2, \cdots, \Delta_n] \in R^{6n\times1}$; and

$\mathbf{A} = \text{row}[Ad_{T_{0,1}(0)}, Ad_{T_{0,1}}(Ad_{T_{1,2}(0)}), \cdots, Ad_{T_{0,n-1}}(Ad_{T_{n-1,n}(0)})] \in R^{6\times6n}$.

In Equation (33), x represents the error parameters to be identified in a modular robot assembly. The quantities in matrix $\mathbf{A}$ and T_{0n}^{-1} are determined from the nominal model. T_{0n}' comes from the actual measured data. To improve the accuracy of the calibration model, the kinematic calibration procedure usually requires the position of the end-effector to be measured in several different robot postures. For the ith measurement, we obtain y_i and $\mathbf{A}_i$. After taking m measurements,

$$\tilde{\mathbf{Y}} = \tilde{\mathbf{A}}x , \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (34)$$

Where

$\tilde{\mathbf{Y}} = \text{column}[y_1, y_2, \cdots, y_m] \in R^{6m\times1}$; and

$\tilde{\mathbf{A}} = \text{column}[\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_m] \in R^{6m\times6n}$.

The least-squares solution for x can be obtained by

$$x = \tilde{\mathbf{A}}^{\dagger}\tilde{\mathbf{Y}} , \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (35)$$

where $\tilde{\mathbf{A}}^{\dagger}$ is the pseudo-inverse of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{A}}^{\dagger} = (\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^T$ for $m > n$; $\tilde{\mathbf{A}}^{\dagger} = \tilde{\mathbf{A}}^T(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^{-1}$ for $m < n$; $\tilde{\mathbf{A}}^{\dagger} = \tilde{\mathbf{A}}^{-1}$ for $m = n$.

The calibration procedure is illustrated in the diagram of Figure 8(a). Computer simulation and actual experiment on the modular robot systems described by Chen & Yang (1997) and Chen et al. (2001) have shown that this calibration method can improve the accuracy in end-effector positioning by up to two orders of magnitudes.

(a)   **(b)**
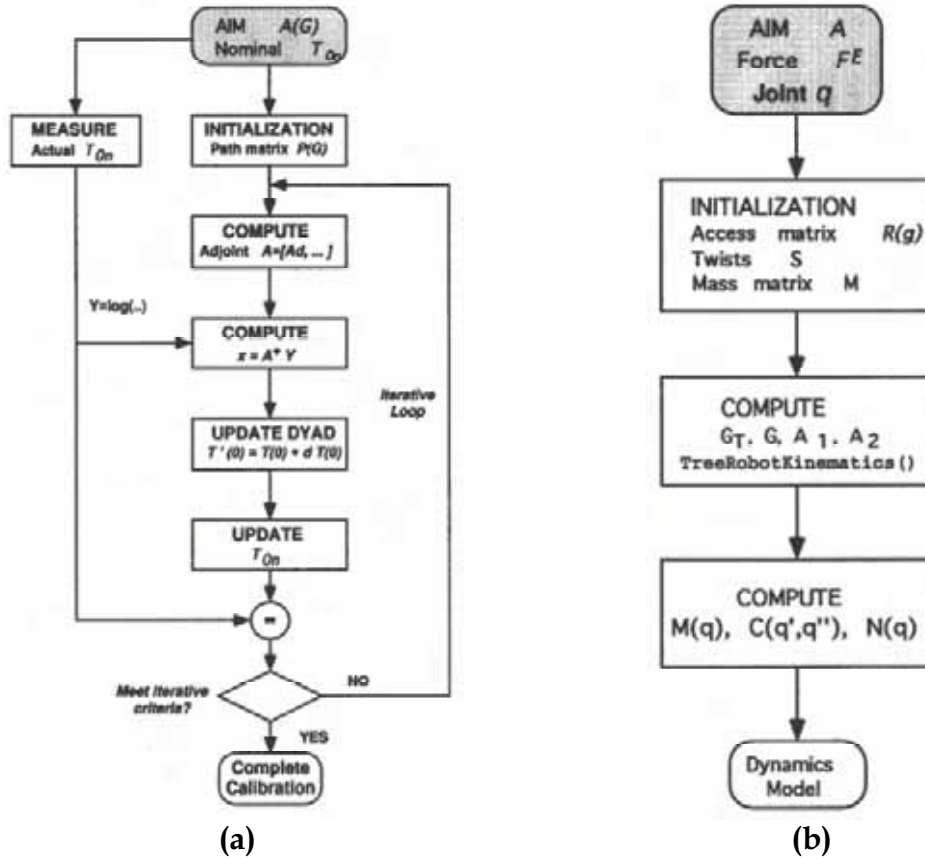
Figure 8. (a) Calibration algorithm for modular robots; (b) Dynamic model generation

4.4 Dynamics

The dynamic model of a robot can be formulated with an iterative method through a recursive Newton-Euler equation. This method can be generally applied to branch-type robots without modification. Here we present a method to generate the closed-form dynamic models of modular robots using the AIM and the recursive algorithm.

4.4.1 Newton-Euler Equation for link assembly

Assume that the mass center of link assembly j is coincident with the origin of the link module frame j. The Newton-Euler equation of this rigid link assembly with respect to frame j is (Murray et al 1994)

$$\mathbf{F}_j = \begin{bmatrix} f_j \\ \tau_j \end{bmatrix} = \begin{bmatrix} m_j I & 0 \\ 0 & J_j \end{bmatrix} \begin{bmatrix} \dot{v}_j \\ \dot{w}_j \end{bmatrix} + \begin{bmatrix} w_j \times m_j v_j \\ w_j \times J_j w_j \end{bmatrix}, \tag{36}$$

where $\mathbf{F}_j \in R^{6 \times 1}$ is the resultant wrench applied to the center of mass relative to frame j. The total mass of link assembly j is m_j (which is equal to the sum of link $\mathbf{v}_j$ and joint $\mathbf{e}_j$). The inertia tensor of the link assembly about frame j is J_j. Transforming eq. (36) into the adjoint representation, we have

$$\mathbf{F}_j = M_j \dot{V}_j - ad_{V_j}^T (M_j V_j) . \tag{37}$$

The following notations are adopted:

- $M_j = \begin{bmatrix} m_j & 0 \\ 0 & J_j \end{bmatrix} \in R^{6 \times 6}$ is the generalized mass matrix;

- $V_j = \begin{bmatrix} v_j \\ w_j \end{bmatrix} \in R^{6 \times 1}$ is the generalized body velocity, where v_j and w_j are 3×1 vectors defining body translational velocity, $v_j = (v_x, v_y, v_z)^T$, and the angular velocity, $w_j = (w_x, w_y, w_z)^T$, respectively;

- $ad_{V_j}^T \in R^{6 \times 6}$ is the transpose of adjoint matrix ad_{V_j} related to V_j

$$ad_{V_j}^T = (ad_{V_j})^T = \begin{bmatrix} \hat{w}_j & \hat{v}_j \\ 0 & \hat{w}_j \end{bmatrix}^T = \begin{bmatrix} -\hat{w}_j & 0 \\ -\hat{v}_j & -\hat{w}_j \end{bmatrix} ; \tag{38}$$

- $\hat{v}_j$ and $\hat{w}_j \in R^{3 \times 3}$ are skew-symmetric matrices related to v_j and w_j respectively; and $\dot{V}_j = \begin{bmatrix} \dot{v}_j \\ \dot{w}_j \end{bmatrix} \in R^{6 \times 1}$ is the generalized body acceleration.

4.4.2 Recursive Newton-Euler algorithm

The recursive algorithm is a two-step iteration process. For a branch-type robot, the generalized velocity and acceleration of each link are propagated from the base to the tips of all branches. The generalized force of each link is propagated backward from the tips of the branches to the base. At the branching module, generalized forces transmitted back from all branches are summed.

FORWARD ITERATION
The generalized velocity and acceleration of the base link are given initially,

$$V_b = V_0 = (0,0,0,0,0,0)^T \tag{39}$$
$$\dot{V}_b = \dot{V}_0 = (0,0,g,0,0,0)^T \tag{40}$$

where V_b and $\dot{V}_b$ are expressed in the base frame 0. We assume that the base frame coincides with the spatial reference frame. The generalized acceleration (eq. (40)) is initialized with the gravitation acceleration g to compensate for the effect of gravity. Referring to Figure 6, the recursive body velocity and acceleration equations can be written as

$$V_j = Ad_{T_{ij}^{-1}}(V_i) + s_j \dot{q}_j \tag{41}$$

$$\dot{V}_j = Ad_{T_{ij}^{-1}}(\dot{V}_i) + ad_{Ad_{T_{ij}^{-1}}(V_i)}(s_j \dot{q}_j) + s_j \ddot{q}_j \tag{42}$$

where all the quantities, if not specified, are expressed in link frame j.

- V_j and $\dot{V}_j$ are the generalized velocity and acceleration of link-assembly j;
- $\dot{q}_j$ and $\ddot{q}_j$ are the velocity and acceleration of joint $\mathbf{e}_j$ respectively;
- $Ad_{T_{ij}^{-1}}$ is the adjoint representation of $T_{ij}^{-1}(q_j)$, where $T_{ij}(q_j) \in SE(3)$ is the position of frame j relative to frame i with joint angle q_j and $Ad_{T_{ij}^{-1}} = (Ad_{T_{ij}})^{-1}$; and
- $s_j \in R^{6\times 1}$ is the twist coordinates of joint $\mathbf{e}_j$.

BACKWARD ITERATION

The backward iteration of the branch-type robot starts simultaneously from all the pendant link assembly. Let $\mathcal{V}_{PD} \subset \mathcal{V}$ be set of the pendant links of the branch-type robot. For every pendant link assembly d_i ($\mathbf{v}_{d_i} \in \mathcal{V}_{PD}$), the Newton-Euler equation (eq. (37)) can be written as

$$F_{d_i} = -F_{d_i}^e + M_{d_i}\dot{V}_{d_i} - ad_{V_{d_i}}^T(M_{d_i}V_{d_i}), \tag{43}$$

where F_{d_i} is the wrench exerted on link-assembly $\mathbf{v}_{d_i}$ by its parent (preceding) link relative to frame d_i; and $F_{d_i}^e$ is the external wrench exerted on v_{d_i}. Note that the total wrench is $\mathbf{F}_{d_i} = F_{d_i} + F_{d_i}^e$. Now traverse the links in the robot backward from the pendant links. Let $\mathcal{V}_{Hi}$ be the set of successors of link $\mathbf{v}_i$. For every link assembly i, the Newton-Euler equation (eq. (37)) can be written in the following form:

$$F_i = \sum_{j \in V_{Hi}} Ad_{T_{ij}^{-1}}^T(F_j) - F_i^e + M_i \dot{V}_i - ad_{V_i}^T(M_i V_i), \tag{44}$$

where all quantities, if not specified, are expressed in link-frame i; $F_i \in R^{6\times 1}$ is the wrench exerted *to* link-assembly i by its predecessor; $F_j \in R^{6\times 1}$ is the wrench exerted *by* link-assembly i to the successor $\mathbf{v}_j \in \mathcal{V}_{Hi}$ expressed in link-frame j; F_i^e is the external wrench applied to link-assembly i. The total wrench is $\mathbf{F}_i = F_i - \sum_{j \in V_{Hi}} Ad_{T_{ij}^{-1}}^T (F_j) + F_i^e$.

The applied torque/force to link assembly i by the actuator at its input joint $\mathbf{e}_i$, can be calculated by

$$\tau_i = s_i^T F_i. \tag{45}$$

4.4.3 Closed-Form Equations of Motion

By iteratively expanding the recursive Newton-Euler equations (eqs. (39)-(44)) in the body coordinates, we obtain the generalized velocity, generalized acceleration, and generalized force equations in matrix form:

$$\mathbf{V} = \mathbf{GS\dot{q}} \tag{46}$$

$$\dot{\mathbf{V}} = \mathbf{G_{T_0}}\dot{\mathbf{V}}_0 + \mathbf{GS\ddot{q}} + \mathbf{GA_1V} \tag{47}$$

$$\mathbf{F} = \mathbf{G}^T\mathbf{F}^E + \mathbf{G}^T\mathbf{M}\dot{\mathbf{V}} + \mathbf{G}^T\mathbf{A_2MV} \tag{48}$$

$$\mathbf{t} = \mathbf{S}^T\mathbf{F} \tag{49}$$

where
- $\mathbf{V} = \text{column}[V_1, V_2, \cdots, V_n] \in R^{6n\times 1}$ is the generalized body-velocity vector;
- $\dot{\mathbf{V}} = \text{column}[\dot{V}_1, \dot{V}_2, \cdots, \dot{V}_n] \in R^{6n\times 1}$ is the generalized body-acceleration vector;
- $\mathbf{F} = \text{column}[F_1, F_2, \cdots, F_n] \in R^{6n\times 1}$ is the body-wrench vector;
- $\mathbf{t} = \text{column}[\tau_1, \tau_2, \cdots, \tau_n] \in R^{n\times 1}$ is the applied joint-torque/force vector;
- $\dot{\mathbf{q}} = \text{column}[\dot{q}_1, \dot{q}_2, \cdots, \dot{q}_n] \in R^{n\times 1}$ is the joint-velocity vector;
- $\ddot{\mathbf{q}} = \text{column}[\ddot{q}_1, \ddot{q}_2, \cdots, \ddot{q}_n] \in R^{n\times 1}$ is the joint-acceleration vector;
- $\dot{\mathbf{V}}_0 = (0,0,g,0,0,0)^T \in R^{6\times 1}$ is the generalized acceleration of the base link;
- $\mathbf{S} = \text{diag}[s_1, s_2, \cdots, s_n] \in R^{6n\times n}$ is the joint-twist matrix in the respective body coordinates;
- $\mathbf{M} = \text{diag}[M_1, M_2, \cdots, M_n] \in R^{6n\times 6n}$ is the total generalized-mass matrix;
- $\mathbf{A_1} = \text{diag}[-ad_{s_1\dot{q}_1}, -ad_{s_2\dot{q}_2}, \cdots, -ad_{s_n\dot{q}_n}] \in R^{6n\times 6n}$;
- $\mathbf{A_2} = \text{diag}[-ad_{V_1}^T, -ad_{V_2}^T, \cdots, -ad_{V_n}^T] \in R^{6n\times 6n}$;
- $\mathbf{F}^E = \text{column}[F_1^e, F_2^e, \cdots, F_n^e] \in R^{6n\times 1}$ is the external wrench vector;

$$\mathbf{G_{T_0}} = \begin{bmatrix} Ad_{T_{01}^{-1}} \\ Ad_{T_{02}^{-1}} \\ \vdots \\ Ad_{T_{0n}^{-1}} \end{bmatrix} \in R^{6n \times 6} \text{ ;and}$$

$$\mathbf{G} = \begin{bmatrix} I_{6 \times 6} & 0 & 0 & \cdots & 0 \\ r_{12}Ad_{T_{12}^{-1}} & I_{6 \times 6} & 0 & \cdots & 0 \\ r_{13}Ad_{T_{13}^{-1}} & r_{23}Ad_{T_{23}^{-1}} & I_{6 \times 6} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{1n}Ad_{T_{1n}^{-1}} & r_{2n}Ad_{T_{2n}^{-1}} & r_{3n}Ad_{T_{3n}^{-1}} & \cdots & I_{6 \times 6} \end{bmatrix} \in R^{6n \times 6n} .$$

Note that $\mathcal{R}(\vec{\mathcal{G}}) = [r_{ij}] \in R^{(n+1) \times (n+1)}$ is the accessibility matrix. The matrix $\mathbf{G}$ is called a *transmission matrix*. Substituting eqs. (46)-(48) into eq. (49), we obtain the closed-form equation of motion for a branch-type modular robot with $n+1$ modules (including the base module)

$$\mathbf{M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) = t} \tag{50}$$

where

$$\mathbf{M(q)} = \mathbf{S}^T\mathbf{G}^T\mathbf{MGS} \tag{51}$$

$$\mathbf{C(q,\dot{q})} = \mathbf{S}^T\mathbf{G}^T(\mathbf{MGA_1 + A_2M})\mathbf{GS} \tag{52}$$

$$\mathbf{N(q)} = \mathbf{S}^T\mathbf{G}^T\mathbf{MG_{T_0}}\dot{\mathbf{V}}_0 + \mathbf{S}^T\mathbf{G}^T\mathbf{F^E} \tag{53}$$

The mass matrix is $\mathbf{M(q)}$; $\mathbf{C(q,\dot{q})}$ represents the centrifugal and Coriolis accelerations; $\mathbf{N(q)}$ represents the gravitational force and external forces. The procedure for obtaining the closed-form equation (eq. (50)) is summarized in Figure 8(b). It has been successfully implemented in Mathematica code.

5. Configuration Optimization

Introducing modularity in a robotic system implies that the system perform-ance can be optimized through proper selection and reconfiguration of module components. The task planner for the modular robotic workcell will be able to determine the optimal robot configuration and geometry for a given task from an inventory of robot modules. Figure 9 depicts the general approach for de-termining the optimal assembly configuration. Shaded blocks represent the basic formulation of the optimization problem. With a given set of modules se-lected from the component database, all possible and unique assembly con-

figurations can be generated and identified through an enumeration algorithm (Chen & Burdick 1998). In the second step, an objective function is formulated to evaluate the performance of every assembly configuration, based on the task specifications. A basic robot task contains task specifications that are provided by the task planner---the goal positions/orientations, force application, accuracy, and dexterity of the end-effectors---and constraints to be overcome---obstacle avoidance, workspace limit, singularity and kinematic redundancy (Chen & Burdick 1995; Yang & Chen 2001). A search/optimization procedure is employed in the last step to find the optimal assembly configuration.
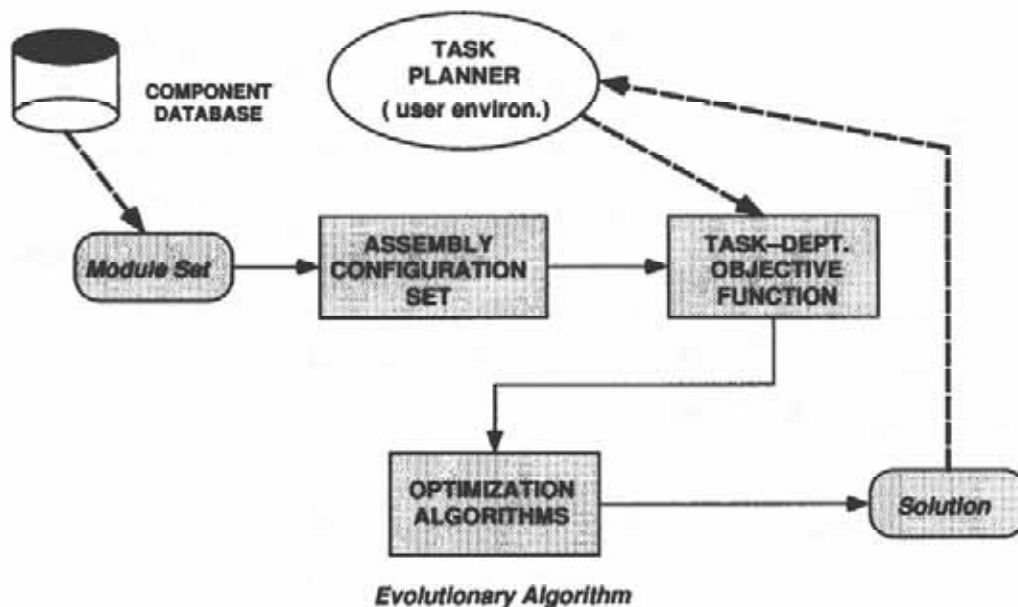
Figure 9. Determination of a task-optimal configuration

Note that all the dimensions of the modules have been previously designed and fixed at the selection stage. With a given set of modules, the possible combination of robot-assembly configurations is always a finite number. Therefore, the parameter space for the optimization is discrete, and combinatorial optimization methods can be applied. Exhaustive search algorithms can be used to find the exact optimal solution, but the exponential growth of the data set impedes the efficient implementation of such an algorithm. Random-search techniques such as genetic algorithms (GA) (Chen 1994) and simulated annealing (SA) (Paredis & Khosla 1995) are more suitable for such problems. Transition rules for data points required in GA and SA can be easily implemented based on a data-representation scheme such as AIM.
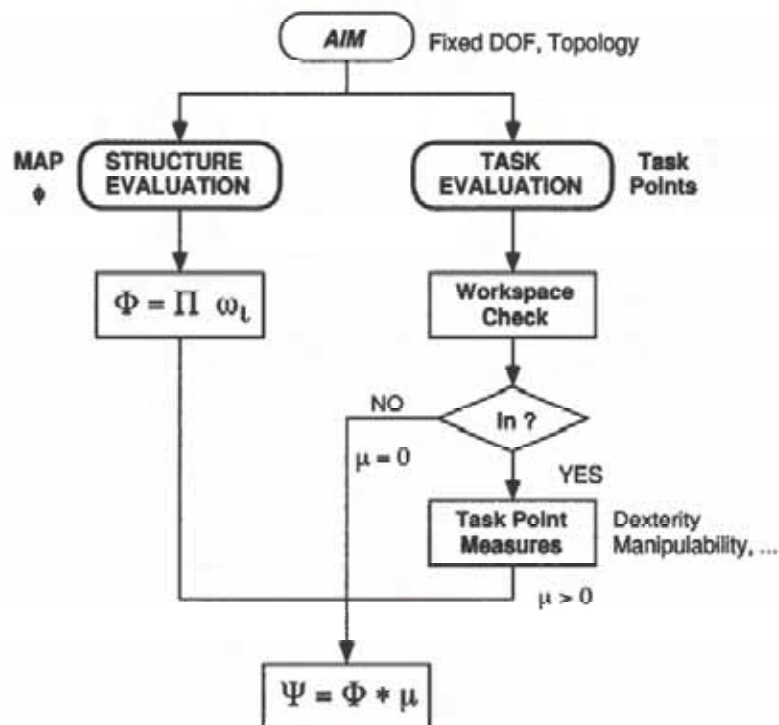
Figure 10. The ACEF for serial modular robots

5.1 Task-Oriented Objective Function

The crucial point in determining the optimal robot configuration is formulating an objective function that will assign a "goodness" value to every assembly configuration accomplishing a specified task. The form of the objective function should be general enough so that it is applicable to a wide variety of task requirements. Two components of a robot task---task specifications and constraints---must be considered in formulating the objective function. We call this function an *assembly configuration evaluation function* (ACEF). The assembly configuration with the greatest ACEF value is deemed optimal. It is also important to note that from a given set of modules it is possible to construct robots with various topologies, such as serial or parallel kinematic structures. Even with a fixed robot-topology class, the number of degrees of freedom (DOF) can alter the kinematic functionality of the system. Here we propose a solution strategy for modular robot with a fixed topology and a fixed number of DOF.

The structure of the ACEF for a serial modular robot is shown in Figure 10. The input is an AIM with a predefined number of DOFs and predefined topology. The output is the ``goodness'' of the AIM in terms of a non-negative real

number. An AIM with a large ACEF value represents a good assembly configuration. The ACEF consists of two parts: task and structure evaluations. Task evaluation is performed according to the given task specifications: the task points (or the positions of the end-effector) and a designated criteria measure, such as the dexterity or the manipulability. A workspace check on the task points is executed before computing the measures for filtering out inaccessible points. Structure evaluation assesses the kinematic constraints (joint singularity and redundancy, link interference) and environmental constraints (workspace obstacles) imposed on the robot in accomplishing the assigned task. The proposed ACEF assumes the modular robot is operated in a structured environment, and that there are no obstacles in the workspace. An auxiliary function, termed the *module-assembly preference* (MAP) is defined on the AIM to exclude undesirable kinematic features. Detailed implementation of the task and structure evaluation can be obtained from Chen (1996).

5.2 Evolutionary Algorithms

An evolutionary algorithm is a probabilistic search/optimization method based on the principle of evolution and hereditary of nature systems (Michalewicz 1994). In this algorithm, a population of individuals for each generation is maintained. The individual is implemented with some data structure and is evaluated by a ``fitness function'' to give a measure of its ``fitness''. A new population is formed by selecting the more suitable individuals. Members in the new population undergo transformations by the ``genetic operators'' to form new solutions. Through structured random information changes, the new generation is more ``fit'' than the previous generation. After a number of iterations, the individuals will converge to an optimal or near-optimal solution. Here we attempt to use the AIMs as the data structure of the solution, and define AIM-related genetic operators (Chen 1996) as solving the task-optimal problem in an evolutionary approach, because AIM is a natural representation of the modular robot and is topologically independent.

Figure 11 depicts the application of the evolutionary algorithm in solving the task-optimal configuration problem. An example of optimizing the configuration of a 4-DOF modular robot is provided in the following example. Suppose we wish to find a 4-DOF fixed-base serial robot with revolute joints that passes through two task points $\mathbf{p}_1$ and $\mathbf{p}_2$. Also suppose that we require there be no redundant joints, and minimum link interference. Let the performance measure of the robot be the manipulability. The initial set of AIMs randomly generated is shown in Figure 12. The population size is 8, and the evolution stopped after 30 generations. The assembly configuration in the target generation that has the highest fitness value is chosen as the optimal one (Fig. 13a). The average and maximum fitness values in every generation are shown in Figure

13(b). As can be seen, the evolutionary algorithm does increase the fitness values generation by generation. Although the best solution may not be guaranteed, a suboptimal solution can always be found, and in return, the efficiency of finding the solution is increased.
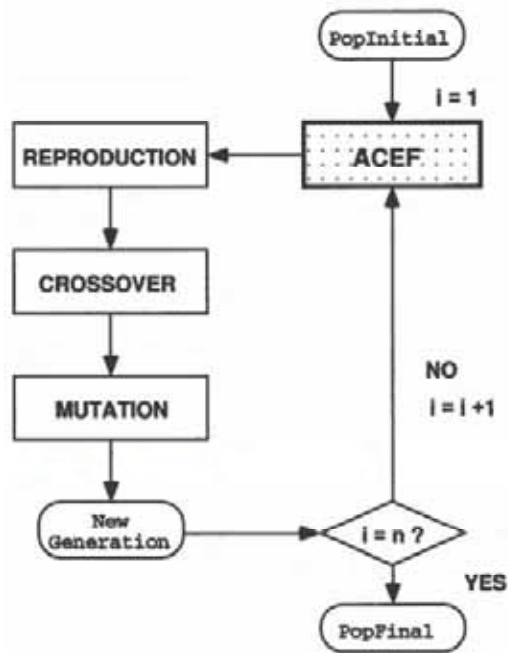
Figure 11. The evolution algorithm

Figure 12. The initial generation

(a)

(b)
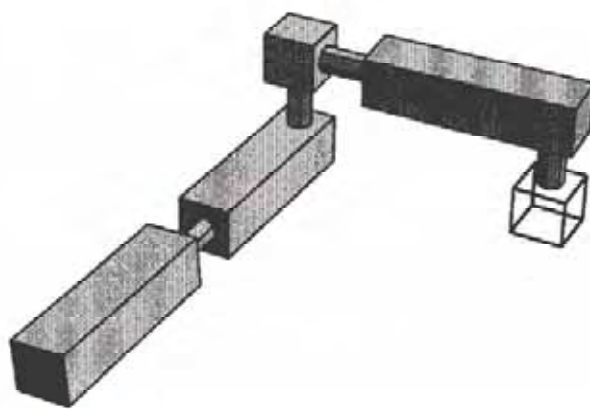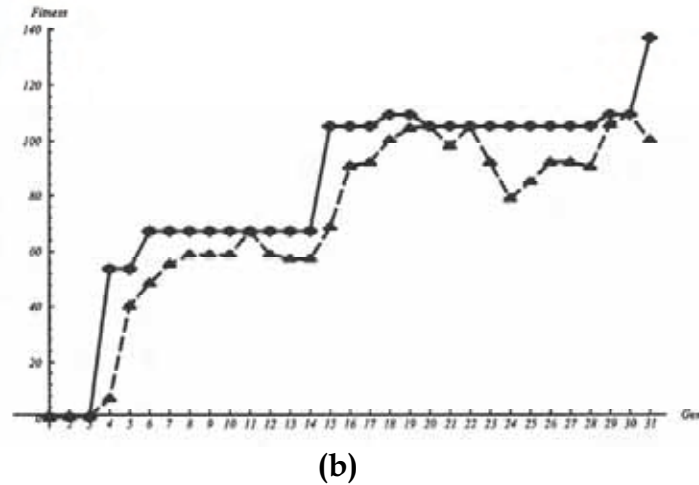
Figure 13. (a) Optimal assembly configuration; (b) average and maximum fitness in each generation

6. Simulation Software for Modular Robots

To visualize and simulate the performance of an assembled robot, such as reachability and workspace, a robot simulation software application is necessary. The **S**imulation **E**nvironment for **MO**dular **R**obot **S**ystem (a.k.a. *SEMORS*) is a Windows NT-based object-oriented software application developed for this purpose. Based on the proposed local POE models and AIM data structures, *SEMORS* offers uniform and automatic model construction effort (kinematics, dynamics and calibration) across computer simulation and real-time control of arbitrary robot configurations (Chen et al. 1999). The basic graphical user interface of *SEMORS* is illustrated in Figure 14. *SEMORS* is intended to be a uniform interface for all modular robots and is portable to modular robot systems from different vendors. It will be used both for simulation and for on-line execution of a task, regardless of whether the robot is executing (or is simulated to be executing) the task as a stand-alone application, or as part of a workcell process. Thus, it allows the user to quickly integrate the hardware components into modular robots, and to manage their operations in the reconfigurable workcell. Key features of *SEMORS* include:

- Module and robot builder
- D graphical task simulation
- "Universal" inverse kinematics
- Full dynamics models
- Trajectory and task planning
- Transparent workcell network connectivity
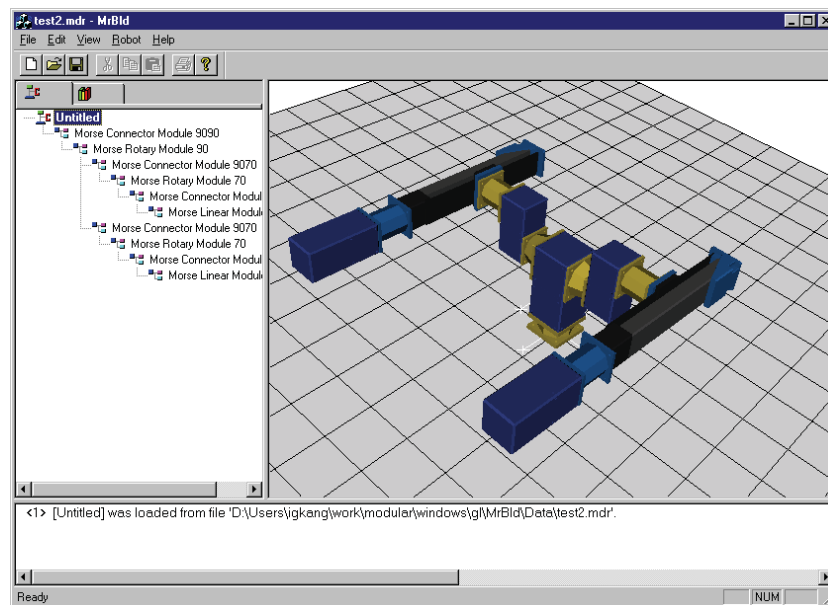
Figure 14. User interface of SEMORS

In addition to the simulation of modular robots, extended features like robot configuration planning/optimization and module database management are implemented as separate application packages to be used along with *SEMORS*. The task-based robot configuration optimization mentioned in Section 5 is a generic and platform-independent methodology. With the capability of task-based robot configuration optimization, designing the modular robot configuration using *SEMORS* becomes no longer an ad hoc approach. The software system will provide end-user an optimized robot configuration according to the input task requirements. The user does not need to start the design work from scratch. Rather, based on the result of optimization, he can fine-tune the suggested robot design or layout. The development effort and time for the workcell can be greatly reduced.

7. Prototype of Reconfigurable Robotic Workcell

To effectively demonstrate the use of a modular reconfigurable robotic system, we have constructed a prototype workcell for light-machining tasks in an industrial exhibition in 1999 (Figure 15). This workcell was built with multiple reconfigurable robots along with other supporting devices under a unified modular approach.

Figure 15. A light machining workcell with modular robot components

- Preliminary design stage

To make use of the advantages of both parallel-typed and serial-typed robots, we intend to make the workcell to perform a complete milling operation of a workpiece, starting from picking up the object, transferring the object to a milling robot, starting the milling process, and returning the workpiece back to a storage rack. Based on this preliminary concept, we decide to use two reconfigurable robots in this workcell: one is a serial-typed robot for the pick-and-place operation, and the other is a parallel-typed robot for the milling operation because of its structural rigidity. The task is to perform milling operation on a dome-shaped top of a cylindrical workpiece with 15cm in diameter. A workpiece transfer system should be used in between the two robots.

- Robot configuration selection and construction

Based on the preliminary task description, the workcell is configured with a 7-DOF redundant serial-type robot, a 6-DOF articulate RRRS parallel robot, and a 1-DOF linear motion stage. From the robot configuration optimization, a 4-DOF SCARA-type robot is sufficient to perform the task. Deploying a redundant robot here is to demonstrate that the proposed model generation algorithms used in *SEMORS* and in robot control are universally applicable for any configuration.

The configuration design of the parallel robot follows a systematic approach (Yang et al. 1999). In principle, a 3-branch parallel structure is used because of the structure stiffness and dexterity. Each branch consists of three rotary joints

(two are active and one is passive) and a passive spherical joint. Once the geometry is determined, the workspace analysis is performed. From the result of this analysis, the lengths of the rigid links and connectors are determined. Because of the modular design, the actuator modules can be freely located at the nine revolute joints. The workspace of the robot changes according to the locations of the actuator modules. A disk-shaped moving platform is attached to the three branches. An end-mill tool actuated by an intelligent motor is mounted at the center of the platform. This motor uses the same control interface as the standard actuator modules. Because of the lack of the force sensor, the task is only carried out in simulated manner, i.e., the end-mill tool only goes through the milling path without touching the surface of the workpiece.

The 1-DOF linear motion stage uses two standard modules: one rotary module to drive the linear slide and one gripper module to hold the workpiece, to ensure uniformity in the workcell control. The specifications of the robots and the motion stage are listed in Table 1.

• Workcell construction and fine-tuning

After the robots and motion stage are constructed, the robot controllers are connected to the robots. Two Pentium II-based industrial PC robot controllers are used to perform high-level trajectory control of the serial robot and the parallel robot respectively. The kinematic models of both serial and parallel robots are generated automatically in *SEMORS* and stored in the robot controllers. Kinematic calibration of both robots is performed before the operation. The kinematic calibration is conducted by using articulate-typed coordinate measuring equipment, called "Spin Arm". The obtained calibration data is transferred to the robot controller and then *SEMORS* computes and updates the corrected kinematic models of the robots automatically. Because of its simplicity, the control of the motion stage is done by one of the robot controller for this implementation.

• Finalize task sequence and control of the workcell actions

With updated kinematic models, the detailed task sequence of all robots (Table 2) is laid out. The tasks are then programmed into the respective robot controllers. The two robot controllers are connected to a closed-loop workcell LAN running at 10MB/sec. A separate notebook computer is also connected to the workcell network performing supervisory control of the workcell through *SEMORS* running on the individual robot controllers. The task sequence of the workcell is monitored and supervised by the notebook supervisor.

Based on the actual construction, to assembly the described 7-DOF serial-typed robot takes two users about 30 minutes. The time to construct the parallel robot requires two persons about two hours because of the complexity of the structure. Adding the time to install the motion stage, calibrate the robots and fine-tune the workcell hardware, it will take about four hours in total to com-

plete the entire workcell set-up excluding the time spent on the preliminary design stage.

Light-machining Workcell	
7-DOF Redundant Serial Robot	
Work envelope	Approx. sphere, SR = 1200mm
Max speed	750 mm/s
Repeatability	+/- 0.10 mm
Max Payload	5 Kg (excluding end-effector)
Weight	16 Kg (excluding base)
6-DOF RRRS Articulate Parallel Robot	
Work envelope	Approx. hemisphere, SR = 500mm
Max speed	500 mm/s
Repeatability	+/- 0.05mm
Max Payload	25 Kg (excluding end-effector)
Weight	30 Kg (excluding base)
1-DOF Linear Motion Stage	
Effective stroke	L = 1500mm
Max speed	500 mm/s
Repeatability	+/- 0.025mm
Max Payload	45 Kg (excluding fixture)
Weight	35 Kg

Table 1. Specifications of the light-machining workcell

8. Conclusion

We have presented a generic method to automate the model generation for modular reconfigurable robots based on a graph representation of robot geometry, called an assembly incidence matrix, and geometry-independent model building algorithms for the kinematics, dynamics and error models of a robot. The AIMs of the assembly configuration of modular robots facilitate the determination of optimal robot configuration for a specific task using combinatorial optimization techniques. We also presented here an approach to solve the task optimal problem using evolutionary algorithms with customized genetic operators based on the AIM of the robot. The application of this automatic modeling technique is implemented in a modular robot control and simulation software application, *SEMORS* (Simulation Environment for MOdular Robot Systems). In this software system, it is not necessary to maintain a library of robot models, since the possible assembly configurations of a robot is not a fixed number. Instead, only a small set of component database and kernel functions are kept in the robot controller, and required robot models are generated automatically. From the prototype construction, we can con-

firm the advantage of using modular components in constructing the complex robotic workcell with different configurations. The plug-and-play kinematics, dynamics, and calibration robot models are also verified through the actual implementation in the robot controller and the simulation software.

Acknowledgment:

Task Sequence	
1	Robot A picks up workpiece from fixture
2	Robot A places workpiece on the motion stage
3	Motion stage moves workpiece under Robot B
4	Robot B performs milling task
5	Robot A shifts locations of un-processed workpieces
6	Robot B finishes milling task
7	Motion stage moves processed workpiece back
8	Robot A picks up processed workpiece from motion stage
9	Robot A places processed workpiece to the fixture

*Robot A: 7-DOF serial robot, Robot B: 6-DOF parallel robot

Table 2. Task Sequence

9. References

Ambrose, R. O. (1995). Interactive robot joint design, analysis and prototyping, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 2119-2124, Washington DC, USA.

Benhabib, B.; Zak, G. & and Lipton, M. G. (1989). A generalized kinematic modeling method for modular robots, *Journal of Robotics Systems*, Vol. 60, No. 5, pp. 545-571.

Chen, I.-M. (1994). Theory and Applications of Modular Reconfigurable Robotic Systems, PhD thesis, California Institute of Technology, Division of Engineering and Applied Science, U. S. A.

Chen, I.-M. (1996). On optimal configuration of modular reconfigurable robots, *Proc. Int. Conf. Control, Automation, Robotics, and Vision*, pp. 1855-1859, Singapore.

Chen, I.-M. (2000). Realization of a rapidly reconfigurable robotic workcell, *Journal of Japan Society of Precision Engineering*, Vol. 66, No. 7, pp. 1024-1030.

Chen, I.-M. (2001). Rapid response manufacturing through reconfigurable robotic workcells, *Journal of Robotics and Computer Integrated Manufacturing*, Vol. 17, No. 3, pp. 199-213.

Chen, I.-M. & Burdick, J. W. (1995). Determining task optimal modular robot assembly configurations. *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 132-137, Nagoya, Japan.

Chen, I.-M. & Burdick, J. W. (1998). Enumerating Non-Isomorphic Assembly Configurations of a Modular Robotic System, *International Journal of Robotics Research*, Vol. 17. No. 7, pp. 702-719.

Chen, I.-M. & Yang, G. (1996). Configuration independent kinematics for modular robots. *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1845-1849, Minneapolis, MN, USA.

Chen, I.-M. & Yang, G. (1997). Kinematic calibration of modular reconfigurable robots using product-of-exponentials formula, *Journal of Robotic Systems*, Vol. 14, No. 11, pp. 807-821.

Chen, I.-M. & Yang, G. (1999). Numerical inverse kinematics for modular re-configurable robots, *Journal of Robotics Systems*, Vol. 16, No. 4, pp. 213-225.

Chen, I.-M.; Yang, G.; Yeo, S. H. & Chen, G. (1999). Kernel for modular robot applications - automatic modeling techniques, *International Journal of Robotics Research*, Vol. 18, No. 2, pp. 225-242.

Chen, I.-M.; Tan, C. T.; Yang, G. & Yeo, S. H. (2001). A local POE model for ro-bot kinematic calibration, *Mechanism and Machine Theory*, Vol. 36, No. 11, pp. 1215-1239.

Cohen, R.; Lipton, M. G.; Dai, M. Q. & Benhabib, B. (1992). Conceptual design of a modular robot, *ASME Journal Mechanical Design*, Vol. 114, pp. 117-125.

Cormen, T.; Leiserson, C. & Rivest, R. (1990). *Introduction to Algorithms*, ISBN 0262032937, MIT Press, Cambridge, MA, USA.

Deo, N. (1974). *Graph Theory with Applications to Engineering and Computer Science*, ISBN 0133634736, Prentice-Hall, New York, USA.

Dobrjanskyj, L. & Freudenstein, F. (1967). Some applications of graph theory to the structural analysis of mechanisms, *ASME Journal Engineering for Industry*, Vol. 89, pp. 153-158.

Featherstone, R. (1987). *Robot Dynamics Algorithms*, ISBN 0898382300, Kluwer Academic Publishers, Holland.

Fukuda, T. & Nakagawa, S. (1988). Dynamically reconfigurable robotic system, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1581-1586, Philadelphia, PA, USA.

Hollerbach, J. M. (1980). A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity, *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 10, pp. 730-736.

Kelmar, L. & Khosla, P. (1988). Automatic generation of kinematics for a reconfigurable modular manipulator system, *Prof. IEEE Int. Conf. Robotics and Automation*, pp. 663-668, Philadelphia, PA, USA.

Khosla, P. K.; Neuman, C. & Prinz, F. (1985). An algorithm for seam tracking applications, *International Journal of Robotics Research*, Vol. 40, No. 1, pp. 27-41.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*, 2ed., ISBN 540580905, Springer-Verlag, Berlin, Germany.

Murray, R.; Li, Z. & Sastry, S. (1994) *A Mathematical Introduction to Robotic Manipulation*, ISBN 0849379814, CRC Press, Boca Raton, FL, USA.

Paredis, C. J. J.; Brown, H. B. & Khosla, P. (1997). A rapidly deployable manipulator system, *Robotics and Autonomous Systems,* Vol. 21, No. 3, pp. 289-304.

Paredis, C. J. J. & Khosla, P. K. (1995). Design of modular fault tolerant manipulators, In: *Algorithmic Foundations of Robotics*, Goldberg, K. (ed.), pp. 371-383, A. K. Peters, ISBN 1568810458, Wellesley, MA, USA.

Park, F. C. & Bobrow, J. E. (1994). A recursive algorithm for robot dynamics using lie groups, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1535-1540, San Diego, CA, USA.

Park, F. C.; Bobrow, J. E. & Ploen, S. R. (1995). A lie group formulation of robot dynamics, *International Journal of Robotics Research*, Vol. 14, No. 6, pp. 609-618.

Rodriguze, G.; Jain, A. & Kreutz-Delgado, K. (1991). A spatial operator algebra for manipulator modeling and control, *International Journal of Robotics Research*, Vol. 10, No. 4, pp. 371-381.

Schmitz, D.; Khosla, P. K. & Kanade, T. (1988). *The CMU reconfigurable modular manipulator system*, Technical Report CMU-RI-TR-88-7, Robotics Institute, Carnegie Mellon University.

Wurst, K. H. (1986). The conception and construction of a modular robot system. *Proc. 16th Int. Sym. Industrial Robotics (ISIR)*, pp. 37-44, Brussels, Belgium.

Yang, G. & Chen, I.-M. (2000). Task-based optimization of modular robot configurations – MDOF approach, *Mechanism and Machine Theory*, Vol. 35, No. 4, pp. 517-540.

Yang, G.; Chen, I.-M.; Lim, W. K. & Yeo, S.H. (1999). Design and kinematic analysis of modular reconfigurable parallel robots, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 2501-2506, Detroit, MI, USA.

Kinematic Design and Description of Industrial Robotic Chains

Peter Mitrouchev

1. Introduction

Today, industrial robots can replace humans in carrying out various types of operations. They can as well serve machine tools as to carry out various tasks like welding, handling, painting, assembling, dismantling, foundering, forging, packaging, palletizing ….in different areas of the mechanical, car, aerospace, automotive, electronics … and other industries. However, the complexity of the industrial process poses difficult problems for insertion and generation of the movements of a robot: the working environment of the robot is often complex and varied (presence of obstacles during the execution of a task for example).

One of the objectives concerning the problems of computer-aided design (CAD) of robots is the validation of their topological structures. The robot-design engineer puts forth assumptions as regards the provision of the links and the joints of the mechanical articulated system. A first validation of this choice is of a geometrical nature (Merlet, 1996). At first sight the design of a mechanical architecture for a robot appears rather simple and yet it presents a very complex basic difficulty, as it must take into account not only the mechanical possibilities of realization but also the possibilities of control's development, which passes by generating of a mathematical model. The latter strongly affects the mechanical design if good performances are sought. Many methods for mechanism description appeared with the creation of CAD systems (Warnecke, 1977) and (Coiffet, 1992). The existing methods may be separated into two categories:

- description methods for classification (Roth, 1976),
- methods for mathematical modelling (Borel, 1979), (Khalil, 1976),
  (Renaud, 1975) and (Touron, 1984).

Mechanisms and Machines Theory (MMT) contributed greatly to planar and spatial mechanism synthesis with different degrees of freedom (Hwang & Hwang, 1992; Hervè, 1994; Gonzales, 1996; Karouia & Hervè, 2005). Some of

the current industrial robots with planar chains have a structure created by the kinematic graphs of MMT (Manulescu et al. 1987; Ma & Angeles, 1991).

The morphological (topological) synthesis of kinematic chains has, for a long time, been the subject of many papers. There are different methods for number synthesis of planar kinematic chains with simple revolute joints, with different degrees of mobility and different numbers of links and joints. These chains are usually called "planar pin-joined" chains in MMT. While number synthesis originates from kinematics of mechanisms, all methods entail operations on graphs, which in one way or another, represent kinematic chains.

There exist different methods for kinematic synthesis of planar chains with simple joints (Tischler et al., 1995; Belfiore, 2000; Rao & Deshmukh, 2001): intuition and inspection (Crossley, 1964), graphs theory (Dobrjanskyi and Freudenstein, 1967; Woo, 1967). Others consist in transformation of binary chains (Mruthyunjaya, 1979; Mruthyunjaya, 1984-a; Mruthyunjaya, 1984-b; Mruthyunjaya, 1984-c), in the concept of Assur groups (Manolescu et al., 1987; Manolescu, 1964; Manolescu, 1979; Manolescu, 1987), or Franke's notation (Davies & Crossley, 1966; Crossley, 1966). Recently, new methods based on genetic algorithms or neuronal networks are also used (Tejomurtula & Kak, 1999; Abo-Hamour et al., 2002; Cabrera et al., 2002; Laribi et al., 2004).

The analysis of existing methods shows that there are several methods applied to the development of a mathematical model concerning its application for the control design of the robot. However, concerning the topological description of the chains and robots, only Roth-Piper's method (Roth, 1976; Pieper & Roth, 1969) tends towards mechanism description with a view to classify robots.

Generally speaking, the problem of synthesis of mechanism falls into three sub problems:

- specification of the problem: topological and functional specifications and constraints imposed by the environment,
- topological synthesis of the mechanism: enumeration and evaluation of possible topologies,
- dimensional synthesis: choice of dimensions of the mechanism for the selected types of morphologies.

This chapter relates in particular to the second sub problem. Its *principal goal* is to present an overview concerning the chronology of the design of an industrial robot kinematic chain. The chapter starts with a brief reminder of the theory of Modular Structural Groups (MSG), and of the *connectivity* and *mobility* laws of MMT presented in § 2. Afterwards, a new method for structural synthesis of planar link chains in robotics is presented in § 3. It is based on the notion of *logical equations*. Various levels of abstraction are studied concerning the complexity of the structure. This permits the synthesis of planar chains with various degrees of complexity expressed by the number of links, joints and the

degree of mobility. The logical equations allow the association of MSGs of type A and closed chains of type G. The rules for associations of groups are also presented. The aim is to execute all the possible combinations to *join* or *transform* links in order to obtain as many structures as possible by avoiding those which are isomorphic. The association of two groups allows the elaboration of different closed chains of upper level. However there are some defective structures, which do not respect the connectivity and mobility laws. Therefore great care has been taken to avoid them. The problem of degenerated structures is central in their synthesis. It especially concerns chains with two and more degrees of mobility. The problem of *defect*, *degeneration* and *isomorphism* is then approached. Later, a method for description of chains by *contours* and *molecules* is proposed in § 4. It enables to reduce the number of the topological structures of the robots concerning its frame and end-effector position by comparing their respective molecules. This method is then applied in § 5 to describe the structures thus obtained (by logical equations approach) and the topology of the *principal structure* of the industrial robots in the initial phase of their design. Finally a classification of industrial robot structures by different levels of complexity is presented in § 6.

2. Topology of a linked mechanical structure, definitions, terminologies and restrictions

The Manipulation System (MS) of the robot is a mechanism composed of links (elements) joined by kinematic joints often with one degree of mobility (rotary or prismatic joints). These elements are considered as rigid bodies which form a kinematic chain, plane or spatial, which can be open or closed. The type of a kinematic link noted "j" is given by the number of joints enabling its association with other links (Erdman & Sandor, 1991). There are links of "binary", "ternary", "quaternary"... "polynary" type with j=1,2,3..., following from the fact that the link contains 1,2,3,... kinematic joints :
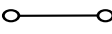
link	binary	ternary	quaternary	...
representation				...
notation	N_2	N_3	N_4	...

Table 1. Planar link types

MMT proposes various ways of representing kinematic structures. The most common, the *kinematic graph*, consists in conserving a shape for the links in or-

der to better appraise the topology of the structure. Nevertheless this presentation is difficult to manipulate. Any kinematic structure may be transformed into Crossley's *inverse graph* (Crossley, 1964) replacing every link (binary, ternary…) by a point. Lines linking the points concerned represent the joints themselves.
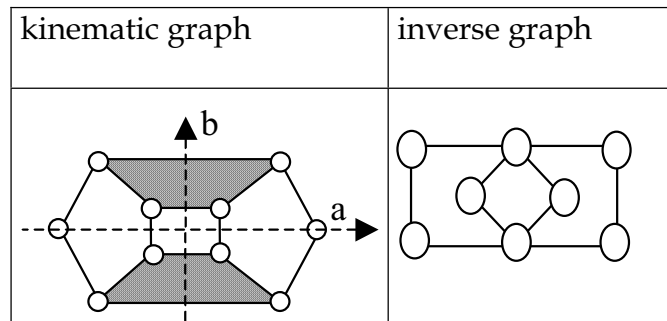
Figure 1. Representation of a structure by kinematic and Crossley's inverse graph

Kinematic *chain* (named Grübler chain in MMT) is a structure which "floats" in space and all its links have identical status. If a frame and an input link (links) is (are) indicated the chain becomes a *mechanism*. The possible motions in a chain are called *degree of freedom* (or absolute degree of mobility) whereas for a mechanism they are called *degree of mobility* (or relative degree of mobility) of N-link Grübler chain.

As it has been said, the arm of a robot or the MS is a mechanism composed of a set of rigid links, which form an open or closed kinematic chain. Existing robots may be topologically classified in two categories according to whether their structure is open or closed:

- robots with simple (open) structure: one can traverse all the kinematic joints by making an open chain. They are noted $A_{i,j}$ where i and j are respectively the degree of mobility and the number of links of the structure (cf. § 3.1).
- robots with complex (arborescent or closed) structure: one can traverse all the kinematic joints by making a closed loop. They are noted $G_{k,l}$ where k and l are respectively the degree of mobility and the number of links of the structure (cf. § 3.1).

Another method allows them to be classified mechanically as robots with planar or spatial chains.
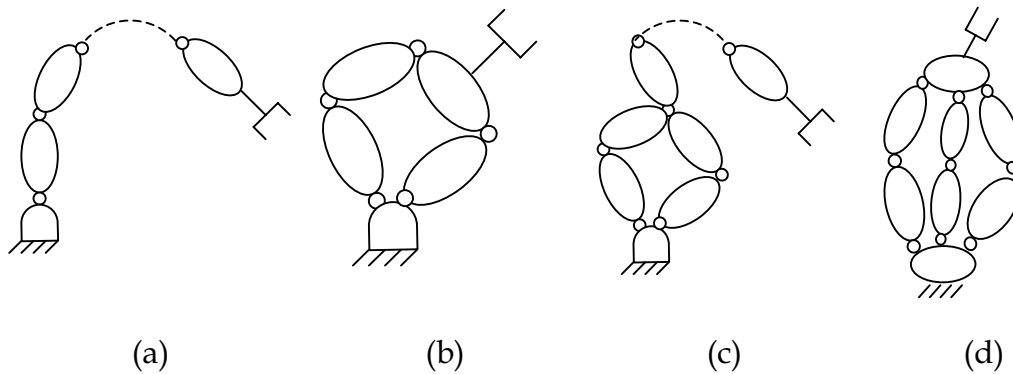
Figure 2. Robots with: simple-open (a), closed (b), arborescent (c) and complex structures (d)

Robots, being complex mechanical systems with an important interaction between their links, their architecture is defined by (Mitrouchev, 1999):

- the *main structure* which generates the main motion of the robot and upon which is situated the rest of the MS,
- the *regional structure* which is composed of the arm and of the forearm (mechanical arm),
- the *local structure* which is the wrist of the robot often with three degrees of freedom in rotation.
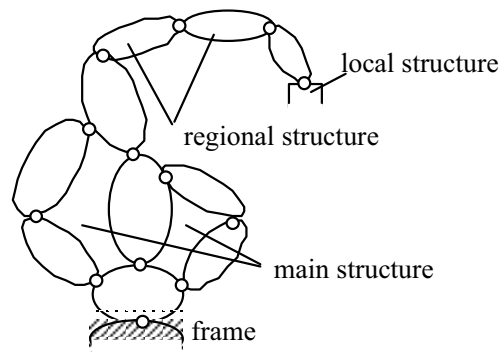
Figure 3. Topological structure of a robot

The number M of degrees of mobility of an isostatic chain relative to a fixed frame is given by Somov-Malisev's formula (Artobolevski, 1977):

$$M = 6N^* - \sum_{k=1}^{5} kC_k \tag{1}$$

with: C_k – number of simple joints of class "k",
N^* - number of mobile links (in general $N^* = N-1$).

If the kinematic chain only contains simple rotary joints of class 5 (one degree of freedom joint), instead of equation (1), the Tchebychev-Grübler equation is used:

$$M = 3N^* - 2C_5 \qquad (2)$$

with: C_5 - number of simple rotary joints with one degree of mobility.

In this chapter only planar structures with simple rotary joints of class 5 shall be considered. If the value of M in Tchebychev-Grübler's equation (2) is M=0 it becomes:

$$3N^* = 2C_5 \qquad (3)$$

Some of the most characteristic MSGs (called Assur groups) resulting from equation (3) are presented in Table 2 by their kinematic graphs or structural diagrams (Manolescu, 1964).

notation	N*	C5	Assur groups	derivative groups
$A_{0,2}$	2	3		....
$A_{0,4}$	4	6		
$A_{0,6}$ ...	6 ...	9 ...	....	....

Table 2. Modular Structural Groups (MSGs) with zero degree of mobility

If the value of M in Tchebychev-Grübler's equation (2) is M=1 it becomes:

$$3N^* - 1 = 2C_5 \qquad (4)$$

Some of its solutions are presented in table 3.
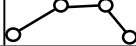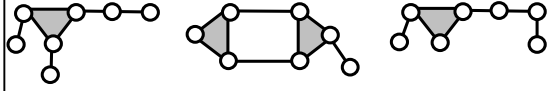
notation	N*	C5	kinematic graphe (structural diagram)
$A_{1,3}$	3	4	
$A_{1,5}$ ...	5 ...	7 ...	....

Table 3. SMGs with one degree of mobility

Finally for the MSGs adding two degrees of mobility to the structures (M=2), one obtains:

$$3N^* - 2 = 2C_5 \tag{5}$$

Some of its solutions are presented in table 4:
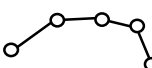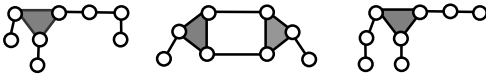
notation	N*	C5	structural diagram
$A_{2,4}$	4	5	
$A_{2,6}$ ...	6 ...	8 ...	

Table 4. MSGs with two degrees of mobility

3. Proposed method and results

Let us consider a plane kinematic structure comprising N links and C_5 kinematic joints in open, arborescent or closed chain. The first link S_1 is the presumed fixed frame of the robot. MMT, being part of the technological sciences, is at the base of mechanism design in robotics as has been previously mentioned. The question is: for a given degree of mobility and for a given number of links and joints, how many possibilities are there to join them in a mechanism suitable for application in kinematic chain design in robotics? The answer to this question is presented in this paragraph and in the following one. In order to answer the question above, let us consider a mechanism with a given degree of mobility (M). Gauhmann's generalised law (called *connectivity* law) giving the relationship between the number of the links and the joints is (Manolescu, 1987):

$$C_5 = I + N \tag{6}$$

with: I=R-2 -structural number representing the complexity level of
       the structure,

      R   - number of closed loops in the chain allowing to "read" (to pass
       through) twice each kinematic joint.

For the planar kinematic structures the *general law of mobility* giving the relationship between the number of degrees of mobility and the joints is:

$$C_5 = M + 3I + 3 \tag{7}$$

The equations (6) and (7) give:

$$N = M + 2I + 3 \tag{8}$$

Those two latter equations (7 and 8) lead to the *first manner* of calculating the number of joints C_5 and the number of links N for planar kinematic structures with different degrees of relative mobility M. The *general law of mobility* giving the relationship between the number of degrees of mobility and the joints is (Manolescu, 1987):

$$M + (1+I)H = \sum_{k=f+1}^{5} (6-k)C_k \tag{9}$$

with: - H=(6-F),
       - F the number of imposed constraints in a chain.

For a planar kinematic structures, F=3 hence H=3. Equation (9) becomes:

$$M + 3(1+I) = C_5 + 2C_4 \tag{10}$$

This latter equation allows the *second manner* of calculating the number of links and joints for planar kinematic structures with different degrees of mobility.

3.1 Logical equation

3.1.1 Notations

We note by:

- $A_{i,j}$ a MSG (Assur group or its derivative) which may be open or closed. There are always some exterior (free) joints which only become active when the group is in connection with a joint shared with a chain or a mechanism (Manolescu, 1964),
- $G_{k,l}$ a closed group (structure) without exterior joints.

The first mark represents the degree of mobility of the group and the second one represents the number of its links (cf. Table 5).
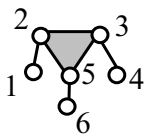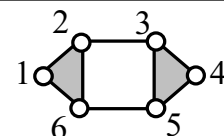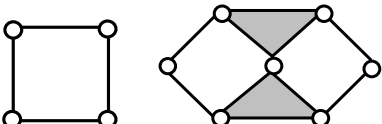
groupes	structural diagram	joints	
		exterior	interior
Ai,j		1,4,6	2,3,5
		1,4	2,3,5,6
Gk,l		interior only	

Table 5. Structural diagram and types of joints

A *logical equation* is defined as the association (combination) of two initial terms (marks in lower case letters). The result of this association is a final closed kinematic structure of type $G_{M,N}$ (marks in capital letters) of upper level. The two initial terms may be (Mitrouchev, 2001):

1.   Two identical MSG groups, for example: $A_{i,j} + A_{i,j} = G_{M,N}$

with: M=2i+1;   for M=1 and I=0,2
                                 for M=3 and I=0
           N=2j.

2.   Two different MSG groups, for example: $A_{i,j} + A_{i',j'} = G_{M,N}$

with: M=i+i'+1;   for M=1, and I=1,2,3
                                for M=2, and I=0
                N=j+j'.

3.   One MSG group and one closed group of type $G_{k,l}$: $A_{i,j} + G_{k,l} = G_{M,N}$

with: M=i+k;   for M=1, and I=1,2,3
                                for M=2,3 and I=1,2
                N=j+l

It can be noted that the association of two closed structures is not possible.

3.1.2 Rules for associations

The association of two groups may be done by two operations: direct *junction* and *transformation* of links. They may be executed separately or together. The aim is to use all possible combinations to join or transform the links in order to obtain as many structures as possible (for given N and C_5) by avoiding those, which are isomorphic. Some examples are shown in the Table 6:
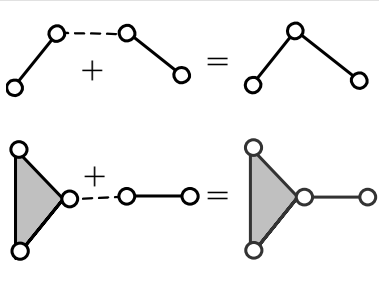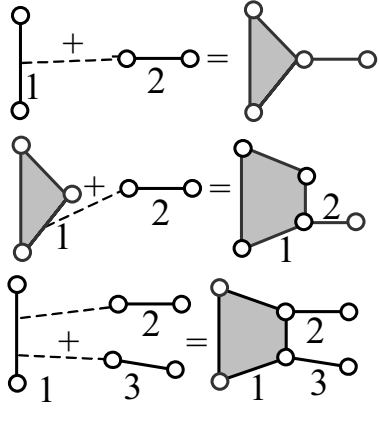
Table 6. Rules for associations of groups

3.2 Generation of chains with one relative degree of mobility (M=1)

Fof M=1 the equations (7) and (8) give:

$$C_5 = 4 + 3I \tag{11}$$

and
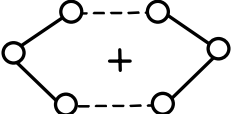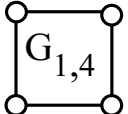
$$N = 4 + 2I \tag{12}$$

3.2.1 First level of abstraction, (I=0)

For I=0, equations (11) and (12) give C_5=4 and N=4. Therefore only one closed planar structure of type $G_{1,4}$ can be synthesised from the following logical equation:

$$A_{0,2} + A_{0,2} = G_{1,4} \tag{13}$$

as follows:

logical equation	schematic	closed structure
$A_{0,2} + A_{0,2} = G_{1,4}$		

The equation (6) gives C_5=N. For C_5=N=4 there is a mono-contour mobile structure called a group of type G4. For M=1, the equation (10) gives:

$$C_5 + 2C_4 = 4 \tag{14}$$

The possible solutions of this equation are: a) C_4=0, C_5=4, b) C_4=1, C_5=2 and c) C_4=2, C_5=0.

From the solution a) only one closed planar structure of type G4 and noted $G_{1,4}$ can be synthesised by the logical equation (13) above.

3.2.2 Second level of abstraction, (I=1)

For I=1, equations (11) and (12) give C_5=7 and N=6. The law for generation a group of elementary modules allows the calculation of the number of links, which are ternary and more (quaternary, quintary and so on) in a closed chain (Manolescu et al, 1972).

$$2I = \sum_{j=3}^{R}(j-2)N_j \tag{15}$$

For I=1 equation (15) gives N_3=2. From equation (10) one deduces:

$$7 = C_5 + 2C_4 \tag{16}$$

The possible solutions of this equation are: a) C_4=0, C_5=7; b) C_4=1, C_5=5; c) C_4=2, C_5=3 and d) C_4=3, C_5=1. The number of the links is calculated by equation (6). It becomes:

$$N = C_5 - I = 6 \tag{17}$$

Consequently the number of binary links is:

$$N_2 = N - N_3 \tag{18}$$

The second level planar structures, of type $G_{1,6}$ are obtained from the following logical equations:

$$G_{1,4} + A_{0,2} = G_{1,6} \tag{19}$$

$$A_{0,4} + A_{0,2} = G_{1,6} \tag{20}$$

Both equations (19) and (20) give two solutions: Watt's and Stephenson's structures. These latter are described by their kinematic graphs (structural diagram) in table 7.
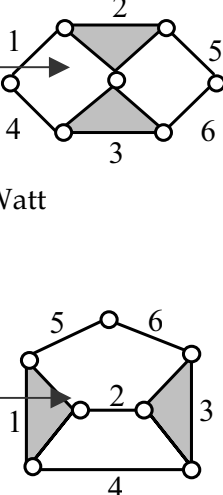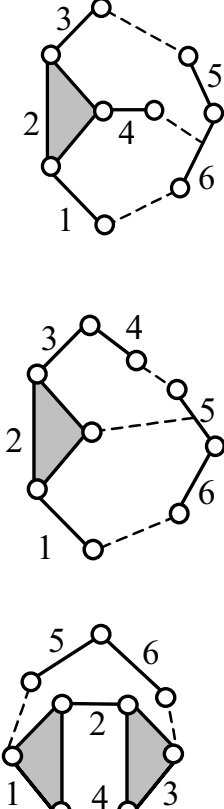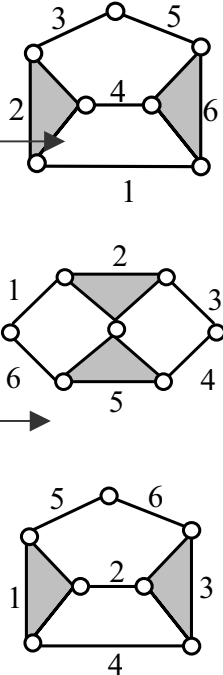
logical equation	groups	method for association	structural diagram
$G_{1,4} + A_{0,2} = G_{1,6}$		by transformation of links	Watt Stephenson
$A_{0,4} + A_{0,2} = G_{1,6}$		by junction and transformation of links	
		by junction and transformation of links	
		by junction	

Table 7. Watt's and Stephenson's structures

It may be noted that there are three isomorphic structures, which restrict their number to two. Watt's and Stephenson's structures are generated from *closed immobile structure* composed of two ternary links by adding to it four binary links as presented in figure 4.
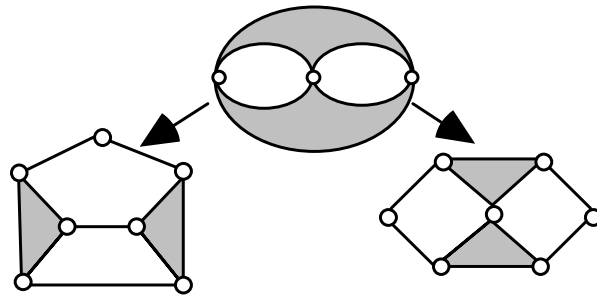
Figure 4. Closed immobile structure (collapse) and its derivatives

The closed immobile structures have no degree of mobility. These are hyperstatic systems, thus they are invariant concerning the mobility parameter. The theory of closed structures is applied just as much in robotics as in civil engineering, concerning bridges and roadways, seismic structures etc. They are equally applicable for planar structures as for spatial ones.

3.2.3 Third level of abstraction, (I=2)

According to expressions (11) and (12) for I=2 one obtains C_5=10 and N=8 in other words the structure has 8 links and 10 joints. According to equation (15) the number of links is:

$$N_3 + 2N_4 = 4 \tag{21}$$

The possible solutions of this equation and the associated closed structures are given in table 8:

solution	a) N_3=4, N_4=0	b) N_3=2, N_4=1	c) N_3=0, N_4=2
associated closed structure			

Table 8. Third level closed structures

For the case of planar structure, the number of the binary links is:

$$N_2 = N_0 + \sum_{j=4}^{R=I+2} (j-3)N_j \tag{22}$$

where $N_0 = M+H$ is the number of links in the main Crossley's chain ($N_0 = 1+3 = 4$) hence:

$$N_2 = 4 + \sum_{j=4}^{R} (j-3)N_j \tag{23}$$

For I=0 and I=1 the equation (23) has no physical meaning. For $j \leq 3$, the number of binary links is $N_2 = 4$, so to have a mobile mechanism one needs four binary links, hence for j=4:

$$N_2 = 4 + N_4 \tag{24}$$

The possible solutions of this equation are: a) $N_2 = 4$, $N_4 = 0$, b) $N_2 = 5$, $N_4 = 1$, c) $N_2 = 6$, $N_4 = 2$ and d) $N_2 = 7$, $N_4 = 3$. The three solutions of (24) which respect the solutions of (21) are a), b) and c).

The third level planar structures of type $G_{1,8}$ are obtained from the following logical equations:

$$G_{1,6} + A_{0,2} = G_{1,8} \tag{25}$$

$$G_{1,4} + A_{0,4} = G_{1,8} \tag{26}$$

$$A_{0,4} + A_{0,4} = G_{1,8} \tag{27}$$

$$A_{0,2} + A_{0,6} = G_{1,8} \tag{28}$$

Equation (25) gives 12 structures; equation (26) gives two structures and a double (isomorphic); equation (27) gives one structure, one defect and one double. Finally, equation (28) gives one structure and two doubles. Table 14 (cf. § 4.) recapitulates the sixteen structures solutions of these four logical equations.

3.2.4 Problem of defect, degeneration and isomorphism

The problem of degenerated structures is central in synthesis of structures. It does not concern the chains with one degree of mobility (M=1). But if the chains have two, three or more degrees of mobility there are two degenerate forms of chains called *partial* and *fractionated* mobilities (Manolescu, 1964). If any link of the chain is selected to be the frame of a mechanism and if one chooses any M members to be the independent driving links (called "motor" links), then the position of every remaining member is dependent on the position of all links, such a mechanism is said to have a *total degree of mobility* (Davies & Crossley, 1966). This is the case of structures: 2 of Table 11, and 2, 3, 4 and 6 of Table 12. A mechanism may possess a *partial degree of mobility* if it cannot fulfil the conditions of total degree of mobility mechanism. In all or some of the derived mechanisms, the movement of certain of the driven links depends on the movements of only a number Mp of the motor links, where Mp<M (Manolescu, 1964). This is the case of structures: 1 and 3 of Table 11, and 1, of Table 12. Finally, a mechanism has a *fractionated degree of mobility* if it does not satisfy the conditions for a total degree of mobility. Such a mechanism possesses at least one link (with at least four joints), which may be decomposed into two or more parts, with the result that each sub-mechanism forms a closed chain. The mobility of the whole mechanism is equal to the sum of the mobilities of all sub-mechanisms (Davies & Crossley, 1966). This is the case of structures 4 of table 11; and 5 of table 12.

The rules for association of two groups allow the elaboration of different closed chains. But there are some defect structures, which do not respect the connectivity and mobility laws. Therefore great care has been taken to avoid them. For instance a defect structure is the one among the three solutions of the logical equation (27) shown in Table 9.
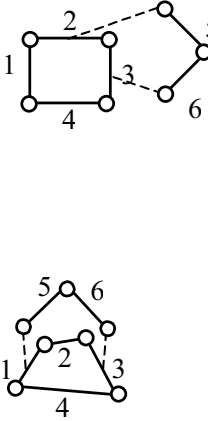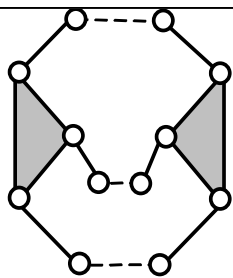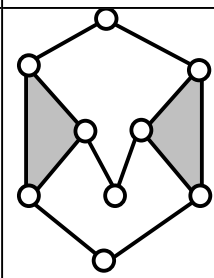
logical equation	groups	method for association	structural diagram
$A_{0,4} + A_{0,4} = G_{1,8}$		by junction	

Table 9. Defect structure

By junction of the six binary links, which is the simplest way, the resulting structure is a defect because it has I=9-8=1 and M=21-18=3. Nevertheless the

structural number I is equal to 2 and the degree of mobility M is equal to 1. Finally there are some isomorphic structures too. Consequently the defect structures and the isomorphic ones are systematically moved away from the possible solutions.

3.2.5 Fourth level of abstraction, I=3

According to expressions (11) and (12) for I=3, one obtains: C_5=13 and N=10. The number of links, which are ternary or greater, is calculated from equation (15). It becomes:

$$6 = N_3 + 2N_4 + 3N_5 \qquad (29)$$

The possible solutions of this equation and their associated closed structures are given in the table 10:
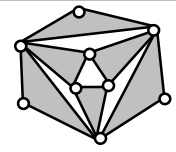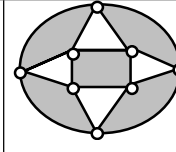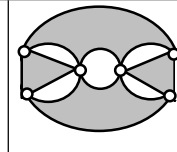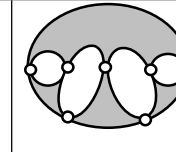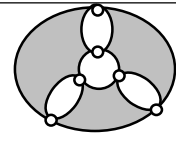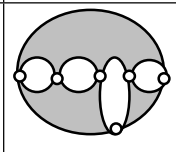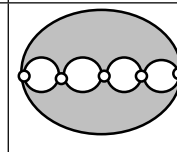
solution	a)N_3=6, N_4=0, N_5=0	b)N_3=4, N_4=1, N_5=0	c)N_3=2, N_4=2, N_5=0	d)N_3=3, N_4=0, N_5=1
associated closed structure				
solution	e)N_3=0, N_4=3, N_5=0	f)N_3=1, $_4$=1, N_5=1	g)N_3=0, N_4=0, N_5=2	
associated closed structure				

Table 10. Fourth level closed structures

The number of binary or greater links is calculated by equation (23):

$$N_2 = 4 + N_4 + 2N_5 \qquad (30)$$

The possible solutions of this equation are:

a) N_2=4, N_4=0, N_5=0, b) N_2=5, N_4=1, N_5=0, c) N_2=6, N_4=2, N_5=3, d) N_2=6, N_4=0, N_5=1, e) N_2=7, N_4=3, N_5=0, f) N_2=7, N_4=1, N_5=1, g) N_2=8, N_4=0, N_5=2 and h) N_2=8, N_4=4, N_5=0.

The solutions of (30), which respect the solutions of (29), are a), b), c), d), e) f) and g). Thus the fourth level planar structures of type $G_{1,10}$ are obtained from the following logical equations:

$$G_{1,8} + A_{0,2} = G_{1,10} \tag{31}$$

$$G_{1,6} + A_{0,4} = G_{1,10} \tag{32}$$

$$G_{1,4} + A_{0,6} = G_{1,10} \tag{33}$$

$$A_{0,2} + A_{0,8} = G_{1,10} \tag{34}$$

$$A_{0,6} + A_{0,4} = G_{1,10} \tag{35}$$

Equation (31) gives 50 structures, (32) gives 95 structures, (33) gives 57 structures, (34) gives 18 structures and (35) gives 10 structures. In total 230 structures obtained also by Woo (Woo, 1967) using the graph theory.

3.3 Generation of chains with two relative degrees of mobility (M=2)
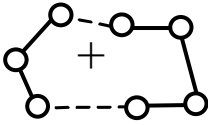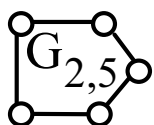
For M=2 the equations (7) and (8) give:

$$C_5 = 5 + 3I \tag{36}$$
And

$$N = 5 + 2I \tag{37}$$

For I=0 the solutions of equations (36) and (37) are C_5=5 and N=5. The only kinematic structure of type $G_{2,5}$ is provided by the following schematic:

logical equation	Schematic	closed structure
$A_{0,2} + A_{1,3} = G_{2,5}$		

For I=1, the expressions (36) and (37) give C_5=8 and N=7. The four second level planar structures, of type $G_{2,7}$ are shown in table 11:
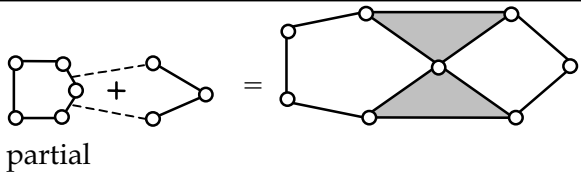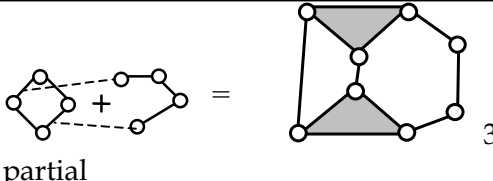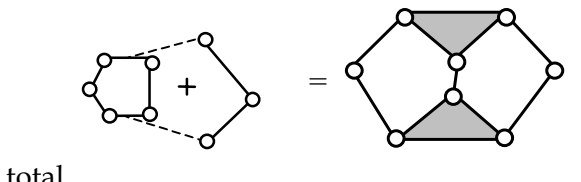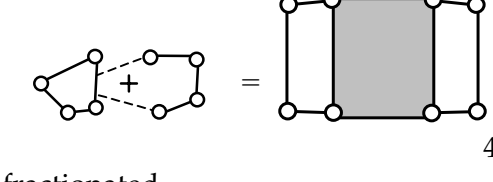
$G_{2,5} + A_{0,2} = G_{2,7}$	$G_{1,4} + A_{1,3} = G_{2,7}$
1 partial	3 partial
2 total	4 fractionated

Table 11. Planar structures of type $G_{2,7}$

For I=2, the solutions of equations (36) and (37) are respectively C_5=11 and N=9. The third level planar structures, of type $G_{2,9}$ are obtained from the following logical equations:

$$G_{2,7} + A_{0,2} = G_{2,9} \tag{38}$$

$$G_{2,5} + A_{0,4} = G_{2,9} \tag{39}$$
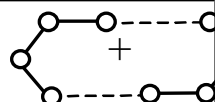
$$G_{1,6} + A_{1,3} = G_{2,9} \tag{40}$$

$$G_{1,4} + A_{1,5} = G_{2,9} \tag{41}$$

The forty structures produced by these four equations were also obtained by Manolescu (Manolescu, 1964) using the method of Assur groups.

3.4 Generation of chains with three relative degrees of mobility, (M=3)

For M=3 the equations (7) and (8) give: $C_5 = 6 + 3I$ and $N = 6 + 2I$. For I=0 one obtains C_5=6 and N=6. The only kinematic structure of type $G_{3,6}$ is provided by the following schematic:

logical equation	Schematic	closed struct.
$A_{1,3} + A_{1,3} = G_{3,6}$		

For I=1, one obtains C_5=9 and N=8. The six second level planar structures, of type $G_{3,8}$ are presented in Table 12:

$G_{3,6} + A_{0,2} = G_{3,8}$	$G_{2,5} + A_{1,3} = G_{3,8}$
 1 partial	 total 4
 2 total	 fractionated 5
 3 total	 Total 6

Table 12. Planar structures of type $G_{3,8}$.

One can notice that there are two isomorphic structures 2 and 6 that restrict their number to five.

For I=2, one obtains C_5=12 and N=10. The third level planar structures, of type $G_{3,10}$ are obtained from the following logical equations:

$$G_{3,8} + A_{0,2} = G_{3,10} \tag{42}$$

$$G_{3,6} + A_{0,4} = G_{3,10} \tag{43}$$

$$G_{2,7} + A_{1,3} = G_{3,10} \tag{44}$$

$$G_{2,5} + A_{1,5} = G_{3,10} \tag{45}$$

$$G_{1,6} + A_{2,4} = G_{3,10} \tag{46}$$

$$G_{1,4} + A_{2,6} = G_{3,10} \tag{47}$$

The ninety-seven structures produced from these six relations were also obtained by T.S. Mruthyunjaya (Mruthyunjaya, 1984-a; Mruthyunjaya, 1984-b; Mruthyunjaya, 1984-c) using the method of transformation of binary chains. These mechanisms with three degrees of mobility are often present in the design of industrial robots.

4. Description of planar structures

Apart from the obvious manner to describe a structure by its *kinematic graph*, the following two ways to describe the planar structures thus obtained are proposed.

1. By *contours*: a contour is defined by a closed loop allowing one to pass through the kinematic joints belonging to it. Two kinds of contours are distinguished: *internal* and *external*.

- An internal contour is defined by a closed loop allowing one to travel trough the kinematic joints belonging to the interior of a structure. A single circle represents it and is noted by a lower case Greek letter.
- An external contour is defined by a closed loop allowing one to travel through the kinematic joints belonging to the exterior of a structure. A double circle represents it and is noted by a capital Greek letter.
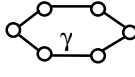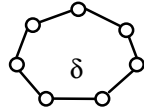
contour	α	β	γ	δ	...	...
notation	a	β	γ	δ	ε	...
C_5	4	5	6	7	8	...

Table 13. Notation of contours

2. By molecules: A molecule is constituted by contours linked by kinematic joints of class 5. This latter is represented by a line.

Let us remember that in the molecules proposed by Davies and Crossley, based on Franke's notation (Davies & Crossley, 1966), a circle is used to represent a polygonal link (ternary, quaternary etc) and a line to represent a band joining the polygonal links either by a kinamatic joint of class 5 or by one or more binary links in a chain. Whereas in the proposed method a circle represents a contour and a line only a kinematic joint of class 5.

For example in Watt's structure α is an internal contour, Γ an external one with N=6, C_5=7, I=1 and R=3.
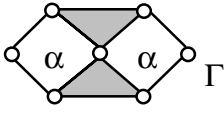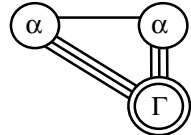
structure	Contours	notation	molecule
α α Γ	Internal (α)	◯	α — α
	External (Γ)		
	a - a - Γ	◎	Γ

Table 14. Watt's structure and its associated molecule

Table 15 represents the sixteen structures, with one degree of mobility, obtained by logical equations method presented in § 3.2.3. Their contours and molecules are also shown.

structure	kinematic graph	contour	molecule
$G_{1\text{-}8}^1$		$a\text{-}a\text{-}a\text{-}E$	
$G_{1\text{-}8}^2$		$\beta\text{-}a\text{-}a\text{-}\Delta$	
$G_{1\text{-}8}^3$		$\gamma\text{-}a\text{-}a\text{-}\Gamma$	
$G_{1\text{-}8}^4$		$a\text{-}a\text{-}\beta\text{-}\Delta$	
$G_{1\text{-}8}^5$		$a\text{-}a\text{-}a\text{-}E$	
$G_{1\text{-}8}^6$		$a\text{-}a\text{-}\gamma\text{-}\Gamma$	
$G_{1\text{-}8}^7$		$\beta\text{-}\beta\text{-}\beta\text{-}B$	
$G_{1\text{-}8}^8$		$\beta\text{-}a\text{-}\beta\text{-}\Gamma$	
$G_{1\text{-}8}^9$		$\beta\text{-}a\text{-}\beta\text{-}\Gamma$	
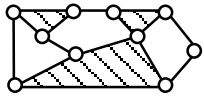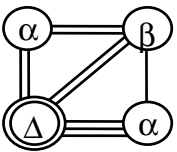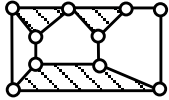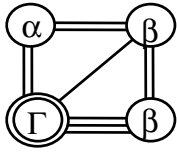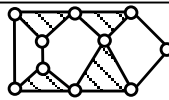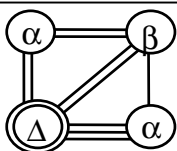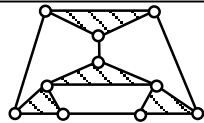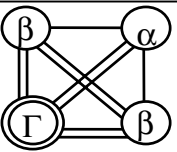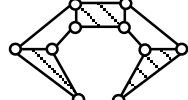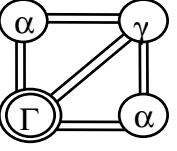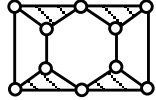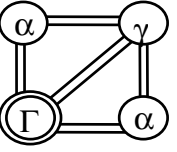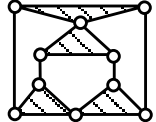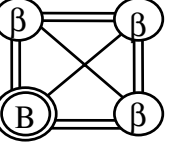
$G_{1\text{-}8}^{10}$		$a - \beta - a - \Delta$	
$G_{1\text{-}8}^{11}$		$a - \beta - \beta - \Gamma$	
$G_{1\text{-}8}^{12}$		$a - \beta - a - \Delta$	
$G_{1\text{-}8}^{13}$		$\beta - a - \beta - \Gamma$	
$G_{1\text{-}8}^{14}$		$a - \gamma - a - \Gamma$	
$G_{1\text{-}8}^{15}$		$a - \gamma - a - \Gamma$	
$G_{1\text{-}8}^{16}$		$\beta - \beta - \beta - B$	

Table 15. Closed structures of type $G_{1,8}$ ant theirs molecules (cf. § 3.2.3)

(Note: 1, 2,..., 16: arbitrary number of the structure)

5. Application in the description of the main structure of industrial robots

The presented method for mechanisms' description is applied for the description and classification of the main structures of industrial robots with different degrees of mobility.

5.1 Robots with open kinematic structures

The simplest structures applied in robotic design are the open loop simple chain. In this case the definitions of contours and molecules have no physical sense. Such a structure is shown in figure 5.
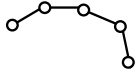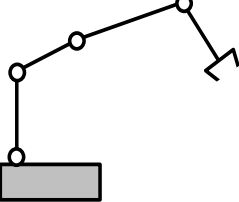
kinematic graph	contour	molecule	mechanism	robot
	non valid	non valid		
legend: ▭ frame				

Figure 5. Staübli robot

For example STAÜBLI RX90L industrial robot (average carrier) has a simple (open) kinematic chain. With a capacity of 6 kg and operating range of 1185 mm it is used in different industrial fields like painting, handling, serving of machine tools etc. .(source: http:// www.staubli.com/ web/web_fr /robot/division.nsf).

5.2 Robots with closed kinematic structures

5.2.1 Robot with a main structure having one degree of mobility and I=0

For M=1 and I=0, the most simple structure is the four-links mechanism (cf. § 3.2.1). This structure has only one closed loop to which correspond two identical contours α and A. To one of its links an end-effector is attached, allowing it to manipulate objects. The result of this operation is the mechanism corresponding to the main structure of the level 1 Pick and Place industrial robot. The structure is shown in figure 6:
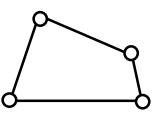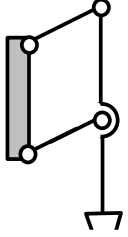
kinematic graph	contour	molecule	mechanism	robot
	$a - A$			

Figure 6. Pick and Place robot

This is the simplest robot with closed structure. Its task consists in moving light parts from one place to another, which explains its name "Pick and Place". This robot is essentially used in clock and pharmaceutical industries.

5.2.2 Robot with a main structure having two degrees of mobility and I=0

For M=2 and I=0, there is only one possible kinematic chain obtained by the schematic of § 3.3. The main structure's kinematic graph of this robot is shown in figure 7. In order for the robot to be able to manipulate objects, this planar structure is connected to a frame by transforming one of its binary links to a ternary one (the choice is arbitrary). Finally an end-effector is added by the same process. The result of these operations is the mechanism corresponding to the main structure of the level 1 HPR Hitachi industrial robot.
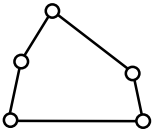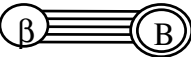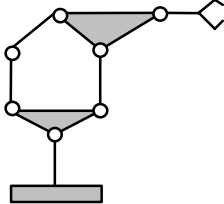
kinematic graph	contour	molecule	mechanism	robot
	$\beta - B$			

Figure 7. HPR Hitachi robot
(source: http://www.hqrd.hitachi.co.jp/merl/robot/robot1.cfm)

5.2.3 Robot with a main structure having two degrees of mobility and I=1

In this case there are many possible structures generated from the four structures of table 11, according to the choice of frame position. Let us consider solution number one of the first column (c.f. table 11, § 3.3). In the same process carried out previously, by linking this structure to a frame and adding an end-effector to it, a structure is obtained corresponding to the *main structure* (cf. fig. 3) of the level 2 AKR-3000 robot. Being able to manipulate a weight up to 15 daN, this robot is essentially used for paintwork.

kinematic graph	contour	Molecule
	$\beta - a - \Delta$	

Mechanisme	robot
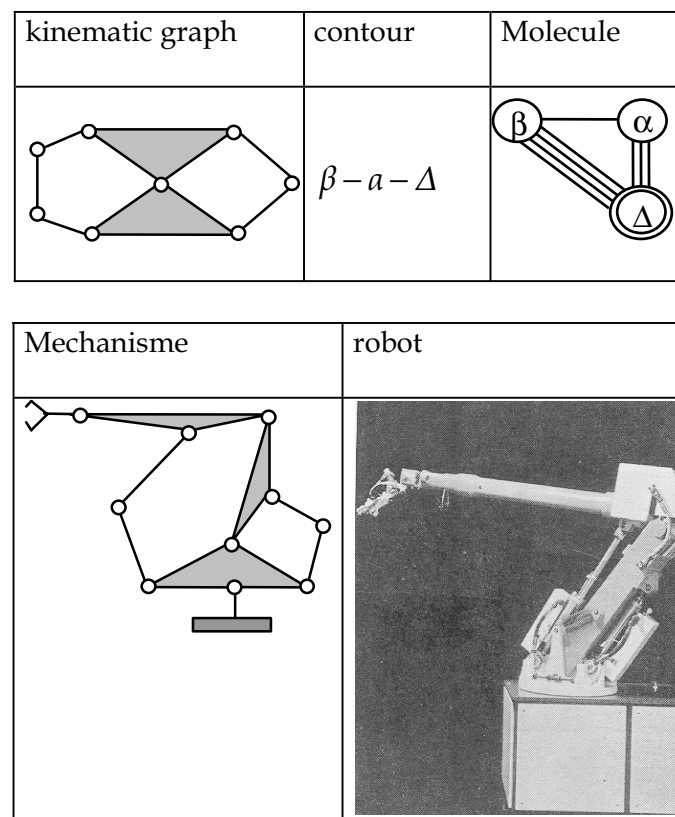
Figure 8. AKR-3000 Robot

5.2.4 Robot with a main structure having one degree of mobility and I=2

For M=1 and I=2 there are many possible structures (c.f. Table 15). Structure $G_{1,8}{}^{10}$ has been chosen here. By linking this structure to a frame and adding an end-effector to it, a structure is obtained corresponding to the main level 3 structure of the Mitsubishi AS50VS Robot presented in Fig. 9.

kinematic graph	contour	molecule
	$a - \beta - a - \Delta$	

mechanism	robot
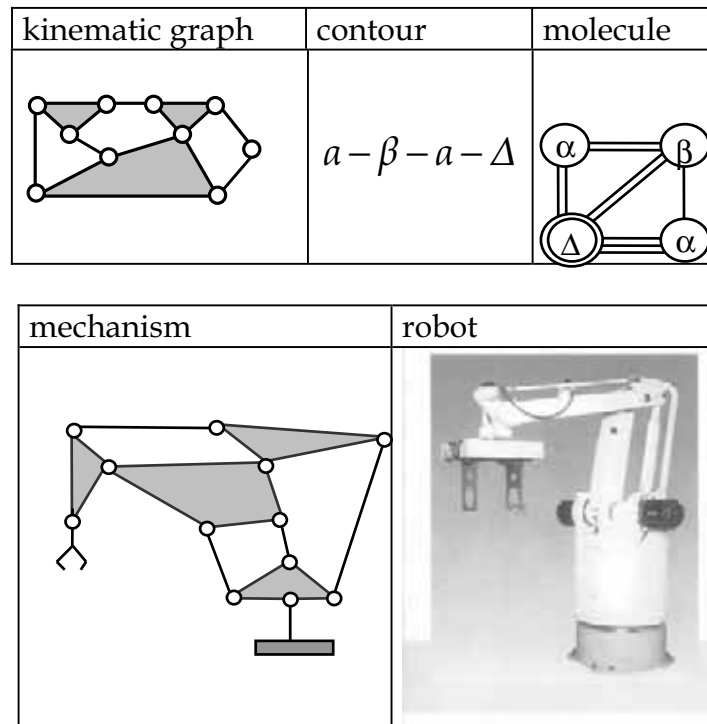
Figure 9. Mitsubishi Electric robot

5.2.5 Robot with a main structure having two degrees of mobility and I=2

The starting point for generating of the kinematic structure is the first logical equation of Table 7 (Watt's structure). For the desired robot, the $G_{1,6}$ structure thus obtained lacks one degree of mobility and five links. The following operations allow its completion (cf. fig. 10). Adding to this structure a frame and an end-effector the resulting mechanism of this operation corresponds to the main structure of the level 4 HPR Andromat robot.

stage/ logical equation	kinematic graph	contour	molecule
first: addition of three links and one degree of mobility $G_{1,6} + A_{1,3} = G_{2,9}$		$\gamma - a - a - E$	
second: addition of two links $G_{2,9} + A_{0,2} = G_{2,11}$	adopted solution	$a - \gamma - a - \gamma - E$	

mechanism	robot
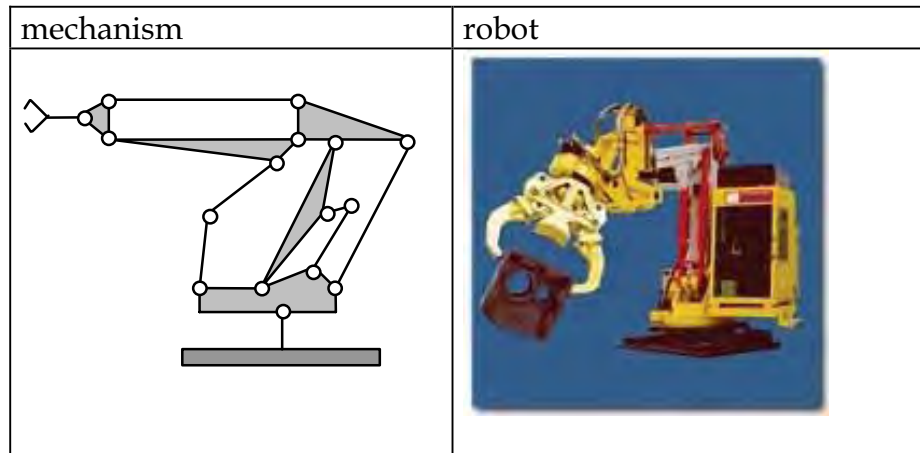
Figure 10. Evolution of the generation of the Andromat robot topological chain
(source: http://arbejdsmiljoweb.dk/upload/rap_1_doc.doc)

Among the one hundred and ten available structures of type $G_{2,11}$, a robot manufacturer has implemented the required solution above in order to design the main structure of the Andromat robot. According to the rules defined in § 3.1.2. the frame, initially a quaternary link, was transformed into a quintarny one and the binary link, where the end-effector was attached, into a ternary one.

This robot is equipped with a pantographic system with a working range of 2,5 m and weight range from 250 kg up to 2000 kg. The Andromat is a world-renowned manipulator, which is widely and successfully used in foundry and forging industries enabling operators to lift and manipulate heavy and awkward components in hostile and dangerous environments. (source: http://www.pearsonpanke.co.uk/).
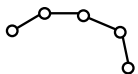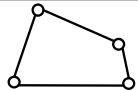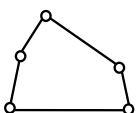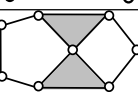
During the initial design of the MS of robots, the validation of their topological structures may be done by studying the kinematic graphs of their main structures. The representation by molecules mainly yields to the usual structural diagram of the mechanism in order to visualise and simplify. This allows the classification of their structures and their assignation to different classes of structures, taking into account of their complexity expressed by the number of closed loops. Those points are the subject of the next paragraph.

6. Classification of industrial robots structures

The structures of robots with *simple kinematic chains* may be represented by one open kinematic structures of type A. We call these open structures 0 (zero) level structures. Many industrial robots are of the same type for example: MA 23 Arm, SCARA carrier, AID-5V, Seiko 700, Versatran Vertical 80, Puma 500, Kawasaki Js-2, Toshiba SR-854HSP and Yamaha robots (Ferreti, 1981; Rob-Aut, 1996).

The main structures of robots with *closed kinematic chains* may be represented by closed kinematic chains of type G derived from MMT. The Pick and Place robot, for instance, has only one closed chain. This is a level 1 (one) robot (cf. § 5.2.1). There are other industrial robots of the same level for example: Tokico, Pana-Robo by Panasonic, SK 16 and SK 120 by Yaskawa, SC 35 Nachi etc (Rob-Aut, 1996).

The main structure of the AKR-3000 robot is composed of two closed loops represented by two internal contours in its molecule. This is a level 2 (two) robot. The main structure of Moise-Pelecudi robot (Manolescu et al, 1987) is composed of three closed chains defining a level 3 (three) robot. The main structure of the Andromat robot is composed of four closed chains. This is a level 4 (four) robot etc. Hence the level n of a robot is defined by the number n of internal contours in its molecule. Table 16 completes this classification of certain robots presented by Ferreti in (Ferreti, 1981):

robot	manufacturer	main structure of the robot	contour	nb. of internal contours	level
Nordson	Nordson France		- (simple chain)	0	zero
Robomatic	Binks Manufacturing Co.			0	zero
Cincinnati T3, HT3	Cincinnati		$a - A$	1	one
HPR- Hita-chi	Milacron France		$\beta - B$	1	one
RASN	AOIP Kremlin, Robotique AKR		$\beta - a - \Delta$	2	two
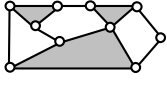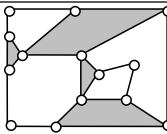
AS50VS	Mitsubishi Electric/ Japan		$a - \beta - a - \Delta$	3	three
Andromat	…/Sweden		$a - \gamma - a - \gamma -$	4	four

Table 16. Levels of different industrial robots

7. Conclusions and Future Plans

In this chapter we presented an overview about the chronology of design process of an industrial robot kinematic chain. The method for symbolical synthesis of planar link mechanisms in robotics presented here allows the generation of plane mechanical structures with different degrees of mobility. Based on the notion of *logical equations*, this enables the same structures obtained using different methods to be found (intuitive methods, Assur's groups, transformation of binary chains etc).

The goal being to represent the complexity of the topological structure of an industrial robot, a new method for description of mechanisms was proposed. It is based on the notions of *contours* and *molecules*. Its advantage, during the initial phase of the design of the robots, is that the validation of their topological structures can be done by comparing their respective molecules. That makes it possible to reduce their number by eliminating those which are isomorphic.

The proposed method is afterwards applied for the description of closed structures derived from MMT for different degrees of mobility. It is then applied to the description and to the classification of the *main structures* of different industrial robots. The proposed method permits the simplification of the visualisation of their topological structures. Finally a classification of industrial robots of different levels taking into account the number of closed loops in their molecules is presented.

In addition to the geometrical, kinematical and dynamic performances, the design of a mechanical system supposes to take into account, the constraints of the kinematic chain according to the:

-   position of the frame,
-   position of the end-effector,
-   and position of the actuators.

The two first aspects above are currently the subjects of our research. The problem is how to choose among the possible structures provided by MMT ac-

cording to the position of the frame and the end-effector. As there may be a large number of these mechanisms, it is usually difficult to make a choice among the available structures in the initial design phase of the robot chain. In fact, taking into account the symmetries it can be noticed that there are a significant number of isomorphic structures according to the position of the frame and of the end-effector of the robot. Our future objectives are:

- to find planar mechanisms with revolute joints that provide guidance of a moving frame, e.g. the end-effector of an industrial robot, relative to a base frame with a given degree of freedom,
- to reduce the number of kinematic structures provided by MMT, which are suitable for robotics applications, taking into account the symmetries the two criteria being the position of the frame and of the end-effector of the robot.

8. References

Abo-Hammour, Z.S.; Mirza, N.M.; Mirza, S.A. & Arif, M. (2002). Cartesian path generation of robot manipulators continuous genetic algorithms, *Robotics and autonomous systems*. Dec 31, 41 (4), pp.179-223.

Artobolevski, I. (1977). *Théorie des mécanismes*, Mir, Moscou.

Belfiore, N.P. (2000). Distributed databases for the development of mechanisms topology, *Mechanism and Machine Theory* Vol. 35, pp. 1727-1744.

Borel, P. (1979). *Modèle de comportement des manipulateurs. Application à l'analyse de leurs performances et à leur commande automatique*, PhD Thesis, Montpellier.

Cabrera, J.A.; Simon, A. & Prado, M. (2002). Optimal synthesis of mechanisms with genetic algorithms, *Mechanism and Machine Theory*, Vol. 37, pp. 1165-1177.

Coiffet, P. (1992). *La robotique. Principes et applications*, Hermès, Paris.

Crossley, F. R. E. (1964). A cotribution to Grubler's theory in the number synthesis of plane mechanisms, Transactions of the ASME, *Journal of Engineering for Industry*, 1-8.

Crossley, F.R.E. (1966). On an unpublished work of Alt, Journal of Mechanisms, 1, 165-170.

Davies, T. & Crossley, F.R.E. (1966). Structural analysis of plan linkages by Franke's condensed notation, Pergamon Press, *Journal of Mechanisms*, Vol., 1, 171-183.

Dobrjanskyi L., Freudenstein F., 1967. Some application of graph theory to the structural analysis of mechanisms, Transactions of the ASME, Journal of Engineering for Industry, 153-158.

Erdman A., Sandor G. N., 1991. *Mechanism Design, Analysis and Synthesis*, Second edition.

Ferreti, M. (1981). Panorama de 150 manipulateurs et robots industriels, *Le Nouvel Automatisme*, Vol., 26, Novembre-Décembre, 56-77.

Gonzales-Palacois, M. & Ahjeles J. (1996). USyCaMs : A software Package for the Interactive Synthesysis of Cam Mecanisms, *Proc. 1-st Integrated Design end Manufacturing in Mechanical Engineering I.D.M.M.E.'96*, Nantes France, 1, 485-494.

Hervé L.-M., 1994. The mathematical group structure of the set of displacements, Mech. and Mach. Theory, Vol. 29, N° 1, 73-81.

Hwang, W-M. & Hwang, Y-W. (1992). Computer-aided structural synthesis of plan kinematic chains with simple joints *Mechanism and Machine Theory*, 27, N°2, 189-199.

Karouia, M. & Hervè, J.M. (2005). Asymmetrical 3-dof spherical parallel mechanisms, *European Journal of Mechanics (A/Solids)*, N°24, pp.47-57.

Khalil, W. (1976). *Modélisation et commande par calculateur du manipulateur MA-23. Extension à la conception par ordinateur des manipulateurs*, PhD Thesis. Montpellier.

Laribi, M.A. ; Mlika, A. ; Romdhane, L. & Zeghloul, S. (2004). A combined genetic algorithm-fuzzy logic method (GA-FL) in mechanisms synthesis, *Mechanism and Machine Theory* 39, pp. 717-735.

Ma, O. & Angeles, J. (1991). Optimum Architecture Design of Platform Manipulators, *IEEE*, 1130-1135.

Manolescu, N. (1964). Une méthode unitaire pour la formation des chaînes cinématiques et des mécanismes plans articulés avec différents degrés de liberté et mobilité, *Revue Roumaine Sciances. Techniques- Mécanique Appliquée*, 9, N°6, Bucarest, 1263-1313.

Manolescu, N. ; Kovacs, F.R. & Oranescu, A. (1972). Teoria Mecanismelor si a masinelor, *Editura didactica si pedagogica*, Bucuresti.

Manolescu, N (1979). A unified method for the formation of all planar joined kinematic chains and Baranov trusses, *Environment and Planning B*, 6, 447-454.

Manolescu, N. ; Tudosie, I. ; Balanescu, I. ; Burciu, D. & Ionescu, T. (1987). Structural and Kinematic Synthesis of Planar Kinematic Chain (PKC) and Mechanisms (PM) with Variable Structure During the Work, *Proc. of the 7-th World Congress, The Theory of Machines and Mechanisms*, 1, Sevilla, Spain, 45-48.

Manolescu N., 1987. Sur la structure des mécanismes en robotique, Conférence à l'Ecole centrale d'Arts et Manufactures, Paris 1987.

Merlet, J.-P. (1996). Workspace-oriented methodology for designing a parallel manipulator ", Proc. 1-st *Integrated Design end Manufacturing in Mechanical Engineering I.D.M.M.E.'96*, April 15-17, Nantes France, Tome 2, 777-786.

Mitrouchev, P. & André, P. (1999). Méthode de génération et description de mécanismes cinématiques plans en robotique, *Journal Européen des Systèmes Automatisés, 33*(3), 285-304.

Mitrouchev, P. (2001). Symbolic structural synthesis and a description method for planar kinematic chains in robotics, *European Journal of Mechanics (A Solids)*, N°20, pp.777-794.

Mruthyunjaya, T.S. (1979). Structural Synthesis by Transformation of Binary Chains, *Mechanism and Machine Theory*, 14, 221-238.

Mruthyunjaya, T.S. (1984-a). A computerized methodology for structural synthesis of kinematic chains: Part 1- Formulation, *Mechanism and Machine Theory*, 19, No.6, 487-495.

Mruthyunjaya, T.S. (1984-b). A computerized methodology for structural synthesis of kinematic chains: Part 2-Application to several fully or partially known cases, *Mechanism and Machine Theory*, 19, No.6, 497-505.

Mruthyunjaya, T.S. (1984-c). A computerized methodology for structural syn-
     thesis of kinematic chains: Part 3-Application to the new case of 10-link,
     three-freedom chains, *Mechanism and Machine Theory*, 19, No.6, 507-530.
Pieper, L. & Roth, B. (1969). The Kinematics of Manipulators Under Computer
     Control, *Proceedings 2-nd International Congress on The Theory of Machines
     and Mechanisms*, 2, 159-168.
Rao, A. C. & Deshmukh, P. B. (2001). Computer aided structural synthesis of
     planar kinematic chains obviating the test for isomorphism, *Mechanism
     and Machine Theory* 36, pp. 489-506.
Renaud, M. (1975). *Contribution à l'étude de la modélisation et de la commande des
     systèmes mécaniques articulés*, Thèse de Docteur ingénieur. Université
     Paul Sabatier, Toulouse.
Rob-Aut. (1996). La robotique au Japon, ROBotisation et AUTomatisation de la
     production, N°12, Janvier-Février, 28-32.
Roth, B. (1976). Performance Evaluation of manipulators from a kinamatic
     viewpoint, Cours de robotique. 1, IRIA.
Tejomurtula, S. & Kak, S. (1999). Inverse kinematics in robotics using neural
     networks, *Information sciences*, 116 (2-4), pp. 147-164.
Tischler, C. R.; Samuel A. E. & Hunt K. H. (1995). Kinematic chains for robot
     hands – I. Orderly number-synthesis, *Mechanism and Machine Theory*, **30**,
     No.8, pp. 1193-1215.
Touron, P. (1984). Modélisation de la dynamique des mécanismes polyarticu-
     lés. *Application à la CAO et à la simulation de robots*, Thèse, Université de
     Montpellier.
Warneke, H.J. (1977). Research activities and the I.P.A. in the field of robotics,
     Proc. of the *7-th ISIR Congress*, Tokyo, 25-35.
Woo, L. S. (1967). Type synthesis of plan linkages, Transactions of the ASME,
     Journal of Engineering for Industry, 159-172.

Robot Kinematics:
Forward and Inverse Kinematics

Serdar Kucuk and Zafer Bingul

1. Introduction

Kinematics studies the motion of bodies without consideration of the forces or moments that cause the motion. Robot kinematics refers the analytical study of the motion of a robot manipulator. Formulating the suitable kinematics models for a robot mechanism is very crucial for analyzing the behaviour of industrial manipulators. There are mainly two different spaces used in kinematics modelling of manipulators namely, Cartesian space and Quaternion space. The transformation between two Cartesian coordinate systems can be decomposed into a rotation and a translation. There are many ways to represent rotation, including the following: Euler angles, Gibbs vector, Cayley-Klein parameters, Pauli spin matrices, axis and angle, orthonormal matrices, and Hamilton 's quaternions. Of these representations, homogenous transformations based on 4x4 real matrices (orthonormal matrices) have been used most often in robotics. Denavit & Hartenberg (1955) showed that a general transformation between two joints requires four parameters. These parameters known as the Denavit-Hartenberg (DH) parameters have become the standard for describing robot kinematics. Although quaternions constitute an elegant representation for rotation, they have not been used as much as homogenous transformations by the robotics community. Dual quaternion can present rotation and translation in a compact form of transformation vector, simultaneously. While the orientation of a body is represented nine elements in homogenous transformations, the dual quaternions reduce the number of elements to four. It offers considerable advantage in terms of computational robustness and storage efficiency for dealing with the kinematics of robot chains (Funda et al., 1990).

The robot kinematics can be divided into forward kinematics and inverse kinematics. Forward kinematics problem is straightforward and there is no complexity deriving the equations. Hence, there is always a forward kinematics solution of a manipulator. Inverse kinematics is a much more difficult problem than forward kinematics. The solution of the inverse kinematics problem is computationally expansive and generally takes a very long time in the real time control of manipulators. Singularities and nonlinearities that make the

problem more difficult to solve. Hence, only for a very small class of kinematically simple manipulators (manipulators with Euler wrist) have complete analytical solutions (Kucuk & Bingul, 2004). The relationship between forward and inverse kinematics is illustrated in Figure 1.
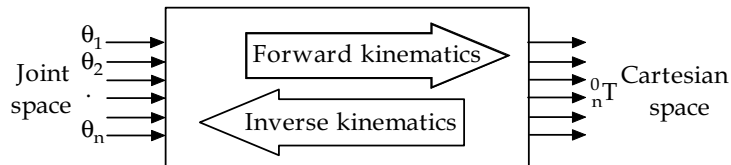
Figure 10. The schematic representation of forward and inverse kinematics.

Two main solution techniques for the inverse kinematics problem are analytical and numerical methods. In the first type, the joint variables are solved analytically according to given configuration data. In the second type of solution, the joint variables are obtained based on the numerical techniques. In this chapter, the analytical solution of the manipulators is examined rather then numerical solution.

There are two approaches in analytical method: geometric and algebraic solutions. Geometric approach is applied to the simple robot structures, such as 2-DOF planar manipulator or less DOF manipulator with parallel joint axes. For the manipulators with more links and whose arms extend into 3 dimensions or more the geometry gets much more tedious. In this case, algebraic approach is more beneficial for the inverse kinematics solution.

There are some difficulties to solve the inverse kinematics problem when the kinematics equations are coupled, and multiple solutions and singularities exist. Mathematical solutions for inverse kinematics problem may not always correspond to the physical solutions and method of its solution depends on the robot structure.

This chapter is organized in the following manner. In the first section, the forward and inverse kinematics transformations for an open kinematics chain are described based on the homogenous transformation. Secondly, geometric and algebraic approaches are given with explanatory examples. Thirdly, the problems in the inverse kinematics are explained with the illustrative examples. Finally, the forward and inverse kinematics transformations are derived based on the quaternion modeling convention.

2. Homogenous Transformation Modelling Convention

2.1. Forward Kinematics

A manipulator is composed of serial links which are affixed to each other revolute or prismatic joints from the base frame through the end-effector. Calculating the position and orientation of the end-effector in terms of the joint variables is called as forward kinematics. In order to have forward kinematics for a robot mechanism in a systematic manner, one should use a suitable kinematics model. Denavit-Hartenberg method that uses four parameters is the most common method for describing the robot kinematics. These parameters a_{i-1}, α_{i-1}, d_i and θ_i are the link length, link twist, link offset and joint angle, respectively. A coordinate frame is attached to each joint to determine DH parameters. Z_i axis of the coordinate frame is pointing along the rotary or sliding direction of the joints. Figure 2 shows the coordinate frame assignment for a general manipulator.
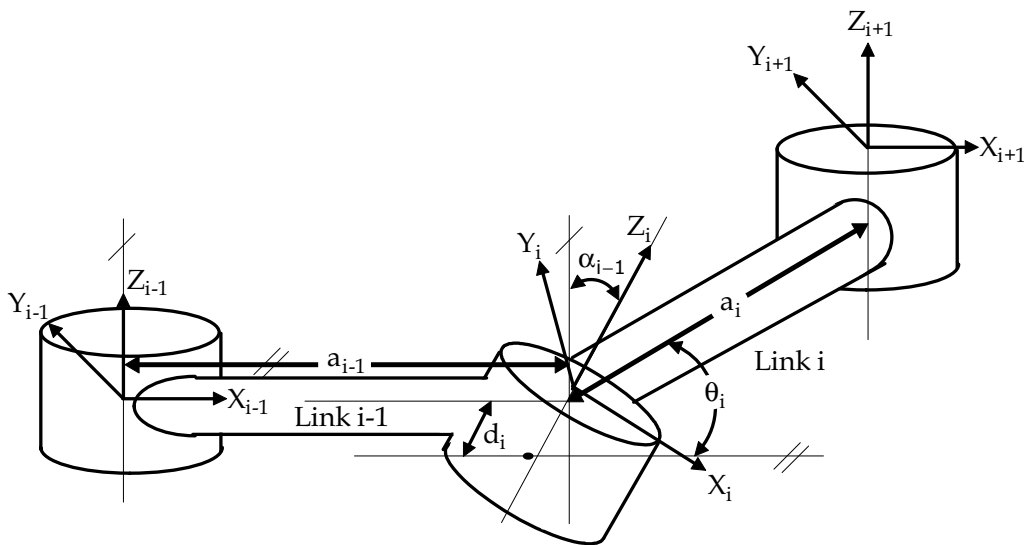
Figure 2. Coordinate frame assignment for a general manipulator.

As shown in Figure 2, the distance from Z_{i-1} to Z_i measured along X_{i-1} is assigned as a_{i-1}, the angle between Z_{i-1} and Z_i measured along X_i is assigned as α_{i-1}, the distance from X_{i-1} to X_i measured along Z_i is assigned as d_i and the angle between X_{i-1} to X_i measured about Z_i is assigned as θ_i (Craig, 1989).
The general transformation matrix $^{i-1}_{i}T$ for a single link can be obtained as follows.

$$^{i-1}_{i}T = R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)Q_i(d_i)$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where R_x and R_z present rotation, D_x and Q_i denote translation, and $c\theta_i$ and $s\theta_i$ are the short hands of $\cos\theta_i$ and $\sin\theta_i$, respectively. The forward kinematics of the end-effector with respect to the base frame is determined by multiplying all of the $^{i-1}_{i}T$ matrices.

$$^{base}_{end\_effector}T = ^{0}_{1}T\,^{1}_{2}T\,...\,^{n-1}_{n}T \tag{2}$$

An alternative representation of $^{base}_{end\_effector}T$ can be written as

$$^{base}_{end-effector}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

where r_{kj}'s represent the rotational elements of transformation matrix (k and j=1, 2 and 3). p_x, p_y and p_z denote the elements of the position vector. For a six jointed manipulator, the position and orientation of the end-effector with respect to the base is given by

$$^{0}_{6}T = ^{0}_{1}T(q_1)\,^{1}_{2}T(q_2)\,^{2}_{3}T(q_3)\,^{3}_{4}T(q_4)\,^{4}_{5}T(q_5)\,^{5}_{6}T(q_6) \tag{4}$$

where q_i is the joint variable (revolute or prismatic joint) for joint i, (i=1, 2, .. .6).

Example 1.

As an example, consider a 6-DOF manipulator (Stanford Manipulator) whose rigid body and coordinate frame assignment are illustrated in Figure 3. Note that the manipulator has an Euler wrist whose three axes intersect at a common point. The first (RRP) and last three (RRR) joints are spherical in shape. P and R denote prismatic and revolute joints, respectively. The DH parameters corresponding to this manipulator are shown in Table 1.
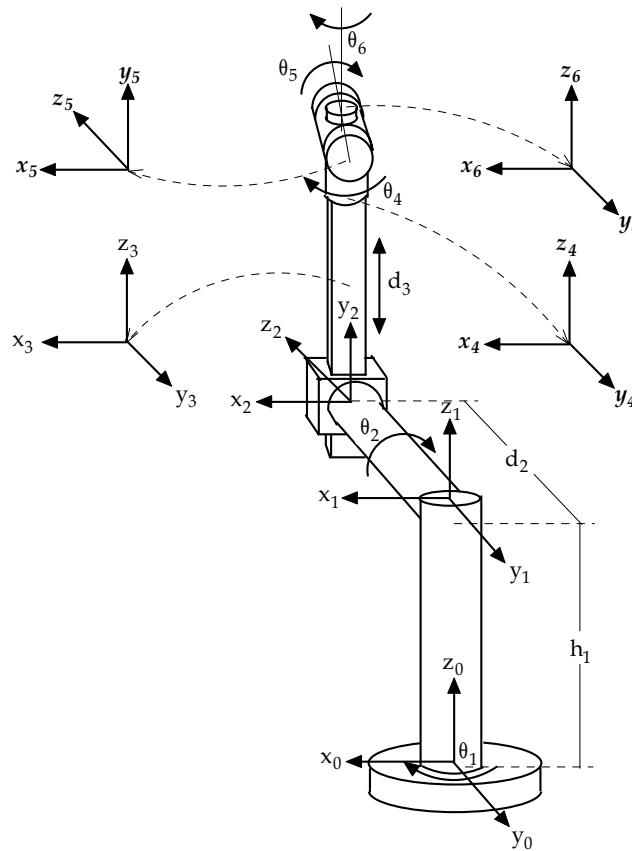
Figure 3. Rigid body and coordinate frame assignment for the Stanford Manipulator.

i	θ_i	α_{i-1}	a_{i-1}	d_i
1	θ_1	0	0	h_1
2	θ_2	90	0	d_2
3	0	-90	0	d_3
4	θ_4	0	0	0
5	θ_5	90	0	0
6	θ_6	-90	0	0

Table 1. DH parameters for the Stanford Manipulator.

It is straightforward to compute each of the link transformation matrices using equation 1, as follows.

$$
{}^0_1T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & h_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}
$$

$$
{}^1_2T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & -1 & -d_2 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}
$$

$$
{}^2_3T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}
$$

$$
{}^3_4T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & 0 \\ s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}
$$

$$
{}^4_5T = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_5 & c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}
$$

$$
{}^5_6T = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_6 & -c\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}
$$

The forward kinematics of the Stanford Manipulator can be determined in the form of equation 3 multiplying all of the ${}^{i-1}_iT$ matrices, where i=1,2, ..., 6. In this case, ${}^0_6T$ is given by

$$
{}^0_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\tag{11}
$$

where

$$r_{11} = -s\theta_6(c\theta_4 s\theta_1 + c\theta_1 c\theta_2 s\theta_4) - c\theta_6(c\theta_5(s\theta_1 s\theta_4 - c\theta_1 c\theta_2 c\theta_4) + c\theta_1 s\theta_2 s\theta_5)$$

$$r_{12} = s\theta_6(c\theta_5(s\theta_1 s\theta_4 - c\theta_1 c\theta_2 c\theta_4) + c\theta_1 s\theta_2 s\theta_5) - c\theta_6(c\theta_4 s\theta_1 + c\theta_1 c\theta_2 s\theta_4)$$

$$r_{13} = s\theta_5(s\theta_1 s\theta_4 - c\theta_1 c\theta_2 c\theta_4) - c\theta_1 c\theta_5 s\theta_2$$

$$r_{21} = s\theta_6(c\theta_1 c\theta_4 - c\theta_2 s\theta_1 s\theta_4) + c\theta_6(c\theta_5(c\theta_1 s\theta_4 + c\theta_2 c\theta_4 s\theta_1) - s\theta_1 s\theta_2 s\theta_5)$$

$$r_{22} = c\theta_6(c\theta_1 c\theta_4 - c\theta_2 s\theta_1 s\theta_4) - s\theta_6(c\theta_5(c\theta_1 s\theta_4 + c\theta_2 c\theta_4 s\theta_1) - s\theta_1 s\theta_2 s\theta_5)$$

$$r_{23} = -s\theta_5(c\theta_1 s\theta_4 + c\theta_2 c\theta_4 s\theta_1) - c\theta_5 s\theta_1 s\theta_2$$

$$r_{31} = c\theta_6(c\theta_2 s\theta_5 + c\theta_4 c\theta_5 s\theta_2) - s\theta_2 s\theta_4 s\theta_6$$

$$r_{32} = -s\theta_6(c\theta_2 s\theta_5 + c\theta_4 c\theta_5 s\theta_2) - c\theta_6 s\theta_2 s\theta_4$$

$$r_{33} = c\theta_2 c\theta_5 - c\theta_4 s\theta_2 s\theta_5$$

$$p_x = d_2 s\theta_1 - d_3 c\theta_1 s\theta_2$$

$$p_y = -d_2 c\theta_1 - d_3 s\theta_1 s\theta_2$$

$$p_z = h_1 + d_3 c\theta_2$$

2.1.1 Verification of Mathematical model

In order to check the accuracy of the mathematical model of the Stanford Manipulator shown in Figure 3, the following steps should be taken. The general position vector in equation 11 should be compared with the zero position vector in Figure 4.
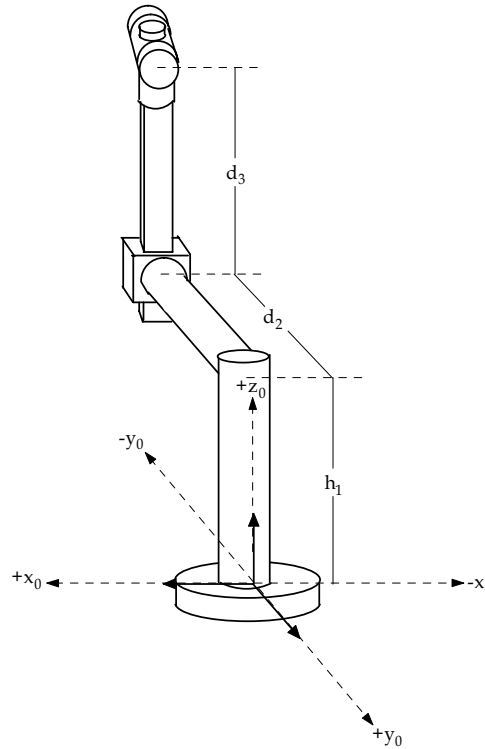
Figure 4. Zero position for the Stanford Manipulator.

The general position vector of the Stanford Manipulator is given by

$$
\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} d_2 s\theta_1 - d_3 c\theta_1 s\theta_2 \\ -d_2 c\theta_1 - d_3 s\theta_1 s\theta_2 \\ h_1 + d_3 c\theta_2 \end{bmatrix} \tag{12}
$$

In order to obtain the zero position in terms of link parameters, let's set $\theta_1 = \theta_2 = 0^\circ$ in equation 12.

$$
\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} d_2 s(0^\circ) - d_3 c(0^\circ) s(0^\circ) \\ -d_2 c(0^\circ) - d_3 s(0^\circ) s(0^\circ) \\ h_1 + d_3 c(0^\circ) \end{bmatrix} = \begin{bmatrix} 0 \\ -d_2 \\ h_1 + d_3 \end{bmatrix} \tag{13}
$$

All of the coordinate frames in Figure 3 are removed except the base which is the reference coordinate frame for determining the link parameters in zero position as in Figure 4. Since there is not any link parameters observed in the direction of $+x_0$ and $-x_0$ in Figure 4, $p_x=0$. There is only d_2 parameter in $-y_0$ direction so p_y equals $-d_2$. The parameters h_1 and d_3 are the $+z_0$ direction, so p_z equals $h_1 + d_3$. In this case, the zero position vector of Stanford Manipulator are obtained as following

$$
\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} 0 \\ -d_2 \\ h_1 + d_3 \end{bmatrix}
\tag{14}
$$

It is explained above that the results of the position vector in equation 13 are identical to those obtained by equation 14. Hence, it can be said that the mathematical model of the Stanford Manipulator is driven correctly.

2.2. Inverse Kinematics

The inverse kinematics problem of the serial manipulators has been studied for many decades. It is needed in the control of manipulators. Solving the inverse kinematics is computationally expansive and generally takes a very long time in the real time control of manipulators. Tasks to be performed by a manipulator are in the Cartesian space, whereas actuators work in joint space. Cartesian space includes orientation matrix and position vector. However, joint space is represented by joint angles. The conversion of the position and orientation of a manipulator end-effector from Cartesian space to joint space is called as inverse kinematics problem. There are two solutions approaches namely, geometric and algebraic used for deriving the inverse kinematics solution, analytically. Let's start with geometric approach.

2.2.1 Geometric Solution Approach

Geometric solution approach is based on decomposing the spatial geometry of the manipulator into several plane geometry problems.It is applied to the simple robot structures, such as, 2-DOF planer manipulator whose joints are both revolute and link lengths are l_1 and l_2 shown in Figure 5a. Consider Figure 5b in order to derive the kinematics equations for the planar manipulator.

The components of the point P (p_x and p_y) are determined as follows.
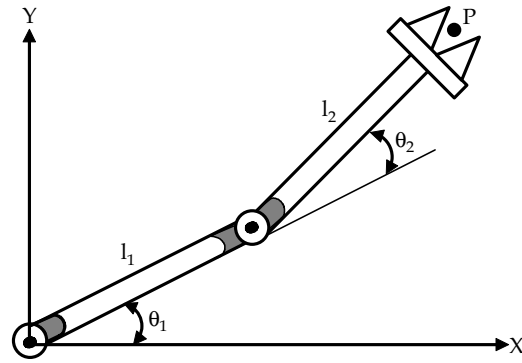
(a)
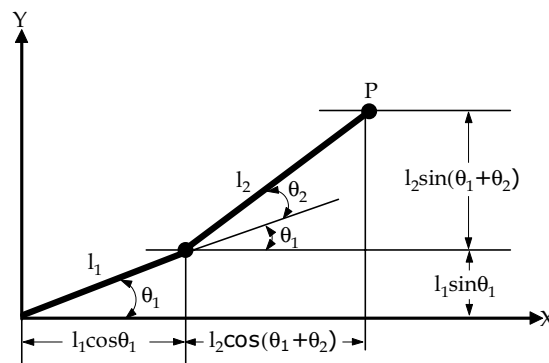
(b)

Figure 5. a) Planer manipulator; b) Solving the inverse kinematics based on trigo-nometry.

$$p_x = l_1 c\theta_1 + l_2 c\theta_{12} \tag{15}$$

$$p_y = l_1 s\theta_1 + l_2 s\theta_{12} \tag{16}$$

where $c\theta_{12} = c\theta_1 c\theta_2 - s\theta_1 s\theta_2$ and $s\theta_{12} = s\theta_1 c\theta_2 + c\theta_1 s\theta_2$. The solution of θ_2 can be computed from summation of squaring both equations 15 and 16.

$$p_x^2 = l_1^2 c^2\theta_1 + l_2^2 c^2\theta_{12} + 2l_1 l_2 c\theta_1 c\theta_{12}$$
$$p_y^2 = l_1^2 s^2\theta_1 + l_2^2 s^2\theta_{12} + 2l_1 l_2 s\theta_1 s\theta_{12}$$
$$p_x^2 + p_y^2 = l_1^2 (c^2\theta_1 + s^2\theta_1) + l_2^2 (c^2\theta_{12} + s^2\theta_{12}) + 2l_1 l_2 (c\theta_1 c\theta_{12} + s\theta_1 s\theta_{12})$$

Since $c^2\theta_1 + s^2\theta_1 = 1$, the equation given above is simplified as follows.

$$p_x^2 + p_y^2 = l_1^2 + l_2^2 + 2l_1l_2(c\theta_1[c\theta_1c\theta_2 - s\theta_1s\theta_2] + s\theta_1[s\theta_1c\theta_2 + c\theta_1s\theta_2])$$
$$p_x^2 + p_y^2 = l_1^2 + l_2^2 + 2l_1l_2(c^2\theta_1c\theta_2 - c\theta_1s\theta_1s\theta_2 + s^2\theta_1c\theta_2 + c\theta_1s\theta_1s\theta_2)$$
$$p_x^2 + p_y^2 = l_1^2 + l_2^2 + 2l_1l_2(c\theta_2[c^2\theta_1 + s^2\theta_1])$$
$$p_x^2 + p_y^2 = l_1^2 + l_2^2 + 2l_1l_2c\theta_2$$

and so

$$c\theta_2 = \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1l_2} \tag{17}$$

Since, $c^2\theta_i + s^2\theta_i = 1$ $(i = 1,2,3,\ldots\ldots)$, $s\theta_2$ is obtained as

$$s\theta_2 = \pm\sqrt{1 - \left(\frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1l_2}\right)^2} \tag{18}$$

Finally, two possible solutions for θ_2 can be written as

$$\theta_2 = A\tan 2\left(\pm\sqrt{1 - \left(\frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1l_2}\right)^2}, \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1l_2}\right) \tag{19}$$

Let's first, multiply each side of equation 15 by $c\theta_1$ and equation 16 by $s\theta_1$ and add the resulting equations in order to find the solution of θ_1 in terms of link parameters and the known variable θ_2.

$$c\theta_1p_x = l_1c^2\theta_1 + l_2c^2\theta_1c\theta_2 - l_2c\theta_1s\theta_1s\theta_2$$
$$s\theta_1p_y = l_1s^2\theta_1 + l_2s^2\theta_1c\theta_2 + l_2s\theta_1c\theta_1s\theta_2$$
$$c\theta_1p_x + s\theta_1p_y = l_1(c^2\theta_1 + s^2\theta_1) + l_2c\theta_2(c^2\theta_1 + s^2\theta_1)$$

The simplified equation obtained as follows.

$$c\theta_1p_x + s\theta_1p_y = l_1 + l_2c\theta_2 \tag{20}$$

In this step, multiply both sides of equation 15 by $-s\theta_1$ and equation 16 by $c\theta_1$ and then adding the resulting equations produce

$$-s\theta_1 p_x = -l_1 s\theta_1 c\theta_1 - l_2 s\theta_1 c\theta_1 c\theta_2 + l_2 s^2\theta_1 s\theta_2$$
$$c\theta_1 p_y = l_1 s\theta_1 c\theta_1 + l_2 c\theta_1 s\theta_1 c\theta_2 + l_2 c^2\theta_1 s\theta_2$$
$$-s\theta_1 p_x + c\theta_1 p_y = l_2 s\theta_2 (c^2\theta_1 + s^2\theta_1)$$

The simplified equation is given by

$$-s\theta_1 p_x + c\theta_1 p_y = l_2 s\theta_2 \qquad (21)$$

Now, multiply each side of equation 20 by p_x and equation 21 by p_y and add the resulting equations in order to obtain $c\theta_1$.

$$c\theta_1 p_x^2 + s\theta_1 p_x p_y = p_x (l_1 + l_2 c\theta_2)$$
$$-s\theta_1 p_x p_y + c\theta_1 p_y^2 = p_y l_2 s\theta_2$$
$$c\theta_1 (p_x^2 + p_y^2) = p_x (l_1 + l_2 c\theta_2) + p_y l_2 s\theta_2$$

and so

$$c\theta_1 = \frac{p_x (l_1 + l_2 c\theta_2) + p_y l_2 s\theta_2}{p_x^2 + p_y^2} \qquad (22)$$

$s\theta_1$ is obtained as

$$s\theta_1 = \pm \sqrt{1 - \left(\frac{p_x (l_1 + l_2 c\theta_2) + p_y l_2 s\theta_2}{p_x^2 + p_y^2} \right)^2} \qquad (23)$$

As a result, two possible solutions for θ_1 can be written

$$\theta_1 = A \tan 2 \left(\pm \sqrt{1 - \left(\frac{p_x (l_1 + l_2 c\theta_2) + p_y l_2 s\theta_2}{p_x^2 + p_y^2} \right)^2} , \frac{p_x (l_1 + l_2 c\theta_2) + p_y l_2 s\theta_2}{p_x^2 + p_y^2} \right) \qquad (24)$$

Although the planar manipulator has a very simple structure, as can be seen, its inverse kinematics solution based on geometric approach is very cumbersome.

2.2.2 Algebraic Solution Approach

For the manipulators with more links and whose arm extends into 3 dimensions the geometry gets much more tedious. Hence, algebraic approach is chosen for the inverse kinematics solution. Recall the equation 4 to find the inverse kinematics solution for a six-axis manipulator.

$$
{}^0_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0_1T(q_1)\, {}^1_2T(q_2)\, {}^2_3T(q_3)\, {}^3_4T(q_4)\, {}^4_5T(q_5)\, {}^5_6T(q_6)
$$

To find the inverse kinematics solution for the first joint (q_1) as a function of the known elements of ${}^{base}_{end-effector}T$, the link transformation inverses are premultiplied as follows.

$$
\left[{}^0_1T(q_1)\right]^{-1}\, {}^0_6T = \left[{}^0_1T(q_1)\right]^{-1}\, {}^0_1T(q_1){}^1_2T(q_2)\, {}^2_3T(q_3)\, {}^3_4T(q_4)\, {}^4_5T(q_5)\, {}^5_6T(q_6)
$$

where $\left[{}^0_1T(q_1)\right]^{-1}{}^0_1T(q_1) = I$, I is identity matrix. In this case the above equation is given by

$$
\left[{}^0_1T(q_1)\right]^{-1}\, {}^0_6T = {}^1_2T(q_2)\, {}^2_3T(q_3)\, {}^3_4T(q_4)\, {}^4_5T(q_5)\, {}^5_6T(q_6) \tag{25}
$$

To find the other variables, the following equations are obtained as a similar manner.

$$
\left[{}^0_1T(q_1)\, {}^1_2T(q_2)\right]^{-1}\, {}^0_6T = {}^2_3T(q_3)\, {}^3_4T(q_4)\, {}^4_5T(q_5)\, {}^5_6T(q_6) \tag{26}
$$

$$
\left[{}^0_1T(q_1)\, {}^1_2T(q_2)\, {}^2_3T(q_3)\right]^{-1}\, {}^0_6T = {}^3_4T(q_4)\, {}^4_5T(q_5)\, {}^5_6T(q_6) \tag{27}
$$

$$
\left[{}^0_1T(q_1)\, {}^1_2T(q_2)\, {}^2_3T(q_3)\, {}^3_4T(q_4)\right]^{-1}\, {}^0_6T = {}^4_5T(q_5)\, {}^5_6T(q_6) \tag{28}
$$

$$
\left[{}^0_1T(q_1)\, {}^1_2T(q_2)\, {}^2_3T(q_3)\, {}^3_4T(q_4)\, {}^4_5T(q_5)\right]^{-1}\, {}^0_6T = {}^5_6T(q_6) \tag{29}
$$

There are 12 simultaneous set of nonlinear equations to be solved. The only unknown on the left hand side of equation 18 is q_1. The 12 nonlinear matrix elements of   right hand side are either zero, constant or functions of q_2 through q_6. If the elements on the left hand side which are the function of q_1 are equated with the elements on the right hand side, then the joint variable q_1

can be solved as functions of $r_{11}, r_{12}, \dots r_{33}$, p_x, p_y, p_z and the fixed link parameters. Once q_1 is found, then the other joint variables are solved by the same way as before. There is no necessity that the first equation will produce q_1 and the second q_2 etc. To find suitable equation for the solution of the inverse kinematics problem, any equation defined above (equations 25-29) can be used arbitrarily. Some trigonometric equations used in the solution of inverse kinematics problem are given in Table 2.

.

	Equations	Solutions
1	$a \sin \theta + b \cos \theta = c$	$\theta = A \tan 2(a, b) \mp A \tan 2\left(\sqrt{a^2 + b^2 - c^2}, c\right)$
2	$a \sin \theta + b \cos \theta = 0$	$\theta = A \tan 2(-b, a)$ or $\theta = A \tan 2(b, -a)$
3	$\cos \theta = a$ and $\sin \theta = b$	$\theta = A \tan 2(b, a)$
4	$\cos \theta = a$	$\theta = A \tan 2\left(\mp \sqrt{1 - a^2}, a\right)$
5	$\sin \theta = a$	$\theta = A \tan 2\left(a, \mp \sqrt{1 - a^2}\right)$

Table 2. Some trigonometric equations and solutions used in inverse kinematics

Example 2.

As an example to describe the algebraic solution approach, get back the inverse kinematics for the planar manipulator. The coordinate frame assignment is depicted in Figure 6 and DH parameters are given by Table 3.

i	θ_i	α_{i-1}	a_{i-1}	d_i
1	θ_1	0	0	0
2	θ_2	0	l_1	0
3	0	0	l_2	0

Table 3. DH parameters for the planar manipulator.
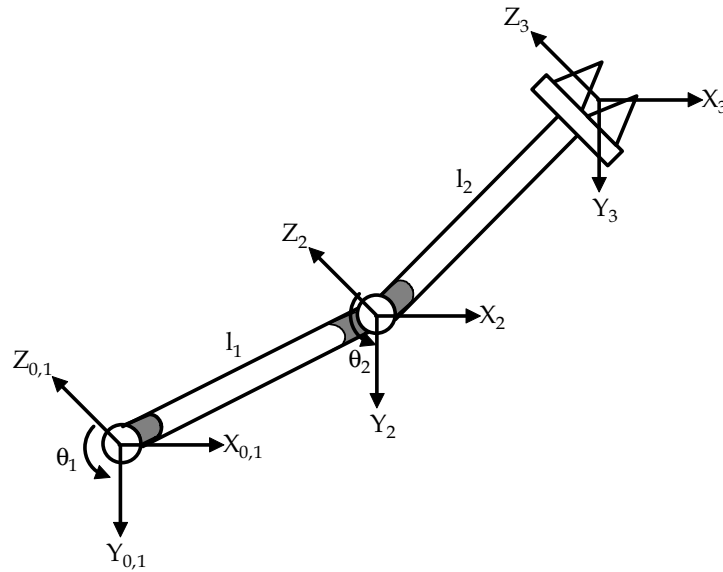
Figure 6. Coordinate frame assignment for the planar manipulator.

The link transformation matrices are given by

$$
{}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{30}
$$

$$
{}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & l_1 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{31}
$$

$$
{}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{32}
$$

Let us use the equation 4 to solve the inverse kinematics of the 2-DOF manipulator.

$$
{}^0_3T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0_1T \; {}^1_2T \; {}^2_3T \tag{33}
$$

Multiply each side of equation 33 by ${}^0_1T^{-1}$

$$
{}^0_1T^{-1} \; {}^0_3T = {}^0_1T^{-1}{}^0_1T \; {}^1_2T \; {}^2_3T \tag{34}
$$

where

$$
{}^0_1T^{-1} = \begin{bmatrix} {}^0_1R^T & -{}^0_1R^{T\,0}P_1 \\ 0\,0\,0 & 1 \end{bmatrix} \tag{35}
$$

In equation 35, ${}^0_1R^T$ and ${}^0P_1$ denote the transpose of rotation and position vector of ${}^0_1T$, respectively. Since, ${}^0_1T^{-1}\,{}^0_1T = I$, equation 34 can be rewritten as follows.

$$
{}^0_1T^{-1} \; {}^0_3T = {}^1_2T \; {}^2_3T \tag{36}
$$

Substituting the link transformation matrices into equation 36 yields

$$
\begin{bmatrix} c\theta_1 & s\theta_1 & 0 & 0 \\ -s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & l_1 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$\tag{37}$$

$$
\begin{bmatrix} . & . & . & c\theta_1 p_x + s\theta_1 p_y \\ . & . & . & -s\theta_1 p_x + c\theta_1 p_y \\ . & . & . & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} . & . & . & l_2 c\theta_2 + l_1 \\ . & . & . & l_2 s\theta_2 \\ . & . & . & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Squaring the (1,4) and (2,4) matrix elements of each side in equation 37

$$c^2\theta_1 p_x^2 + s^2\theta_1 p_y^2 + 2p_x p_y c\theta_1 s\theta_1 = l_2^2 c^2\theta_2 + 2l_1 l_2 c\theta_2 + l_1^2$$

$$s^2\theta_1 p_x^2 + c^2\theta_1 p_y^2 - 2p_x p_y c\theta_1 s\theta_1 = l_2^2 s^2\theta_2$$

and then adding the resulting equations above gives

$$p_x^2(c^2\theta_1 + s^2\theta_1) + p_y^2(s^2\theta_1 + c^2\theta_1) = l_2^2(c^2\theta_2 + s^2\theta_2) + 2l_1 l_2 c\theta_2 + l_1^2$$

$$p_x^2 + p_y^2 = l_2^2 + 2l_1 l_2 c\theta_2 + l_1^2$$

$$c\theta_2 = \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1 l_2}$$

Finally, two possible solutions for θ_2 are computed as follows using the fourth trigonometric equation in Table 2.

$$\theta_2 = A\tan 2\left(\mp\sqrt{1 - \left[\frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right]^2}, \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \tag{38}$$

Now the second joint variable θ_2 is known. The first joint variable θ_1 can be determined equating the (1,4) elements of each side in equation 37 as follows.

$$c\theta_1 p_x + s\theta_1 p_y = l_2 c\theta_2 + l_1 \tag{39}$$

Using the first trigonometric equation in Table 2 produces two potential solutions.

$$\theta_1 = A\tan 2(p_y, p_x) \mp A\tan 2(\sqrt{p_y^2 + p_x^2 - (l_2 c\theta_2 + l_1)^2}, l_2 c\theta_2 + l_1) \tag{40}$$

Example 3.

As another example for algebraic solution approach, consider the six-axis Stanford Manipulator again. The link transformation matrices were previously developed. Equation 26 can be employed in order to develop equation 41. The inverse kinematics problem can be decoupled into inverse position and orientation kinematics. The inboard joint variables (first three joints) can be solved using the position vectors of both sides in equation 41.

$$\left[{}^0_1T\ {}^1_2T\right]^{-1}\ {}^0_6T = {}^2_3T\ {}^3_4T\ {}^4_5T\ {}^5_6T \tag{41}$$

$$\begin{bmatrix} . & . & . & c\theta_2(c\theta_1 p_x + s\theta_1 p_y) + s\theta_2(p_z - h_1) \\ . & . & . & -s\theta_2(c\theta_1 p_x + s\theta_1 p_y) + c\theta_2(p_z - h_1) \\ . & . & . & s\theta_1 p_x - c\theta_1 p_y - d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} . & . & . & 0 \\ . & . & . & d_3 \\ . & . & . & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The revolute joint variables θ_1 and θ_2 are obtained equating the (3,4) and (1,4) elements of each side in equation 41 and using the first and second trigonometric equations in Table 2, respectively.

$$\theta_1 = A\tan 2(p_x, -p_y) \pm A\tan 2(\sqrt{p_x^2 + p_y^2 - d_2^2}, d_2) \qquad (42)$$

$$\theta_2 = \pm A\tan 2(c\theta_1 p_x + s\theta_1 p_y, -p_z + h_1) \qquad (43)$$

The prismatic joint variable d_3 is extracted from the (2,4) elements of each side in equation 41 as follows.

$$d_3 = -s\theta_2(c\theta_1 p_x + s\theta_1 p_y) + c\theta_2(p_z - h_1) \qquad (44)$$

The last three joint variables may be found using the elements of rotation matrices of each side in equation 41. The rotation matrices are given by

$$\begin{bmatrix} . & . & r_{33}s\theta_2 + r_{13}c\theta_1 c\theta_2 + r_{23}c\theta_2 s\theta_1 & . \\ d & e & r_{33}c\theta_2 - r_{13}c\theta_1 s\theta_2 - r_{23}s\theta_1 s\theta_2 & . \\ . & . & r_{13}s\theta_1 - r_{23}c\theta_1 & . \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} . & . & -c\theta_4 s\theta_5 & . \\ c\theta_6 s\theta_5 & -s\theta_5 s\theta_6 & c\theta_5 & . \\ . & . & s\theta_4 s\theta_5 & . \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (45)$$

where $d = r_{31}c\theta_2 - r_{11}c\theta_1 s\theta_2 - r_{21}s\theta_1 s\theta_2$ and $e = r_{32}c\theta_2 - r_{12}c\theta_1 s\theta_2 - r_{22}s\theta_1 s\theta_2$. The revolute joint variables θ_5 is determined equating the (2,3) elements of both sides in equation 45 and using the fourth trigonometric equation in Table 2, as follows.

$$\theta_5 = A\tan 2\left(\pm\sqrt{1 - (r_{33}c\theta_2 - r_{13}c\theta_1 s\theta_2 - r_{23}s\theta_1 s\theta_2)^2}, r_{33}c\theta_2 - r_{13}c\theta_1 s\theta_2 - r_{23}s\theta_1 s\theta_2\right) \qquad (46)$$

Extracting $\cos\theta_4$ and $\sin\theta_4$ from (1,3) and (3,3), $\cos\theta_6$ and $\sin\theta_6$ from (2,1) and (2,2) elements of each side in equation 45 and using the third trigonomet-

ric equation in Table 2, θ_4 and θ_6 revolute joint variables can be computed, respectively.

$$\theta_4 = A\tan 2\left(\frac{r_{13}s\theta_1 - r_{23}c\theta_1}{s\theta_5} \ , \ -\frac{r_{33}s\theta_2 + r_{13}c\theta_1 c\theta_2 + r_{23}c\theta_2 s\theta_1}{s\theta_5} \right) \tag{47}$$

$$\theta_6 = A\tan 2\left(-\frac{r_{32}c\theta_2 - r_{12}c\theta_1 s\theta_2 - r_{22}s\theta_1 s\theta_2}{s\theta_5} \ , \ \frac{r_{31}c\theta_2 - r_{11}c\theta_1 s\theta_2 - r_{21}s\theta_1 s\theta_2}{s\theta_5} \right) \tag{48}$$

2.2.3 Some Drawbacks for the Solution of the Inverse Kinematics Problem

Although solution of the forward kinematics problem is steady forward, the solution of the inverse kinematics problem strictly depend on the robot structures. Here are some difficulties that should be taken in account while driving the inverse kinematics.

The structure of the six-axis manipulators having Euler wrist allows for the decoupling of the position and orientation kinematics. The geometric feature that generates this decoupling is the intersection of the last tree joint axes. Hence, their inverse kinematics problems are quite simple. On the other hand, since the orientation and position of some 6 DOF manipulators having offset wrist (whose three axes does not intersect at a common point) are coupled, such manipulators do not produce suitable equations for the analytical solution. In this case, numerical methods are employed to obtain the solution of the inverse kinematics problem.

Consider the example 3 for describing the singularity. As long as $\theta_5 \neq 0°$ and $\theta_5 \neq 180°$, θ_4 and θ_6 can be solved. A singularity of the mechanism exists when $\theta_5 = 0°$ and $\theta_5 = 180°$. In this case, the manipulator loses one or more degrees of freedom. Hence, joint angles, θ_4 and θ_6 make the same motion of the last link of the manipulator.

The inverse kinematics solution for a manipulator whose structure comprises of revolute joints generally produces multiple solutions. Each solution should be checked in order to determine whether or not they bring the end-effector to the desired poison. Suppose the planar manipulator illustrated in Figure 5, with the link lengths l_1=10 and l_2=5 in some units. Use the inverse kinematics solutions given in equations 38 and 40 to find the joint angles which bring the end-effector at the following position (p_x,p_y)=(12.99, 2.5). Substituting l_1=10, l_2=5 and (p_x,p_y)=(12.99, 2.5) values into equation 38 yields

$$\theta_2 = A\tan 2\left(\mp\sqrt{1 - \left[\frac{12.99^2 + 2.5^2 - 10^2 - 5^2}{2\cdot 10\cdot 5}\right]^2},\ \left[\frac{12.99^2 + 2.5^2 - 10^2 - 5^2}{2\cdot 10\cdot 5}\right]\right)$$

$$= A\tan 2\left(\mp\sqrt{1 - (0.4999)^2},\ 0.4999\right) \tag{49}$$
$$= A\tan 2\left(\mp 0.866,\ 0.4999\right) = \mp 60°$$

As can be seen from equation 49, θ_2 has two solutions, corresponding to the positive (+60°) and negative (-60°) sign choices. Since $\cos(\theta) = \cos(-\theta)$, one ($\theta_2 = 60°$) of above two solutions can be employed to find the numeric values of the first joint as follows.

$$\theta_1 = A\tan 2(2.5,\ 12.99)\mp$$
$$A\tan 2(\sqrt{2.5^2 + 12.99^2 - (5\cdot c(60) + 10)^2},\ 5\cdot c(60) + 10) \tag{50}$$
$$= 10.9 \mp 19.1$$

Clearly, the planar manipulator has four different mathematical solutions given by

$$S_1 = \{\theta_1 = 10.9 + 19.1 = 30°,\ \theta_2 = +60°\} \tag{51}$$

$$S_2 = \{\theta_1 = 10.9 + 19.1 = 30°,\ \theta_2 = -60°\} \tag{52}$$

$$S_3 = \{\theta_1 = 10.9 - 19.1 = -8.20°,\ \theta_2 = +60°\} \tag{53}$$

$$S_4 = \{\theta_1 = 10.9 - 19.1 = -8.20°,\ \theta_2 = -60°\} \tag{54}$$

As a result, these four sets of link angle values given by equations 51 through 54 solve the inverse kinematics problem for the planar manipulator. Figure 7 illustrates the particular positions for the planar manipulator in each of above solutions.
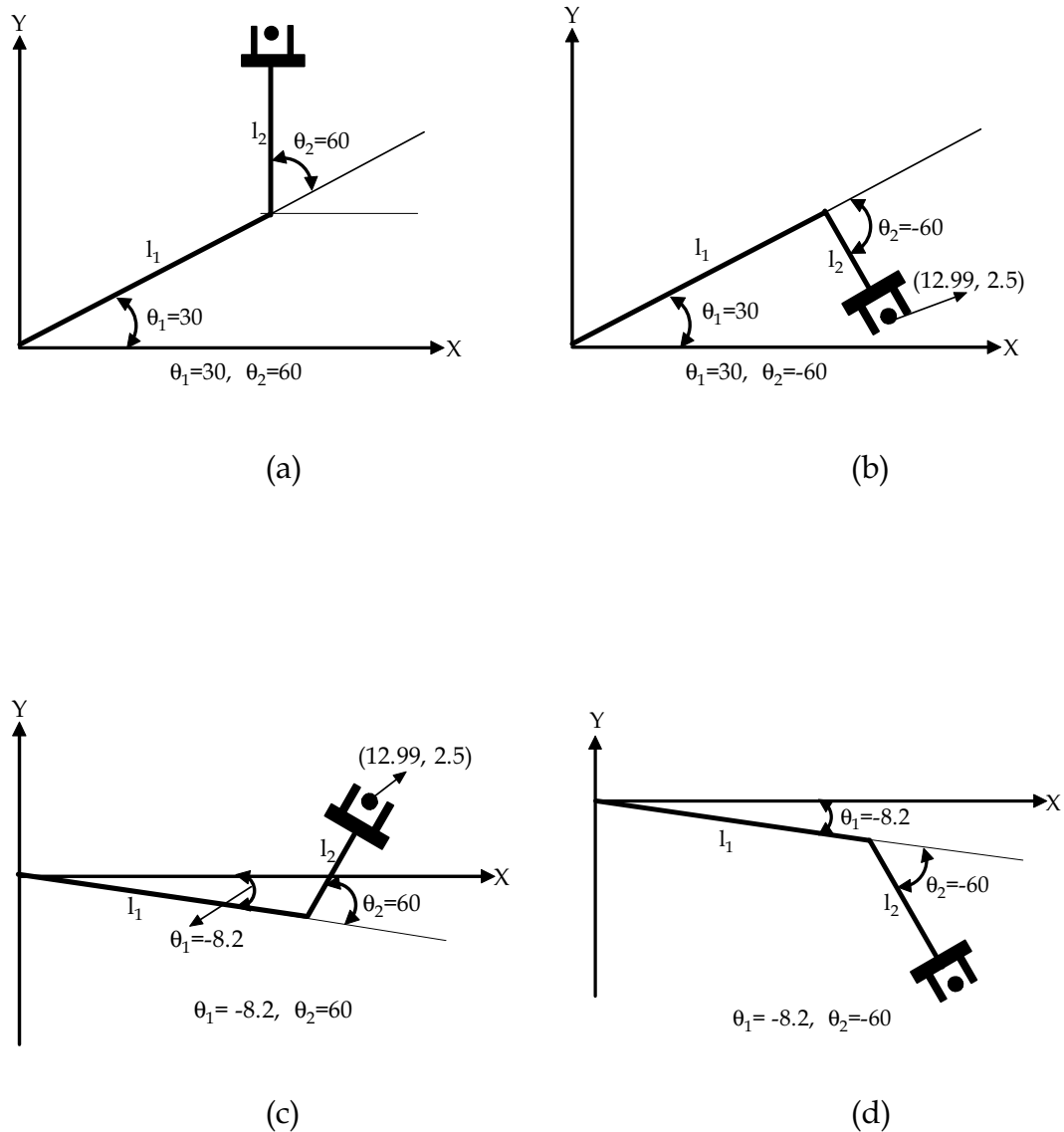
Figure 7. Four particular positions for the planar manipulator.

Although there are four different inverse kinematics solutions for the planar manipulator, only two (Figure 7b and 6c) of these bring the end-effector to the desired position of (px, py)=(12.99, 2.5).

Mathematical solutions for inverse kinematics problem may not always correspond to physical solutions. Another words, there are physical link restrictions for any real manipulator. Therefore, each set of link angle values should be

checked in order to determine whether or not they are identical with the physical link limits. Suppose, $\theta_2 = 180°$, the second link is folded completely back onto first link as shown in Figure 8. One can readily verify that this joint value is not physically attained by the planar manipulator.
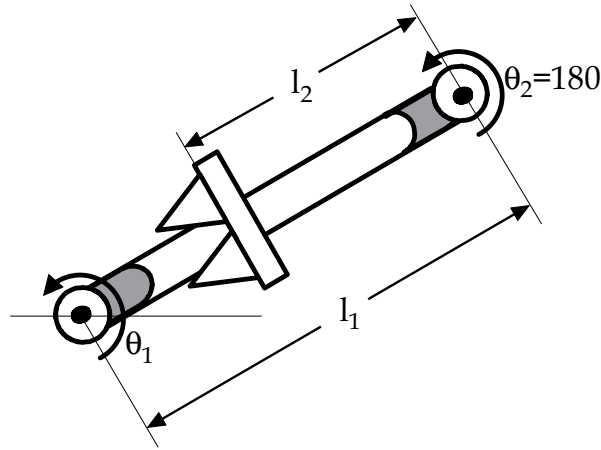
Figure 8. The second link is folded completely back onto the first link when $\theta_2 = 180°$.

3. Quaternion Modelling Convention

Formulating the suitable mathematical model and deriving the efficient algorithm for a robot kinematics mechanism are very crucial for analyzing the behavior of serial manipulators. Generally, homogenous transformation based on 4x4 real matrices is used for the robot kinematics. Although such matrices are implemented to the robot kinematics readily, they include in redundant elements (such matrices are composed of 16 elements of which four are completely trivial) that cause numerical problems in robot kinematics and also increase cost of algorithms (Funda et al., 1990). Quaternion-vector pairs are used as an alternative method for driving the robot kinematics of serial manipulator. The successive screw displacements in this method provide a very compact formulation for the kinematics equations and also reduce the number of equations obtained in each goal position, according to the matrix counterparts. Since (Hamilton, 2004)'s introduction of quaternions, they have been used in many applications, such as, classical and quantum mechanics, aerospace, geometric analysis, and robotics. While (Salamin, 1979) presented advantages of quaternions and matrices as rotational operators, the first application of the former in the kinematics was considered by (Kotelnikov, 1895). Later, general properties of quaternions as rotational operators were studied by (Pervin & Webb, 1982) who also presented quaternion formulation of moving geometric

objects. (Gu & Luh, 1987) used quaternions for computing the Jacobians for robot kinematics and dynamics. (Funda et al., 1990) compared quaternions with homogenous transforms in terms of computational efficiency. (Kim & Kumar, 1990) used quaternions for the solution of direct and inverse kinematics of a 6-DOF manipulator. (Caccavale & Siciliano, 2001) used quaternions for kinematic control of a redundant space manipulator mounted on a free-floating space-craft. (Rueda et al., 2002) presented a new technique for the robot calibration based on the quaternion-vector pairs.

3.1. Quaternion Formulation

A quaternion q is the sum of scalar (s) and three dimensional vectors (v). Other words, it is a quadrinomial expression, with a real angle θ and an axis of rotation n = ix + jy + kz, where i, j and k are imaginary numbers. It may be expressed as a quadruple q = (θ, x, y, z) or as a scalar and a vector q = (θ, u), where u= x, y, z. In this chapter it will be denoted as,

$$q = [s, v] = [\cos(\theta/2),\ \sin(\theta/2) < k_x, k_y, k_z >] \tag{55}$$

where $s \in R$, $v \in R^3$ and θ and k, a rotation angle and unit axis, respectively. For a vector r oriented an angle θ about the vector k, there is a quaternion $q = [\cos(\theta/2), \sin(\theta/2) < k_x,\ k_y,\ k_z >] = [s,\ < x,\ y, z >]$ that represents the orientation. This is equivalent to the rotation matrix R.

$$R = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2sz & 2xz + 2sy \\ 2xy + 2sz & 1 - 2x^2 - 2z^2 & 2yz - 2sx \\ 2xz - 2sy & 2yz + 2sx & 1 - 2x^2 - 2y^2 \end{bmatrix} \tag{56}$$

If R is equated to a 3x3 rotational matrix given by

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{57}$$

and since, q is unit magnitude ($s^2 + x^2 + y^2 + z^2 = 1$) then, the rotation matrix R can be mapped to a quaternion $q = [s,\ < x,\ y, z >]$ as follows.

$$s = \frac{\sqrt{r_{11} + r_{22} + r_{33} + 1}}{2} \tag{58}$$

$$x = \frac{r_{32} - r_{23}}{4s} \tag{59}$$

$$y = \frac{r_{13} - r_{31}}{4s} \tag{60}$$

$$z = \frac{r_{21} - r_{12}}{4s} \tag{61}$$

Although unit quaternions are very suitable for defining the orientation of a rigid body, they do not contain any information about its position in the 3D space. The way to represent both rotation and translation in a single transformation vector is to use dual quaternions. The transformation vector using dual quaternions for a revolute joint is

$$Q(q,p) = ([\cos(\theta/2), \sin(\theta/2) < k_x, k_y, k_z >], < p_x, p_y, p_z > \tag{62}$$

where the unit quaternion q represents appropriate rotation and the vector $p = <p_x, p_y, p_z>$ encodes corresponding translational displacement. In the case of prismatic joints, the displacement is represented by a quaternion-vector pair as follows.

$$Q(q,p) = ([1, < 0, 0, 0 >], < p_x, p_y, p_z >) \tag{63}$$

where $[1, < 0, 0, 0 >]$ represents unit identity quaternion. Quaternion multiplication is vital to combining the rotations. Let, $q_1 = [s_1, v_1]$ and $q_2 = [s_2, v_2]$ denote two unit quaternions. In this case, multiplication process is shown as

$$q_1 * q_2 = [s_1 s_2 - v_1 \cdot v_2, \ s_1 v_2 + s_2 v_1 + v_1 \times v_2] \tag{64}$$

where $(.)$, $(\times)$ and $(*)$ are dot product, cross product and quaternion multiplication, respectively. In the same manner, the quaternion multiplication of two point vector transformations is denoted as,

$$Q_1 Q_2 = (q_1, p_1) * (q_2, p_2) = q_1 * q_2, \ q_1 * p_2 * q_1^{-1} + p_1 \tag{65}$$

where, $q_1 * p_2 * q_1^{-1} = p_2 + 2s_1(v_1 \times p_2) + 2v_1 \times (v_1 \times p_2)$. A unit quaternion inverse requires only negating its vector part, i.e.

$$q^{-1} = [s, \ v] = [s, \ -v] \qquad (66)$$

Finally, an equivalent expression for the inverse of a quaternion-vector pair can be written as,

$$Q^{-1} = (q^{-1}, \ -q^{-1} * p * q) \qquad (67)$$

where $-q^{-1} * p * q = -p + [-2s(v \times (-p)) + 2v \times (v \times (-p))]$.

3.2 Forward Kinematics

Based on the quaternion modeling convention, the forward kinematics vector transformation for an open kinematics chain can be derived as follows: Consider the Stanford Manipulator once more as illustrated in Figure 9. A coordinate frame is affixed to the base of the manipulator arbitrarily and the z-axis of the frame is assigned for pointing along the rotation axis of first joint. This frame does not move and, is considered as the reference coordinate frame. The position and the orientation vectors of all other joints are assigned in terms of this frame. Let's find orientation vectors. Since the z-axis of the reference coordinate frame is the unit line vector along the rotation axis of the first joint, the quaternion vector that represents the orientation is expressed as

$$q_1 = [\cos \overline{\theta}_1, \ \sin \overline{\theta}_1 < 0, 0, 1 >] \qquad (68)$$

where $\overline{\theta}_1 = \theta_1 / 2$. As shown in Figure 9, the unit line vector of the second joint is the opposite direction of the y-axis of the reference coordinate frame, in this case, the orientation of the second joint is given by
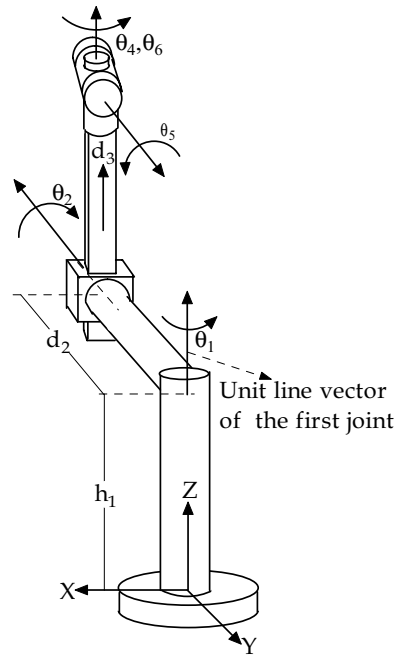
Figure 9. The coordinate frame and unit line vectors for the Stanford Manipulator.

$$q_2 = [\cos\overline{\theta}_2, \ \sin\overline{\theta}_2 < 0, -1, 0 >] \tag{69}$$

Because, the third joint is prismatic; there is only a unit identity quaternion that can be denoted as

$$q_3 = [1, \ < 0, 0, 0 >] \tag{70}$$

Orientations of the last three joints are determined as follows using the same approach described above.

$$q_4 = [\cos\overline{\theta}_4, \ \sin\overline{\theta}_4 < 0, 0, 1 >] \tag{71}$$
$$q_5 = [\cos\overline{\theta}_5, \ \sin\overline{\theta}_5 < 0, 1, 0 >] \tag{72}$$

$$q_6 = [\cos\overline{\theta}_6, \ \sin\overline{\theta}_6 < 0, 0, 1 >] \tag{73}$$

The position vectors are assigned in terms of reference coordinate frame as follows. When the first joint is rotated anticlockwise direction around the z axis of reference coordinate frame by an angle of θ_1, the link d_2 traces a circle in the xy-plane which is parallel to the xy plane of the reference coordinate frame as given in Figure 10a. Any point on the circle can be determined using the vector

$$p_1 = <p_{x1}, p_{y1}, p_{z1}> = <d_2 \sin\theta_1, -d_2 \cos\theta_1, h_1> \tag{74}$$

If the second joint is rotated as in Figure 10b, in this case the rotation will be xz-plane with respect to the reference coordinate frame. The position vector of the second quaternion can be written as

$$p_2 = <-d_3 \sin\theta_2, 0, d_3 \cos\theta_2> \tag{75}$$
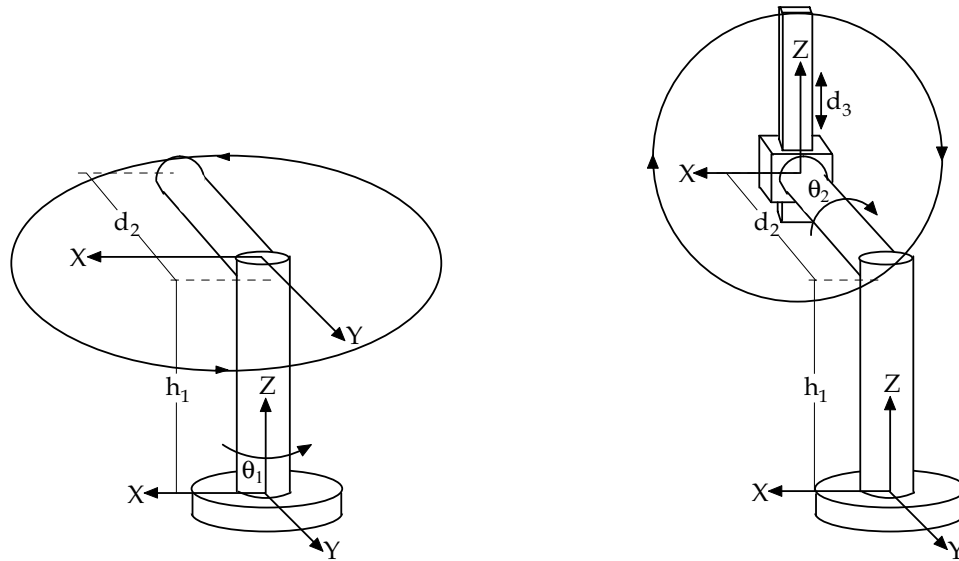
Figure 10. a) The link d_2 traces a circle on the xy-plane; b) The link d_3 traces a circle on the xz-plane.

Since rotation of the last four joints do not create any displacement for the successive joints, the position vectors are denoted as

$$p_3 = p_4 = p_5 = p_6 = <0, 0, 0> \tag{76}$$

Finally, the kinematics transformations for the Stanford Manipulator defining the spatial relationships between successive linkages can be expressed as follows.

$$Q_1 = ([\cos\overline{\theta}_1, \ \sin\overline{\theta}_1 <0, 0, 1>], <d_2 \sin\theta_1, -d_2 \cos\theta_1, h_1>) \tag{77}$$

$$Q_2 = ([\cos\overline{\theta}_2, \ \sin\overline{\theta}_2 <0, -1, 0>], <-d_3 \sin\theta_2, 0, d_3 \cos\theta_2>) \tag{78}$$

$$Q_3 = ([1, \ <0, 0, 0>], <0,0,0>) \tag{79}$$

$$Q_4 = \left([\cos \bar{\theta}_4, \ \sin \bar{\theta}_4 < 0, 0, 1 >], < 0, 0, 0 > \right) \tag{80}$$

$$Q_5 = \left([\cos \bar{\theta}_5, \ \sin \bar{\theta}_5 < 0, 1, 0 >], < 0, 0, 0 > \right) \tag{81}$$

$$Q_6 = \left([\cos \bar{\theta}_6, \ \sin \bar{\theta}_6 < 0, 0, 1 >], < 0, 0, 0 > \right) \tag{82}$$

The forward kinematics of the Stanford Manipulator can be determined in the form of equation 62, multiplying all of the Q_i matrices, where i=1,2, …, 6.

$$Q(q,p) = \left([s, v], < d_2 s \theta_1 - d_3 c \theta_1 s \theta_2, \ - d_2 c \theta_1 - d_3 s \theta_1 s \theta_2, \ h_1 + d_3 c \theta_2 > \right) \tag{83}$$

where $s = M_{11}$ and $v = < M_{12}, M_{13}, M_{14} >$ are given by equation 98.

3.3 Inverse Kinematics

To solve the inverse kinematics problem, the transformation quaternion is defined as

$$[R_w, T_w] = ([w, < a, b, c >], < p_x, p_y, p_z >) \tag{84}$$

where (R_w, T_w) represents the known orientation and translation of the robot end-effector with respect to the base. Let $Q_i \ (1 \le i \le 6)$ denotes kinematics transformations describing the spatial relationships between successive coordinate frames along the manipulator linkages such as $Q_1 = (q_1, p_1)$, $Q_2 = (q_2, p_2)$ … $Q_6 = (q_6, p_6)$.

The quaternion vector products M_i and the quaternion vector pairs N_{j+1} are defined as

$$M_i = Q_i M_{i+1} \tag{85}$$
$$N_{i+1} = Q_i^{-1} N_i \tag{86}$$

where $1 \le i \le 5$. Note that $M_6 = Q_6$ and $N_1 = [R_w, T_w]$. In order to extract joint variables as functions of s, v, p_x, p_y, p_z and fixed link parameters, appropriate M_i and N_{j+1} terms are equated, such as $M_1 = N_1$, $M_2 = N_2$ … $M_6 = N_6$. For the reason of compactness, $\theta_i / 2$, $\sin(\theta_i / 2)$, $\cos(\theta_i / 2)$, $\sin(\theta_i)$ and $\cos(\theta_i)$ will be represented as $\bar{\theta}_i$, $\bar{s}_i$, $\bar{c}_i$, s_i and c_i respectively. The link transformation matrices were formerly developed. The inverse transformations can be written as follows using equation 67.

$$Q_1^{-1} = ([\bar{c}_1, -\bar{s}_1 < 0,\ 0,\ 1 >], < 0, - d_2, - h_1 >) \tag{87}$$

$$Q_2^{-1} = ([\bar{c}_2, \bar{s}_2 < 0,\ 1,\ 0 >], < 0, 0, - d_3 >) \tag{88}$$

$$Q_3^{-1} = ([1, < 0,\ 0,\ 0 >], < 0,0,0 >) \tag{89}$$

$$Q_4^{-1} = ([\bar{c}_4, -\bar{s}_4 < 0,\ 0,\ 1 >], < 0,0,0 >) \tag{90}$$

$$Q_5^{-1} = ([\bar{c}_5, -\bar{s}_5 < 0,\ 1,\ 0 >], < 0,0,0 >) \tag{91}$$

$$Q_6^{-1} = ([\bar{c}_6, -\bar{s}_6 < 0,\ 0,\ 1 >], < 0,0,0 >) \tag{92}$$

The quaternion vector products are

$$M_6 = Q_6 = ([c\bar{\theta}_6,\ s\bar{\theta}_6 < 0,\ 0,\ 1 >], < 0,\ 0,\ 0 >) \tag{93}$$

$$M_5 = Q_5 M_6 = ([\bar{c}_5 \bar{c}_6, < \bar{s}_5 \bar{s}_6,\ \bar{s}_5 \bar{c}_6,\ \bar{c}_5 \bar{s}_6 >], < 0,0,0 >) \tag{94}$$

$$M_4 = Q_4 M_5 = ([M_{41}, < M_{42}, M_{43}, M_{44} >], < 0,0,0 >) \tag{95}$$

where,

$$M_{41} = \bar{c}_5 \bar{c}_{(4+6)},$$
$$M_{42} = -\bar{s}_5 \bar{s}_{(4-6)},$$
$$M_{43} = \bar{s}_5 \bar{c}_{(4-6)} \text{ and}$$
$$M_{44} = \bar{c}_5 \bar{s}_{(6+4)}.$$

$$M_3 = Q_3 M_4 = ([M_{31}, < M_{32}, M_{33}, M_{34} >], < 0,0,0 >) \tag{96}$$

where,

$$M_{32} = M_{42},$$
$$M_{33} = M_{43} \text{ and}$$
$$M_{34} = M_{44}.$$

$$M_2 = Q_2 M_3 = ([M_{21}, < M_{22}, M_{23}, M_{24} >], < -d_3 s_2,\ 0,\ d_3 c_2 >) \tag{97}$$

where,

$M_{21} = \overline{c}_2 M_{41} - \overline{s}_2 M_{43}$,

$M_{22} = \overline{c}_2 M_{42} - \overline{s}_2 M_{44}$, $M_{23} = \overline{c}_2 M_{43} - \overline{s}_2 M_{41}$ and

$M_{24} = \overline{c}_2 M_{44} + \overline{s}_2 M_{42}$.

$$M_1 = Q_1 M_2 = ([M_{11}, < M_{12}, M_{13}, M_{14} >], < M_{15}, M_{16}, M_{17} >) \qquad (98)$$

where,

$M_{11} = \overline{c}_1 M_{21} - \overline{s}_1 (\overline{c}_2 M_{44} - \overline{s}_2 M_{42})$,

$M_{12} = \overline{c}_1 M_{22} - \overline{s}_1 M_{23}$,

$M_{13} = \overline{c}_1 M_{23} + \overline{s}_1 M_{22}$,

$M_{14} = \overline{s}_1 M_{21} + \overline{c}_1 M_{24}$,

$M_{15} = d_2 s\theta_1 - d_3 c\theta_1 s\theta_2$,

$M_{16} = -d_2 c\theta_1 - d_3 s\theta_1 s\theta_2$ and

$M_{17} = h_1 + d_3 c\theta_2$.

The quaternion vector pairs are

$$N_1 = ([w, < a, b, c >] < p_x, p_y, p_z >) \qquad (99)$$

$$N_2 = Q_1^{-1} N_1 = ([N_{21}, < N_{22}, N_{23}, N_{24} >], < N_{25}, N_{26}, p_z - h_1 >) \qquad (100)$$

where,

$N_{21} = w\overline{c}_1 + c\overline{s}_1$,

$N_{22} = a\overline{c}_1 + b\overline{s}_1$,

$N_{23} = b\overline{c}_1 - a\overline{s}_1$,

$N_{24} = c\overline{c}_1 - w\overline{s}_1$,

$N_{25} = p_x c_1 + p_y s_1$ and

$N_{26} = -p_x s_1 + p_y c_1 - d_2$.

$$N_3 = Q_2^{-1} N_2 = ([N_{31}, < N_{32}, N_{33}, N_{34} >], < N_{35}, N_{36}, N_{37} >) \qquad (101)$$

where,

$N_{31} = \overline{c}_2 N_{21} - \overline{s}_2 N_{24}$,

$N_{32} = \overline{c}_2 N_{22} - \overline{s}_2 N_{23}$,

$N_{33} = \overline{c}_2 N_{23} + \overline{s}_2 N_{22}$,

$N_{34} = \overline{c}_2 N_{24} + \overline{s}_2 N_{21}$,

$N_{35} = c_2 (p_x c_1 + p_y s_1) - s_2 (h_1 - p_z)$, $N_{36} = p_y c_1 - p_x s_1 - d_2$,

$N_{37} = c_2 (p_z - h_1) - s_2 (p_x c_1 + p_y s_1) - d_3$.

$$N_4 = Q_3^{-1} N_3 = ([N_{41}, < N_{42}, N_{43}, N_{44} >], < 0, 0, 0 >) \tag{102}$$

where, $N_{41} = N_{31}$,

$N_{42} = N_{32}$,

$N_{43} = N_{33}$ and

$N_{44} = N_{34}$.

The first joint variable θ_1 can be determined equating the second terms of M_2 and N_2 as follows.

$$\theta_1 = A\tan 2(p_x, -p_y) \pm A\tan 2(\sqrt{p_x^2 + p_y^2 - d_2^2}, d_2) \tag{103}$$

The joint variables θ_2 and d_3 are computed equating the first and third elements of M_3 and N_3 respectively.

$$\theta_2 = \pm A\tan 2(c\theta_1 p_x + s\theta_1 p_y, h_1 - p_z) \tag{104}$$

$$d_3 = -s\theta_2(c\theta_1 p_x + s\theta_1 p_y) + c\theta_2(p_z - h_1) \tag{105}$$

$\bar{s}_5^2 = N_{42}^2 + N_{43}^2$, $\bar{c}_5^2 = N_{41}^2 + N_{44}^2$, $\tan(\bar{\theta}_4 + \bar{\theta}_6) = \dfrac{N_{44}}{N_{41}}$ and $\tan(\bar{\theta}_4 - \bar{\theta}_6) = -\dfrac{N_{42}}{N_{43}}$

equations can be derived form equating the terms M_4 to N_4, where, $\bar{c}_5\bar{c}_{(4+6)} = N_{41}$, $-\bar{s}_5\bar{s}_{(4-6)} = N_{42}$, $\bar{s}_5\bar{c}_{(4-6)} = N_{43}$ and $\bar{c}_5\bar{s}_{(6+4)} = N_{44}$. In this case, the orientation angles of the Euler wrist can be determined as follows.

$$\theta_5 = \arctan 2\left(\pm\sqrt{N_{42}^2 + N_{43}^2}, \sqrt{N_{41}^2 + N_{44}^2}\right) \tag{106}$$

$$\theta_4 = \arctan\left(\frac{N_{44}}{N_{41}}\right) + \arctan\left(-\frac{N_{42}}{N_{43}}\right) \tag{107}$$

$$\theta_6 = \arctan\left(\frac{N_{44}}{N_{41}}\right) - \arctan\left(-\frac{N_{42}}{N_{43}}\right) \tag{108}$$

4. References

Denavit, J. & Hartenberg, R. S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics,* Vol., 1 (June 1955) pp. 215-221

Funda, J.; Taylor, R. H. & Paul, R.P. (1990). On homogeneous transorms, quaternions, and computational efficiency. *IEEE Trans.Robot. Automat.,* Vol., 6 (June 1990) pp. 382–388

Kucuk, S. & Bingul, Z. (2004). The Inverse Kinematics Solutions of Industrial Robot Manipulators, *IEEE Conferance on Mechatronics*, pp. 274-279, Turkey, June 2004, Istanbul

Craig, J. J. (1989). *Introduction to Robotics Mechanics and Control,* USA:Addision-Wesley Publishing Company

Hamilton, W. R. (1869). Elements of quaternions, Vol., I & II, Newyork Chelsea

Salamin, E. (1979). *Application of quaternions to computation with rotations.* Tech., AI Lab, Stanford Univ., 1979

Kotelnikov, A. P. (1895). *Screw calculus and some of its applications to geometry and mechanics.* Annals of the Imperial University of Kazan.

Pervin, E. & Webb, J. A. (1983). Quaternions for computer vision and robotics, *In conference on computer vision and pattern recognition.* pp 382-383, Washington, D.C.

Gu, Y.L. & Luh, J. (1987). Dual-number transformation and its application to robotics. *IEEE J. Robot. Automat.* Vol., 3, pp. 615-623

Kim, J. H. & Kumar, V. R. (1990). Kinematics of robot manipulators via line transformations. *J. Robot. Syst.,* Vol., 7, No., 4, pp. 649–674

Caccavale, F. & Siciliano, B. (2001). Quaternion-based kinematic control of redundant spacecraft/ manipulator systems, *In proceedings of the 2001 IEEE international conference on robotics and automation*, pp. 435-440

Rueda, M. A. P.; Lara, A. L. ; Marinero, J. C. F.; Urrecho, J. D. & Sanchez, J.L.G. (2002). Manipulator kinematic error model in a calibration process through quaternion-vector pairs, *In proceedings of the 2002 IEEE international conference on robotics and automation*, pp. 135-140

Structure Based Classification and Kinematic Analysis of Six-Joint Industrial Robotic Manipulators

Tuna Balkan, M. Kemal Özgören and M. A. Sahir Arıkan

1. Introduction

In this chapter, a complete set of compact, structure based generalized kinematic equations for six-joint industrial robotic manipulators are presented together with their sample solutions. Industrial robots are classified according to their kinematic structures, and their forward kinematic equations are derived according to this classification. The purpose of this classification is to obtain simplified forward kinematic equations considering the specific features of the classified manipulators and thus facilitate their inverse kinematic solutions. For the classification, one hundred industrial robots are surveyed. The robots are first classified into kinematic main groups and then into subgroups under each main group. The main groups are based on the end-effector rotation matrices and characterized by the twist angles. On the other hand, the subgroups are based on the wrist point positions and characterized by the link lengths and offsets. The reason for preferring the wrist point rather than the tip point in this classification is that, the wrist point and rotation matrix combination contain the same amount of information as the tip point and rotation matrix combination about the kinematic features of a manipulator, and the wrist point coordinates are simpler to express in terms of the joint variables. After obtaining the forward kinematic equations (i.e. the main group rotation matrix equations and the subgroup wrist point equations), they are simplified in order to obtain compact kinematic equations using the numerous properties of the exponential rotation matrices (Özgören, 1987-2002). The usage of the exponential rotation matrices provided important advantages so that simplifications are carried out in a systematic manner with a small number of symbolic matrix manipulations. Subsequently, an inverse kinematic solution approach applicable to the six-joint industrial robotic manipulators is introduced. The approach is based on the kinematic classification of the industrial robotic manipulators as explained above. In the inverse kinematic solutions of the surveyed industrial robots, most of the simplified compact equations can be solved analytically and the remaining few of them can be solved semi-analytically through a numerical solution of a single univariate equation. The semi-analytical method

is named as the *Parametrized Joint Variable (PJV)* method. In these solutions, the singularities and the multiple configurations of the manipulators indicated by sign options can be determined easily. Using these solutions, the inverse kinematics can also be computerized by means of short and fast algorithms. Owing to the properties of the exponential rotation matrices, the derived simple and compact equations are easy to implement for computer programming of the inverse kinematic solutions. Besides, the singularities and the multiple configurations together with the working space limitations of the manipulator can be detected readily before the programming stage, which enables the programmer to take the necessary actions while developing the program. Thus, during the inverse kinematic solution, it becomes possible to control the motion of the manipulator in the desired configuration by selecting the sign options properly. In this approach, although the derived equations are manipulator dependent, for a newly encountered manipulator or for a manipulator to be newly designed, there will be no need to follow the complete derivation procedure starting from the beginning for most of the cases; only a few modifications will be sufficient. These modifications can be addition or deletion of a term, or just changing simply a subscript of a link length or offset. Even if the manipulator under consideration happens to generate a new main group, the equations can still be derived without much difficulty by using the procedure described here, since the approach is systematic and its starting point is the application of the Denavit-Hartenberg convention by identifying the twist angles and the other kinematic parameters. In this context, see (Özgören, 2002) for an exhaustive study that covers all kinds of six-joint serial manipulators. The presented method is applicable not only for the serial manipulators but also for the hybrid manipulators with closed chains. This is demonstrated by applying the method to an ABB IRB2000 industrial robot, which has a four-bar mechanism for the actuation of its third link. Thus, alongside with the serial manipulators, this particular hybrid manipulator also appears in this chapter with its compact forward kinematic equations and their inversion for the joint variables. Finally, the chapter is closed by giving the solutions to some typical trigonometric equations encountered during the inverse kinematic solutions.

For the solution of inverse kinematics problem, forward kinematic equations are required. There are three methods for inverse kinematic solution; namely, analytical, semi-analytical, and fully numerical. Presently, analytical methods can be used only for certain manipulators with specific kinematic parameter combinations such as PUMA 560. For a general case where the manipulator does not have specific kinematic parameter combinations, it becomes impossible to obtain analytical solutions. So, either semi-analytical or fully numerical methods have been developed. Since the present general semi-analytical methods are rather cumbersome to use (Raghavan & Roth, 1993; Manseur & Doty, 1996), fully numerical methods are mostly preferred. However, if the forward kinematic equations can be simplified, it becomes feasible to use semi-

analytical and even analytical methods for a large number of present industrial robot types. On the other hand, although the fully numerical methods can detect the singularities by checking the determinant of the Jacobian matrix, they have to do this continuously during the solution, which slows down the process. However, the type of the singularity may not be distinguished. Also, in case of multiple solutions, the desired configurations of the manipulator can not be specified during the solution. Thus, in order to clarify the singularities and the multiple configurations, it becomes necessary to make use of semi-analytical or analytical methods. Furthermore, the analytical or semi-analytical methods would be of practical use if they lead to compact and simple equations to facilitate the detection of singularities and multiple configurations. The methodology presented in this chapter provides such simple and compact equations by making use of various properties of the exponential rotation matrices, and the simplification tools derived by using these properties (Özgören, 1987-2002). Since different manipulator types with different kinematic parameters lead to different sets of simplified equations, it becomes necessary to classify the industrial robotic manipulators for a systematic treatment. For such a classification, one hundred currently used industrial robots are surveyed (Balkan et al., 1999, 2001).

The kinematics of robotic manipulators can be dealt with more effectively and faster by perceiving their particular properties rather than resorting to generality (Hunt, 1986). After the classification, it is found that most of the recent, well-known robotic manipulators are within a specific main group, which means that, instead of general solutions and approaches, manipulator dependent solutions and approaches that will lead to easy specific solutions are more reasonable. The usage of exponential rotation matrices provide important advantages so that simplifications can be carried out in a systematic manner with a small number of symbolic matrix manipulations and the resulting kinematic equations become much simpler especially when the twist angles are either 0° or ± 90°, which is the case with the common industrial robots.

For serial manipulators, the forward kinematics problem, that is, determination of the end-effector position and orientation in the Cartesian space for given joint variables, can easily be solved in closed-form. Unfortunately, the inverse kinematics problem of determining each joint variable by using the Cartesian space data does not guarantee a closed-form solution. If a closed-form solution can not be obtained, then there are different types of approaches for the solution of this problem. The most common one is to use a completely numerical solution technique such as the Newton-Raphson algorithm. Another frequently used numerical method is the "resolved motion rate control" which uses the inverse of the Jacobian matrix to determine the rates of the joint variables and then integrates them numerically with a suitable method (Wu & Paul, 1982). Runge-Kutta of order four is a common approach used for this purpose. As an analytical approach, it is possible to convert the forward kine-

matic equations into a set of polynomial equations. Then, they can be reduced to a high-order single polynomial equation through some complicated algebraic manipulations. Finally, the resulting high-order equation is solved numerically. However, requiring a lot of polynomial manipulations, this approach is quite cumbersome (Wampler & Morgan, 1991; Raghavan & Roth, 1993).

On the other hand, the approach presented in this chapter aims at obtaining the inverse kinematic solutions analytically by manipulating the trigonometric equations directly without converting them necessarily into polynomial equations. In a case, where an analytical solution cannot be obtained this way, then a semi-analytical solution is aimed at by using the method described below.

As explained before, the PJV method is a semi-analytical inverse kinematics solution method which can be applied to different kinematic classes of six-joint manipulators which have no closed-form solutions. In most of the cases, it is based on choosing one of the joint variables as a parameter and determining the remaining joint variables analytically in terms of this parametrized joint variable. Parametrizing a suitable joint variable leads to a single univariate equation in terms of the parametrized joint variable only. Then, this equation is solved using a simple numerical technique and as the final step remaining five joint variables are easily computed by substituting the parametrized joint variable in their analytical expressions. However, for certain kinematic structures and kinematic parameters two and even three equations in three unknowns may arise (Özgören, 2002). Any initial value is suitable for the solution and computational time is very small even for an initial condition far from the solution. The PJV method can also handle the singular configurations and multiple solutions. However, it is manipulator dependent and equations are different for different classes of manipulators. PJV works well also for non-spherical wrists with any structural kinematic parameter combination.

In this chapter, four different subgroups are selected for the demonstration of the inverse kinematic solution method. Two of these subgroups are examples to closed-form and semi-analytic inverse kinematic solutions for the most frequently seen kinematic structures among the industrial robots surveyed in (Balkan et al., 1999, 2001). Since the manipulators in these two subgroups have revolute joints only, the inverse kinematic solution of subgroup 4.4 which includes Unimate 4000 industrial robot is also given to demonstrate the method on manipulators with prismatic joints. The inverse kinematic solution for this class of manipulators happens to be either closed-form or needs the PJV method depending on the selection of one of its parameters. In addition, the inverse kinematic solution for ABB IRB2000 industrial robot, which has a closed chain, is obtained to show the applicability of the method to such manipulators.

2. Kinematic Equations for Six-Joint Robots

In the derivation of the kinematic equations for six-joint manipulators, Denavit-Hartenberg (D-H) convention is used as shown in Figure 1 (Denavit & Hartenberg, 1955), with notation adopted from (Özgören, 2002).
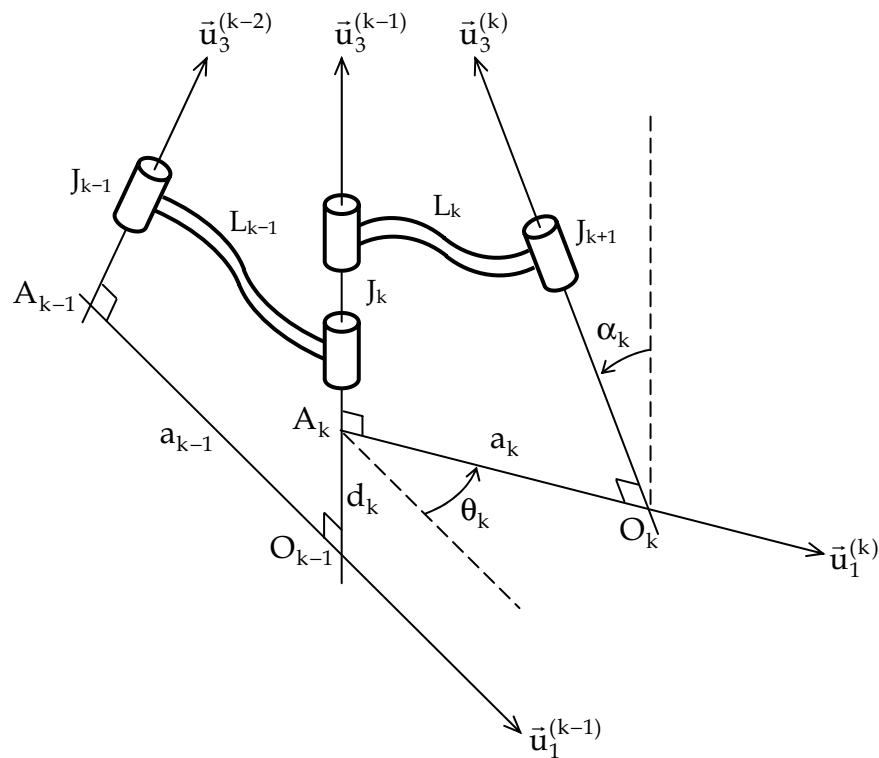
Figure 1. D-H Convention and Related Notation

The symbols in Fig. 1 are explained below.

J_k: Joint k.

L_k: Link k.

O_k: Origin of the reference frame F_k attached to L_k.

A_k: Auxiliary point between L_{k-1} and L_k.

$\vec{u}_i^{(k)}$: i^{th} unit basis vector of F_k ; i = 1, 2, 3.

a_k: Effective length A_kO_k of L_k along $\vec{u}_1^{\ k}$.

d_k: Distance $O_{k-1}A_k$ of L_k from L_{k-1} along $\vec{u}_3^{(k-1)}$. It is a constant parameter, called offset, if J_k is revolute. It is the kth joint variable if J_k is prismatic. It is then denoted as s_k.

θ_k: Rotation angle of L_k with respect to L_{k-1} about $\vec{u}_3^{(k-1)}$. It is the kth joint variable if J_k is revolute. If J_k is prismatic, it is a constant parameter which is either $0°$ or $±90°$ for common industrial robot manipulators.

α_k: Twist angle of J_{k+1} with respect to J_k about $\vec{u}_1^{(k)}$. For common industrial robot manipulators, it is either $0°$ or $±90°$.

Among the industrial robots surveyed in this chapter, there is no industrial robot whose last joint is prismatic. Thus, the wrist point, which is defined as the origin of F_6 is chosen to be coincident with the origin of F_5. That is, $O_5 = O_6$. The other features of the hand frame F_6 are defined as described below.

$$\vec{u}_3^{(6)} = \vec{u}_3^{(5)} \tag{1}$$

$$a_6 = 0, \; d_6 = 0, \; \alpha_6 = 0 \tag{2}$$

The end-effector is fixed in F_6 and assuming that its tip point P is on the axis along the approach vector $\vec{u}_3^{(6)}$, its location can be described as $d_p = O_6P$.
The relationship between the representations of the same vector in two different frames can be written as shown below.

$$\bar{n}^{(a)} = \hat{C}^{(a,b)} \, \bar{n}^{(b)} \tag{3}$$

Here, $\bar{n}^{(a)}, \bar{n}^{(b)}$ are the column representations of the vector $\bar{n}$ in the frames F_a and F_b while $\hat{C}^{(a,b)}$ is the transformation matrix between these two frames.
In order to make the kinematic features of the manipulators directly visible and to make the due simplifications easily, the hand-to-base transformation matrix $\hat{C}^{(0,6)}$ and the wrist point position vector $\bar{r}^{(0)}$, or the tip point position vector $\bar{p}^{(0)}$ are expressed separately, rather than concealing the kinematic features into the overcompact homogeneous transformation matrices, which are also unsuitable for symbolic manipulations. The wrist and tip point position vectors are related as follows:

$$\bar{p}^{(0)} = \bar{r}^{(0)} + d_p \hat{C}^{(0,6)} \bar{u}_3 \tag{4}$$

Here, $\bar{r}^{(0)}$ and $\bar{p}^{(0)}$ are the column matrix representations of the position vectors in the base frame F_0 whereas $\bar{u}_3$ is the column matrix representation of the approach vector in the hand frame F_6.
The overall relative displacement from F_{k-1} to F_k consists of two rotations and two translations, which are sequenced as a translation of s_k along $\vec{u}_3^{(k-1)}$, a rotation of θ_k about $\vec{u}_3^{(k-1)}$, a translation of a_k along $\vec{u}_1^{(k)}$, and a rotation of α_k about $\vec{u}_1^{(k)}$.

Using the link-to-link rotational transformation matrices, $\hat{C}^{(0,6)}$ can be formulated as follows:

$$\hat{C}^{(0,6)} = \hat{C}^{(0,1)} \; \hat{C}^{(1,2)} \; \hat{C}^{(2,3)} \; \hat{C}^{(3,4)} \; \hat{C}^{(4,5)} \; \hat{C}^{(5,6)} \tag{5}$$

According to the D-H convention, the transformation matrix between two successive link frames can be expressed using exponential rotation matrices (Özgören, 1987-2002). That is,

$$\hat{C}^{(k-1,k)} = e^{\tilde{u}_3 \theta_k} \; e^{\tilde{u}_1 \alpha_k} \tag{6}$$

On the other hand, assuming that frame F_b is obtained by rotating frame F_a about an axis described by a unit vector $\bar{n}$ through an angle θ, the matrix $\hat{C}^{(a,b)}$ is given as an exponential rotation matrix by the following equation (Özgören, 1987-2002):

$$\hat{C}^{(a,b)} = e^{\tilde{n}\theta} = \hat{I} \cos\theta + \tilde{n} \sin\theta + \bar{n} \, \bar{n}^T \, (1-\cos\theta) \tag{7}$$

Here, $\hat{I}$ is the identity matrix and $\tilde{n}$ is the skew symmetric matrix generated from the column matrix $\bar{n} = \bar{n}^{(a)}$. This generation can be described as follows.

$$\bar{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \quad \rightarrow \quad \tilde{n} = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \tag{8}$$

Furthermore, if $\bar{n} = \bar{u}_k^{(a)}$ where $\bar{u}_k^{(a)}$ is the kth basis vector of the frame F_a, then $\bar{n} = \bar{u}_k$ and

$$\hat{C}^{(a,b)} = e^{\tilde{u}_k \theta} \tag{9}$$

Here,

$$\bar{u}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \bar{u}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \bar{u}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{10}$$

Using Equation (6), Equation (5) can be written as

$$\hat{C} = \hat{C}^{(0,6)} = e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_1 \alpha_1} e^{\tilde{u}_3 \theta_2} e^{\tilde{u}_1 \alpha_2} e^{\tilde{u}_3 \theta_3} e^{\tilde{u}_1 \alpha_3} e^{\tilde{u}_3 \theta_4} e^{\tilde{u}_1 \alpha_4} e^{\tilde{u}_3 \theta_5} e^{\tilde{u}_1 \alpha_5} e^{\tilde{u}_3 \theta_6} \tag{11}$$

On the other hand, the wrist point position vector can be expressed as

$$\vec{r} = \vec{r}_{01} + \vec{r}_{12} + \vec{r}_{23} + \vec{r}_{34} + \vec{r}_{45} + \vec{r}_{56} \tag{12}$$

Here, $\vec{r}_{ij}$ is the vector from the origin O_i to the origin O_j.

Using the column matrix representations of the vectors in the base frame F_0, Equation (12) can be written as

$$\bar{r} = \bar{r}^{(0)} = d_1\bar{u}_3 + a_1\hat{C}^{(0,1)}\bar{u}_1 + d_2\hat{C}^{(0,1)}\bar{u}_3 + a_2\hat{C}^{(0,2)}\bar{u}_1 + d_3\hat{C}^{(0,2)}\bar{u}_3 + a_3\hat{C}^{(0,3)}\bar{u}_1$$
$$+ d_4\hat{C}^{(0,3)}\bar{u}_3 + a_4\hat{C}^{(0,4)}\bar{u}_1 + d_5\hat{C}^{(0,4)}\bar{u}_3 + a_5\hat{C}^{(0,5)}\bar{u}_1 \tag{13}$$

Substitution of the rotational transformation matrices and manipulations using the exponential rotation matrix simplification tool E.2 (Appendix A) result in the following simplified wrist point equation in its most general form.

$$\bar{r} = d_1\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + d_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_1\alpha_1}\bar{u}_3 + a_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_1\alpha_1}e^{\tilde{u}_3\theta_2}\bar{u}_1$$
$$+ d_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_1\alpha_1}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_1\alpha_2}\bar{u}_3 + a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_1\alpha_1}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_1\alpha_2}e^{\tilde{u}_3\theta_3}\bar{u}_1$$
$$+ d_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_1\alpha_1}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_1\alpha_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_1\alpha_3}\bar{u}_3$$
$$+ a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_1\alpha_1}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_1\alpha_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_1\alpha_3}e^{\tilde{u}_3\theta_4}\bar{u}_1$$
$$+ d_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_1\alpha_1}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_1\alpha_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_1\alpha_3}e^{\tilde{u}_3\theta_4}e^{\tilde{u}_1\alpha_4}\bar{u}_3$$
$$+ a_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_1\alpha_1}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_1\alpha_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_1\alpha_3}e^{\tilde{u}_3\theta_4}e^{\tilde{u}_1\alpha_4}e^{\tilde{u}_3\theta_5}\bar{u}_1 \tag{14}$$

3. Classification of Six-Joint Industrial Robotic Manipulators

As noticed in Equations (11) and (14), the general $\bar{r}$ expression contains five joint variables and the general $\hat{C}$ expression includes all of the angular joint variables. On the other hand, it is an observed fact that in the six-joint industrial robots, many of the structural length parameters (a_k and d_k) are zero (Balkan et al., 1999, 2001). Due to this reason, there is no need to handle the inverse kinematics problem in a general manner. Instead, the zero values of a_k and d_k of these robots can be used to achieve further simplifications in Equations (11) and (14). In order to categorize and handle the simplified equations in a systematic manner, the industrial robots are grouped using a two step classification scheme according to their structural parameters a_k, α_k, and d_k for revolute joints or θ_k for prismatic joints. The primary classification is based on the twist angles (α_k) and it gives the *main groups*. Whereas, the secondary classification is based on the other structural parameters (a_k and d_k or θ_k) and it gives the *subgroups* under each main group.

In the main groups, the simplified $\bar{r}$ and $\hat{C}$ expressions are obtained using the fact that the twist angles are either $0°$ or $\pm 90°$. The $\hat{C}$ expression for each main group is the same, because the rotation angles (θ_k) are not yet distinguished at this level whether they are constant or not. At the level of the subgroups, the values of the twist and constant rotation angles are substituted into the $\bar{r}$ and $\hat{C}$ expressions, together with the other parameters. Then, the properties of the exponential rotation matrices are used in order to obtain simplified equations with reduced number of terms, which can be used with convenience for the inverse kinematic solutions. The main groups with their twist angles and the number of robots in each main group are given in Table 1 considering the industrial robots surveyed here. The subgroups are used for finer classification using the other structural parameters. For the manipulators in this classification, the $\bar{r}$ expressions are simplified to a large extent especially when zeros are substituted for the vanishing structural parameters.

Main Group	α_1	α_2	α_3	α_4	α_5	α_6	Number of Robots
1	$-90°$	$0°$	$90°$	$-90°$	$90°$	$0°$	73
2	$-90°$	$0°$	$0°$	$90°$	$-90°$	$0°$	12
3	$-90°$	$90°$	$0°$	$-90°$	$90°$	$0°$	5
4	$-90°$	$90°$	$-90°$	$90°$	$-90°$	$0°$	4
5	$0°$	$0°$	$0°$	$-90°$	$90°$	$0°$	1
6	$-90°$	$90°$	$0°$	$0°$	$-90°$	$0°$	1
7	$0°$	$-90°$	$0°$	$90°$	$-90°$	$0°$	1
8	$0°$	$-90°$	$90°$	$-90°$	$90°$	$0°$	2
9	$-90°$	$0°$	$90°$	$0°$	$-90°$	$0°$	1

Table 1. Main Groups of Surveyed Six-Joint Industrial Robots

3.1 Main Group Equations

Substituting all the nine sets of the twist angle values given in Table 1 into Equations (11) and (14), the main group equations are obtained. The terms of $\bar{r}$ involving a_k and d_k are denoted as $T(a_k)$ and $T(d_k)$ as described below.

$$T(a_k) = a_k\, e\, (\theta_1, \dots, \theta_k, \alpha_1, \dots, \alpha_k)\, \bar{u}_1 \tag{15}$$

$$T(d_k) = d_k\, e\, (\theta_1, \dots, \theta_{k-1}, \alpha_1, \dots, \alpha_{k-1})\, \bar{u}_3 \tag{16}$$

Here, e stands for a product of exponential rotation matrices associated with the indicated angular arguments as exemplified by the following terms.

$$T(a_2) = a_2 e(\theta_1, \theta_2, \alpha_1)\bar{u}_1 = a_2 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_1\alpha_1} e^{\tilde{u}_3\theta_2} \bar{u}_1 \tag{17}$$

$$T(d_2) = d_2 e(\theta_1, \alpha_1)\bar{u}_3 = d_2 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_1\alpha_1}\bar{u}_3 \tag{18}$$

Here, the derivation of equations is given only for the main group 1, but the equations of the other groups can be obtained in a similar systematic manner by applying the *exponential rotation matrix simplification tools* given in Appendix A. The numbers (E.#) of the employed tools of Appendix A during the derivation of the $\hat{C}$ matrices and the terms (a_k) and (d_k) are shown in Table 2.

Main Group	$\hat{C}$	d_2	a_2	d_3	a_3	d_4	a_4	d_5	a_5
1	4, 6	8	10, 2, 6	2, 8	4, 6	4, 6	4, 6	4, 6, 8	4, 10, 2, 6
2	4, 6	8	10, 2, 6	2, 8	4, 10, 2, 6	4, 2, 8	4, 10, 2, 6	4, 6	4, 6
3	4, 6	8	10, 2, 6	6	6	2, 6	4, 6	4, 6, 8	4, 10, 2, 6
4	4, 6	8	10, 2, 6	6	6	6, 8	10, 2, 6	6	6
5	4, 6	2	4	4, 2	4	4, 2	4	4, 8	4, 10, 2, 6
6	4, 6	8	10, 2, 6	6	6	2, 6	4, 6	4, 2, 6	4, 6
7	4, 6	2	4	4, 8	4, 10, 2, 6	2, 4, 8	4, 10, 2, 6	4, 6	4, 6
8	4, 6	2	4	4, 8	4, 10, 2, 6	4, 6	4, 6	4, 6, 8	4, 10, 2, 6
9	4, 6	8	10, 2, 6	2, 8	4, 10, 2, 6	4, 6	4, 6	2, 4, 6	4, 6

Table 2. Exponential Rotation Matrix Simplification Tool Numbers (E.#) Applied for Derivation of $\hat{C}$ Matrices and Terms (a_k) and (d_k) in Main Group Equations

Equations of Main Group 1
Let $\bar{\alpha}$ denote the set of twist angles. For the main group 1, $\bar{\alpha}$ is

$$\bar{\alpha} = [-90°, \ 0°, \ 90°, \ -90°, \ 90° , 0°]^T. \tag{19}$$

Substituting $\bar{\alpha}$ into the general $\hat{C}$ equation results in the following equation.

$$\hat{C} = e^{\tilde{u}_3\theta_1} e^{-\tilde{u}_1\pi/2} e^{\tilde{u}_3\theta_2} e^{\tilde{u}_3\theta_3} e^{\tilde{u}_1\pi/2} e^{\tilde{u}_3\theta_4} e^{-\tilde{u}_1\pi/2} e^{\tilde{u}_3\theta_5} e^{\tilde{u}_1\pi/2} e^{\tilde{u}_3\theta_6} \tag{20}$$

Using the exponential rotation matrix simplification tools E.4 and E.6, the rotation matrix for the main group 1, i.e. $\hat{C}_1$, can be obtained as follows.

$$\hat{C}_1 = e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_{23}} e^{\tilde{u}_3\theta_4} e^{\tilde{u}_2\theta_5} e^{\tilde{u}_3\theta_6} \tag{21}$$

Here, $\theta_{jk} = \theta_j + \theta_k$ is used as a general way to denote joint angle combinations.

Substituting $\bar{\alpha}$ into the general $\bar{r}$ expression results in the following equation.

$$\bar{r} = d_1\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + d_2 e^{\tilde{u}_3\theta_1}e^{-\tilde{u}_1\pi/2}\bar{u}_3 + a_2 e^{\tilde{u}_3\theta_1}e^{-\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_2}\bar{u}_1$$

$$+ d_3 e^{\tilde{u}_3\theta_1}\ e^{-\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_2}\bar{u}_3 + a_3 e^{\tilde{u}_3\theta_1}e^{-\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_3\theta_3}\bar{u}_1$$

$$+ d_4 e^{\tilde{u}_3\theta_1}e^{-\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_1\pi/2}\bar{u}_3 + a_4 e^{\tilde{u}_3\theta_1}e^{-\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_4}\bar{u}_1$$

$$+ d_5 e^{\tilde{u}_3\theta_1}\ e^{-\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_4}e^{-\tilde{u}_1\pi/2}\bar{u}_3$$

$$+ a_5 e^{\tilde{u}_3\theta_1}e^{-\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_4}e^{-\tilde{u}_1\pi/2}e^{\tilde{u}_3\theta_5}\bar{u}_1 \tag{22}$$

The simplifications can be made for the terms $T(a_k)$ and $T(d_k)$ of Equation (22) as shown in Table 3 using the indicated simplification tools given in Appendix A.

E.8	$T(d_2) = d_2 e^{\tilde{u}_3\theta_1}\ \bar{u}_2$
E.10, E.6 and E.2	$T(a_2) = a_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1$
E.2 and E.8	$T(d_3) = d_3 e^{\tilde{u}_3\theta_1}\bar{u}_2$
E.4 and E.6	$T(a_3) = a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}\bar{u}_1$
E.4 and E.6	$T(d_4) = d_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}\bar{u}_3$
E.4 and E.6	$T(a_4) = a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}\bar{u}_1$
E.4, E.6 and E.8	$T(d_5) = d_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}\bar{u}_2$
E.4, E.6 and E.10	$T(a_5) = a_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}e^{\tilde{u}_2\theta_5}\bar{u}_1$

Table 3. Simplifications of the terms $T(a_k)$ and $T(d_k)$ in Equation (22)

Replacing the terms $T(a_k)$ and $T(d_k)$ in Equation (22) with their simplified forms given in Table 3, the wrist point location for the main group 1, i.e. $\bar{r}_1$, can be obtained as follows:

$$\bar{r}_1 = d_1\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + d_2 e^{\tilde{u}_3\theta_1}\bar{u}_2 + a_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + d_3 e^{\tilde{u}_3\theta_1}\bar{u}_2 + a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}\bar{u}_1$$

$$+ d_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}\bar{u}_3 + a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}\bar{u}_1 + d_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}\bar{u}_2$$

$$+ a_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}e^{\tilde{u}_2\theta_5}\bar{u}_1 \tag{23}$$

The simplified equation pairs for $\hat{C}$ and $\bar{r}$ pertaining to the other main groups can be obtained as shown below by using the procedure applied to the main group 1 and the appropriate simplification tools given in Appendix A. The subscripts indicate the main groups in the following equations. In these equations, d_{ij} denotes $d_i + d_j$. Note that, if J_k is prismatic, then the offset d_k is to be replaced with the joint variable s_k as done in obtaining the subgroup equations in Subsection 3.2.

$$\hat{C}_2 = e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{234}}e^{\tilde{u}_3\theta_5}e^{\tilde{u}_2\theta_6}e^{-\tilde{u}_1\pi/2} \tag{24}$$

$$\bar{r}_2 = d_1\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + d_{234}e^{\tilde{u}_3\theta_1}\bar{u}_2 + a_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}\bar{u}_1$$
$$+a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{234}}\bar{u}_1 + d_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{234}}\bar{u}_3 + a_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{234}}e^{\tilde{u}_3\theta_5}\bar{u}_1 \tag{25}$$

$$\hat{C}_3 = e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_{34}}e^{\tilde{u}_2\theta_5}e^{\tilde{u}_3\theta_6} \tag{26}$$

$$\bar{r}_3 = d_1\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + d_2 e^{\tilde{u}_3\theta_1}\bar{u}_2 + a_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + d_{34}e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_3$$
$$+a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_3}\bar{u}_1 + a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_{34}}\bar{u}_1 + d_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_{34}}\bar{u}_2$$
$$+a_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_{34}}e^{\tilde{u}_2\theta_5}\bar{u}_1 \tag{27}$$

$$\hat{C}_4 = e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_2\theta_4}e^{\tilde{u}_3\theta_5}e^{\tilde{u}_2\theta_6}e^{-\tilde{u}_1\pi/2} \tag{28}$$

$$\bar{r}_4 = d_1\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + d_2 e^{\tilde{u}_3\theta_1}\bar{u}_2 + a_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + d_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_3$$
$$+a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_3}\bar{u}_1 + d_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_3}\bar{u}_2 + a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_2\theta_4}\bar{u}_1$$
$$+d_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_2\theta_4}\bar{u}_3 + a_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_3}e^{\tilde{u}_2\theta_4}e^{\tilde{u}_3\theta_5}\bar{u}_1 \tag{29}$$

$$\hat{C}_5 = e^{\tilde{u}_3\theta_{1234}}e^{\tilde{u}_2\theta_5}e^{\tilde{u}_3\theta_6} \tag{30}$$

$$\bar{r}_5 = d_{1234}\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + a_2 e^{\tilde{u}_3\theta_{12}}\bar{u}_1 + a_3 e^{\tilde{u}_3\theta_{123}}\bar{u}_1 + a_4 e^{\tilde{u}_3\theta_{1234}}\bar{u}_1 + d_5 e^{\tilde{u}_3\theta_{1234}}\bar{u}_2$$
$$+a_5 e^{\tilde{u}_3\theta_{1234}}e^{\tilde{u}_2\theta_5}\bar{u}_1 \tag{31}$$

$$\hat{C}_6 = e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_{345}}e^{\tilde{u}_2\theta_6}e^{-\tilde{u}_1\pi/2} \tag{32}$$

$$\bar{r}_6 = d_1\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + d_2 e^{\tilde{u}_3\theta_1}\bar{u}_2 + a_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + d_{345}e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_3$$
$$+a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_3}\bar{u}_1 + a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_{34}}\bar{u}_1 + a_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_{345}}\bar{u}_1 \tag{33}$$

$$\hat{C}_7 = e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_{34}}e^{\tilde{u}_3\theta_5}e^{\tilde{u}_2\theta_6}e^{-\tilde{u}_1\pi/2} \tag{34}$$

$$\bar{r}_7 = d_{12}\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + a_2 e^{\tilde{u}_3\theta_{12}}\bar{u}_1 + d_{34}e^{\tilde{u}_3\theta_{12}}\bar{u}_2 + a_3 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_3}\bar{u}_1$$
$$+a_4 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_{34}}\bar{u}_1 + d_5 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_{34}}\bar{u}_3 + a_5 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_{34}}e^{\tilde{u}_3\theta_5}\bar{u}_1 \tag{35}$$

$$\hat{C}_8 = e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_3}e^{\tilde{u}_3\theta_4}e^{\tilde{u}_2\theta_5}e^{\tilde{u}_3\theta_6} \tag{36}$$

$$\bar{r}_8 = d_{12}\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + a_2 e^{\tilde{u}_3\theta_{12}}\bar{u}_1 + d_3 e^{\tilde{u}_3\theta_{12}}\bar{u}_2 + a_3 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_3}\bar{u}_1 + d_4 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_3}\bar{u}_3$$
$$+a_4 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_3}e^{\tilde{u}_3\theta_4}\bar{u}_1 + d_5 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_3}e^{\tilde{u}_3\theta_4}\bar{u}_2 + a_5 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_3}e^{\tilde{u}_3\theta_4}e^{\tilde{u}_2\theta_5}\bar{u}_1 \tag{37}$$

$$\hat{C}_9 = e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_{45}}e^{\tilde{u}_2\theta_6}e^{-\tilde{u}_1\pi/2} \tag{38}$$

$$\bar{r}_9 = d_1\bar{u}_3 + a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + d_{23}e^{\tilde{u}_3\theta_1}\bar{u}_2 + a_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}\bar{u}_1$$
$$+d_{45}e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}\bar{u}_3 + a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}\bar{u}_1 + a_5 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_{45}}\bar{u}_1 \tag{39}$$

3.2 Subgroups and Subgroup Equations

The list of the subgroups of the nine main groups is given in Table 4 with the non-zero link parameters and the number of industrial robots surveyed in this study. In the table, the first digit of the subgroup designation indicates the underlying main group and the second non-zero digit indicates the subgroup of that main group (e.g., subgroup 2.6 indicates the subgroup 6 of the main group 2). The second zero digit indicates the main group itself. The brand names and the models of the surveyed industrial robots are given in Appendix B with their subgroups and non-zero link parameters. If the joint J_k of a manipulator happens to be prismatic, the offset d_k becomes a joint variable, which is then denoted by s_k. In the column titled "Solution Type", CF denotes that a *closed-form* inverse kinematic solution can be obtained analytically and PJV denotes that the inverse kinematic solution can only be obtained semi-analytically using the so called *parametrized joint variable* method. The details of these two types of inverse kinematic solutions can be seen in Section 4.

Sub-Group	Twist Angles or Nonzero Link Parameters	Nr of Robots	Solution Type	Sub-Group	Twist Angles or Nonzero Link Parameters	Nr of Robots	Solution Type
1.0	-90°, 0°, 90°,-90°, 90°, 0°	73	-	3.2	$a_1, a_3, s_1, s_2, s_3, d_4$	1	CF
1.1	a_2, d_4	25	CF	3.3	d_2, s_3	1	CF
1.2	a_2, d_2, d_4	4	CF	3.4	a_2, s_3, d_5	1	PJV
1.3	a_1, a_2, d_4	3	CF	4.0	-90°, 90°, -90°, 90°,-90°, 0°	4	-
1.4	a_2, a_3, d_4	9	CF	4.1	$a_1, a_2, s_1, s_2, s_3, d_4$	1	CF
1.5	a_2, d_2, d_3, d_4	2	CF	4.2	s_2, d_4	1	CF
1.6	a_1, a_2, a_3, d_4	21	CF	4.3	s_3, d_5	1	CF/PJV
1.7	a_2, d_4, d_5	6	PJV	4.4	a_2, s_3, d_5	1	CF/PJV
1.8	a_2, d_2, d_4, d_5	1	PJV	5.0	0°, 0°, 0°, - 90°, 90°, 0°	1	-
1.9	a_1, a_2, a_3, d_4, d_5	1	PJV	5.1	a_1, a_2, s_3	1	CF
1.10	a_1, a_2, a_3, d_4, d_5	1	PJV	6.0	- 90°, 90°, 0°, 0°, - 90°, 0°	1	-
2.0	-90°, 0°, 0°, 90°, -90°, 0°	12	-	6.1	a_3, a_4	1	CF
2.1	a_2, a_3, d_4	1	CF	7.0	0°, - 90°, 0°, 90°, - 90°, 0°	1	-
2.2	a_2, a_3, a_4	6	CF	7.1	a_2, s_2, s_3	1	CF
2.3	a_2, a_3, d_5	2	CF	8.0	0°, -90°, 90°, -90°, 90°, 0°	2	-
2.4	a_2, a_3, d_2, d_5	1	CF	8.1	a_1, d_3, d_4	1	CF
2.5	a_1, a_2, a_3, d_2, d_5	1	CF	8.2	a_1, a_2, d_3, d_4	1	CF
2.6	a_1, a_2, a_3, d_4, d_5	1	CF	9.0	- 90°, 0°, 90°, 0°, - 90°, 0°	1	-
3.0	-90°, 90°, 0°,-90°, 90°, 0°	5	-	9.1	$a_1, a_2, a_3, a_4, d_4, d_5$	1	PJV
3.1	s_1, s_2, s_3	1	CF				

Table 4. Subgroups of Six-Joint Robots

Using the information about the link lengths and the offsets, the simplified subgroup equations are obtained for the wrist locations as shown below by using again the exponential rotation matrix simplification tools given in Appendix A. In these equations, the first and second subscripts associated with the

wrist locations indicate the related main groups and subgroups. For all sub-groups of the main groups 1 and 2, the rotation matrix is as given in the main group equations.

$$\overline{r}_{11} = a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3 \tag{40}$$

$$\overline{r}_{12} = d_2 e^{\tilde{u}_3 \theta_1} \overline{u}_2 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3 \tag{41}$$

$$\overline{r}_{13} = a_1 e^{\tilde{u}_3 \theta_1} \overline{u}_1 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3 \tag{42}$$

$$\overline{r}_{14} = a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3 \tag{43}$$

$$\overline{r}_{15} = d_{23} e^{\tilde{u}_3 \theta_1} \overline{u}_2 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3 \tag{44}$$

$$\overline{r}_{16} = a_1 e^{\tilde{u}_3 \theta_1} \overline{u}_1 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3 \tag{45}$$

$$\overline{r}_{17} = a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3 + d_5 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} e^{\tilde{u}_3 \theta_4} \overline{u}_2 \tag{46}$$

$$\overline{r}_{18} = d_2 e^{\tilde{u}_3 \theta_1} \overline{u}_2 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3 + d_5 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} e^{\tilde{u}_3 \theta_4} \overline{u}_2 \tag{47}$$

$$\overline{r}_{19} = a_1 e^{\tilde{u}_3 \theta_1} \overline{u}_1 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3$$
$$+ d_5 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} e^{\tilde{u}_3 \theta_4} \overline{u}_2 \tag{48}$$

$$\overline{r}_{110} = a_1 e^{\tilde{u}_3 \theta_1} \overline{u}_1 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + d_3 e^{\tilde{u}_3 \theta_1} \overline{u}_2 + d_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_3$$
$$+ d_5 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} e^{\tilde{u}_3 \theta_4} \overline{u}_2 \tag{49}$$

$$\overline{r}_{21} = a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} \overline{u}_2 \tag{50}$$

$$\overline{r}_{22} = a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + a_4 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{234}} \overline{u}_1 \tag{51}$$

$$\overline{r}_{23} = a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + d_5 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{234}} \overline{u}_3 \tag{52}$$

$$\overline{r}_{24} = d_2 e^{\tilde{u}_3 \theta_1} \overline{u}_2 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + d_5 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{234}} \overline{u}_3 \tag{53}$$

$$\overline{r}_{25} = a_1 e^{\tilde{u}_3 \theta_1} \overline{u}_1 + d_2 e^{\tilde{u}_3 \theta_1} \overline{u}_2 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + d_5 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{234}} \overline{u}_3 \tag{54}$$

$$\overline{r}_{26} = a_1 e^{\tilde{u}_3 \theta_1} \overline{u}_1 + a_2 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_2} \overline{u}_1 + a_3 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{23}} \overline{u}_1 + d_4 e^{\tilde{u}_3 \theta_1} \overline{u}_2 + d_5 e^{\tilde{u}_3 \theta_1} e^{\tilde{u}_2 \theta_{234}} \overline{u}_3 \tag{55}$$

The constant joint angles associated with the prismatic joints are as follows for the subgroups of the main group 3: For the subgroups 3.1 and 3.2 having s_1, s_2, and s_3 as the variable offsets, the joint angles are $\theta_1 = 0°$, $\theta_2 = 90°$, $\theta_3 = 0°$ or $90°$. For the subgroups 3.3 and 3.4, having s_3 as the only variable offset, θ_3 is either $0°$ or $90°$. This leads to the following equations:

$$\hat{C}_{31} = \hat{C}_{32} = e^{\tilde{u}_1\theta_{34}}e^{\tilde{u}_2\theta_5'}e^{\tilde{u}_3\theta_6} = \begin{cases} e^{\tilde{u}_1\theta_4}e^{\tilde{u}_2\theta_5'}e^{\tilde{u}_3\theta_6} & \text{for} \quad \theta_3 = 0° \\ e^{\tilde{u}_1\theta_4'}e^{\tilde{u}_2\theta_5'}e^{\tilde{u}_3\theta_6} & \text{for} \quad \theta_3 = 90° \end{cases} \tag{56}$$

Here, $\theta_4' = \theta_4 + 90°$ and $\theta_5' = \theta_5 + 90°$.

$$\bar{r}_{31} = s_3\bar{u}_1 + s_2\bar{u}_2 + s_1\bar{u}_3 \tag{57}$$

$$\bar{r}_{32} = s_3'\bar{u}_1 + s_2\bar{u}_2 + s_1\bar{u}_3 - a_3e^{\tilde{u}_1\theta_3}\bar{u}_3 = \begin{cases} s_3'\bar{u}_1 + s_2\bar{u}_2 + s_1'\bar{u}_3 & \text{for} \quad \theta_3 = 0° \\ s_3'\bar{u}_1 + s_2'\bar{u}_2 + s_1\bar{u}_3 & \text{for} \quad \theta_3 = 90° \end{cases} \tag{58}$$

Here, $s_1' = s_1 - a_3$, $s_2' = s_2 + a_3$, and $s_3' = s_3 + a_1 + d_4$.

$$\hat{C}_{33} = \hat{C}_{34} = \begin{cases} e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_4}e^{\tilde{u}_2\theta_5}e^{\tilde{u}_3\theta_6} & \text{for} \quad \theta_3 = 0° \\ e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_4'}e^{\tilde{u}_2\theta_5}e^{\tilde{u}_3\theta_6} & \text{for} \quad \theta_3 = 90° \end{cases} \tag{59}$$

$$\bar{r}_{33} = d_2e^{\tilde{u}_3\theta_1}\bar{u}_2 + s_3e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_3 \tag{60}$$

$$\bar{r}_{34} = a_2e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + s_3e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_3 + d_5e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_{34}}\bar{u}_2$$
$$= \begin{cases} a_2e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + s_3e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_3 + d_5e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_4}\bar{u}_2 & \text{for} \quad \theta_3 = 0° \\ a_2e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + s_3e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_3 + d_5e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_4'}\bar{u}_2 & \text{for} \quad \theta_3 = 90° \end{cases} \tag{61}$$

Here, $\theta_4' = \theta_4 + 90°$.

The constant joint angles associated with the prismatic joints, for the subgroups of the main group 4 are as follows: For the subgroup 4.1 having s_1, s_2, and s_3 as the variable offsets, the joint angles are $\theta_1 = 0°$, $\theta_2 = 90°$, $\theta_3 = 0°$ or $90°$. For the subgroup 4.2 having s_2 as the only variable offset, θ_2 is either $0°$ or $90°$. For the subgroups 4.3 and 4.4 having s_3 as the only variable offset, θ_3 is either $0°$ or $90°$. This leads to the following equations:

$$\hat{C}_{41} = e^{\tilde{u}_1\theta_3} e^{\tilde{u}_2\theta_4'} e^{\tilde{u}_3\theta_5} e^{\tilde{u}_2\theta_6} e^{-\tilde{u}_1\pi/2} = \begin{cases} e^{\tilde{u}_2\theta_4'} e^{\tilde{u}_3\theta_5} e^{\tilde{u}_2\theta_6} e^{-\tilde{u}_1\pi/2} & \text{for} \quad \theta_3 = 0° \\ e^{\tilde{u}_3\theta_4'} e^{-\tilde{u}_2\theta_5} e^{\tilde{u}_3\theta_6} & \text{for} \quad \theta_3 = 90° \end{cases} \tag{62}$$

$$\overline{r}_{41} = s_3'\overline{u}_1 + s_2\overline{u}_2 + s_1'\overline{u}_3 + d_4 e^{\tilde{u}_1\theta_3}\overline{u}_2 = \begin{cases} s_3'\overline{u}_1 + s_2'\overline{u}_2 + s_1'\overline{u}_3 & \text{for} \quad \theta_3 = 0° \\ s_3'\overline{u}_1 + s_2\overline{u}_2 + s_1''\overline{u}_3 & \text{for} \quad \theta_3 = 90° \end{cases} \tag{63}$$

Here, $s_1' = s_1 - a_2$, $s_1'' = s_1' + d_4$, $s_2' = s_2 + d_4$, and $s_3' = s_3 + a_1$.

$$\hat{C}_{42} = \begin{cases} e^{\tilde{u}_3\theta_{13}} e^{\tilde{u}_2\theta_4} e^{\tilde{u}_3\theta_5} e^{\tilde{u}_2\theta_6} e^{-\tilde{u}_1\pi/2} & \text{for} \quad \theta_2 = 0° \\ e^{\tilde{u}_3\theta_1} e^{\tilde{u}_1\theta_3} e^{\tilde{u}_2\theta_4'} e^{\tilde{u}_3\theta_5} e^{\tilde{u}_2\theta_6} e^{-\tilde{u}_1\pi/2} & \text{for} \quad \theta_2 = 90° \end{cases} \tag{64}$$

Here, $\theta_4' = \theta_4 + 90°$.

$$\overline{r}_{42} = s_2 e^{\tilde{u}_3\theta_1}\overline{u}_2 + d_4 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} e^{\tilde{u}_3\theta_3}\overline{u}_2 = \begin{cases} s_2 e^{\tilde{u}_3\theta_1}\overline{u}_2 + d_4 e^{\tilde{u}_3\theta_{13}}\overline{u}_2 & \text{for} \quad \theta_2 = 0° \\ s_2 e^{\tilde{u}_3\theta_1}\overline{u}_2 + d_4 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_1\theta_3}\overline{u}_2 & \text{for} \quad \theta_2 = 90° \end{cases} \tag{65}$$

$$\hat{C}_{43} = \hat{C}_{44} = \begin{cases} e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_{24}} e^{\tilde{u}_3\theta_5} e^{\tilde{u}_2\theta_6} e^{-\tilde{u}_1\pi/2} & \text{for} \quad \theta_3 = 0° \\ e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} e^{-\tilde{u}_1\theta_4} e^{\tilde{u}_3\theta_5'} e^{\tilde{u}_2\theta_6} e^{-\tilde{u}_1\pi/2} & \text{for} \quad \theta_3 = 90° \end{cases} \tag{66}$$

Here, $\theta_5' = \theta_5 + 90°$.

$$\overline{r}_{43} = s_3 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2}\overline{u}_3 + d_5 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} e^{\tilde{u}_3\theta_3} e^{\tilde{u}_2\theta_4}\overline{u}_3$$
$$= \begin{cases} s_3 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2}\overline{u}_3 + d_5 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_{24}}\overline{u}_3 & \text{for} \quad \theta_3 = 0° \\ s_3 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2}\overline{u}_3 + d_5 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} e^{-\tilde{u}_1\theta_4}\overline{u}_3 & \text{for} \quad \theta_3 = 90° \end{cases} \tag{67}$$

$$\overline{r}_{44} = a_2 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2}\overline{u}_1 + s_3 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2}\overline{u}_3 + d_5 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} e^{\tilde{u}_3\theta_3} e^{\tilde{u}_2\theta_4}\overline{u}_3$$
$$= \begin{cases} a_2 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2}\overline{u}_1 + s_3 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2}\overline{u}_3 + d_5 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_{24}}\overline{u}_3 & \text{for} \quad \theta_3 = 0° \\ a_2 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2}\overline{u}_1 + s_3 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2}\overline{u}_3 + d_5 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} e^{-\tilde{u}_1\theta_4}\overline{u}_3 & \text{for} \quad \theta_3 = 90° \end{cases} \tag{68}$$

The constant joint angle θ_3 associated with the prismatic joint J_3 for the sub-group of the main group 5 is either $0°$ or $90°$. This leads to the following equations:

$$\hat{C}_{51} = e^{\tilde{u}_3\theta_{1234}} e^{\tilde{u}_2\theta_5} e^{\tilde{u}_3\theta_6} = \begin{cases} e^{\tilde{u}_3\theta_{124}} e^{\tilde{u}_2\theta_5} e^{\tilde{u}_3\theta_6} & \text{for} \quad \theta_3 = 0° \\ e^{\tilde{u}_3\theta_{124}'} e^{\tilde{u}_2\theta_5} e^{\tilde{u}_3\theta_6} & \text{for} \quad \theta_3 = 90° \end{cases} \tag{69}$$

Here, $\theta'_{124} = \theta_{124} + 90°$.

$$\bar{r}_{51} = a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + a_2 e^{\tilde{u}_3\theta_{12}}\bar{u}_1 + s_3\bar{u}_3 \tag{70}$$

For the subgroup of the main group 6, the rotation matrix is as given in the main group equations and the wrist point location is expressed as

$$\bar{r}_{61} = a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_3}\bar{u}_1 + a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}e^{\tilde{u}_3\theta_{34}}\bar{u}_1 \tag{71}$$

The constant joint angles associated with the prismatic joints for the subgroup of the main group 7 are as follows: The joint angle θ_2 is $90°$ for the prismatic joint J_2 and the joint angle θ_3 is either $0°$ or $90°$ for the prismatic joint J_3. This leads to the following equations:

$$\hat{C}_{71} = e^{\tilde{u}_3\theta'_1}e^{\tilde{u}_2\theta_{34}}e^{\tilde{u}_3\theta_5}e^{\tilde{u}_2\theta_6}e^{-\tilde{u}_1\pi/2} = \begin{cases} e^{\tilde{u}_3\theta'_1}e^{\tilde{u}_2\theta_4}e^{\tilde{u}_3\theta_5}e^{\tilde{u}_2\theta_6}e^{-\tilde{u}_1\pi/2} & \text{for } \theta_3 = 0° \\ e^{\tilde{u}_3\theta'_1}e^{\tilde{u}_2\theta'_4}e^{\tilde{u}_3\theta_5}e^{\tilde{u}_2\theta_6}e^{-\tilde{u}_1\pi/2} & \text{for } \theta_3 = 90° \end{cases} \tag{72}$$

Here, $\theta'_1 = \theta_1 + 90°$ and $\theta'_4 = \theta_4 + 90°$.

$$\bar{r}_{71} = s_2\bar{u}_3 + a_2 e^{\tilde{u}_3\theta_1}\bar{u}_2 - s_3 e^{\tilde{u}_3\theta_1}\bar{u}_1 \tag{73}$$

For the subgroups of the main group 8, the rotation matrix is as given in the main group equations and the wrist point locations are expressed as

$$\bar{r}_{81} = a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + d_3 e^{\tilde{u}_3\theta_{12}}\bar{u}_2 + d_4 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_3}\bar{u}_3 \tag{74}$$

$$\bar{r}_{82} = a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + a_2 e^{\tilde{u}_3\theta_{12}}\bar{u}_1 + d_3 e^{\tilde{u}_3\theta_{12}}\bar{u}_2 + d_4 e^{\tilde{u}_3\theta_{12}}e^{\tilde{u}_2\theta_3}\bar{u}_3 \tag{75}$$

For the subgroup of the main group 9, the rotation matrix is as given in the main group equations and the wrist point location is expressed as

$$\bar{r}_{91} = a_1 e^{\tilde{u}_3\theta_1}\bar{u}_1 + a_2 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}\bar{u}_1 + a_3 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}\bar{u}_1 + d_{45} e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}\bar{u}_3$$
$$+ a_4 e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}\bar{u}_1 \tag{76}$$

4. Classification Based Inverse Kinematics

In the inverse kinematics problem, the elements of $\hat{C}$ and $\bar{r}$ are available and it is desired to obtain the six joint variables. For this purpose, the required elements of the $\bar{r}$ and $\hat{C}$ matrices can be extracted as follows:

$$r_i = \bar{u}_i^T \bar{r} \quad \text{and} \quad c_{ij} = \bar{u}_i^T \hat{C} \, \bar{u}_j \tag{77}$$

For most of the manipulators, which are called *separable*, the wrist point position vector contains only three joint variables. The most typical samples of such manipulators are those with spherical wrists (Pieper & Roth, 1969). Therefore, for this large class of manipulators, Equation (14) is first used to obtain the *arm joint variables*, and then Equation (11) is used to determine the remaining three of the *wrist joint variables* contained in the $\hat{C}$ matrix. After obtaining the arm joint variables, $\hat{C}$ equation is arranged in such a way that the arm joint variables are collected at one side of the equation leaving the remaining joint variables to be found at the other side within a new matrix $\hat{M}$, which is called *modified orientation matrix*. The three arguments of $\hat{M}$ happen to be the wrist joint variables and they appear similarly as an Euler Angle sequence of three successive rotations. After this preparation, the solution of the modified orientation equation directly gives the wrist joint variables. The most commonly encountered sequences are given in Table 5 with their solutions and singularities. In the table, $\sigma = \pm 1$ indicates the alternative solutions. When the sequence becomes singular, the angles ϕ_1 and ϕ_3 can not be determined and the mobility of the wrist becomes restricted. For a more detailed singularity analysis, see (Özgören, 1999 and 2002).

However, there may also be other kinds of separable manipulators for which it is the $\hat{C}$ matrix that contains only three joint variables. In such a case, the solution is started naturally with the $\hat{C}$ equation and then the remaining three joint variables are found from the $\bar{r}$ equation. Besides, there are inseparable manipulators as well for which both of the $\hat{C}$ and $\bar{r}$ equations contain more than three joint variables. The most typical sample of this group is the Cincinnati Milacron-T3 (CM-T3 566 or 856) robot. It has four unknown variables in each of its $\hat{C}$ and $\bar{r}$ equations. For such manipulators, since the $\hat{C}$ and $\bar{r}$ equations are not separable, they have to be solved jointly and therefore a closed-form inverse kinematic solution cannot be obtained in general. Nevertheless, for some special forms of such manipulators, Cincinnati Milacron-T3 being one of them, it becomes possible to obtain a closed-form inverse kinematic solution. For a more detailed analysis and discussion of inverse kinematics covering all possible six-joint serial manipulators, see (Özgören, 2002).

Sequence	3-2-3	2-3-2	1-2-3
$\hat{C}$	$e^{\bar{u}_3\phi_1}e^{\bar{u}_2\phi_2}e^{\bar{u}_3\phi_3}$	$e^{\bar{u}_2\phi_1}e^{\bar{u}_3\phi_2}e^{\bar{u}_2\phi_3}$	$e^{\bar{u}_1\phi_1}e^{\bar{u}_2\phi_2}e^{\bar{u}_3\phi_3}$
ϕ_1	$\mathrm{atan2}\,(\sigma c_{23},\ \sigma c_{13})$	$\mathrm{atan2}\,(\sigma c_{32},\ -\sigma c_{12})$	$\mathrm{atan2}\,(\sigma c_{23},\ \sigma c_{33})$
ϕ_2	$\mathrm{atan2}\,(\sigma\sqrt{1-c_{33}^2}\,,\ c_{33})$	$\mathrm{atan2}\,(\sigma\sqrt{1-c_{22}^2}\,,\ c_{22})$	$\mathrm{atan2}\,(-c_{13},\ \sigma\sqrt{1-c_{13}^2}\,)$
ϕ_3	$\mathrm{atan2}\,(\sigma c_{32},\ -\sigma c_{31})$	$\mathrm{atan2}\,(\sigma c_{23},\ \sigma c_{21})$	$\mathrm{atan2}\,(-\sigma c_{12},\ \sigma c_{11})$
Singularity	$\phi_2=0°$ or $\phi_2=\pm180°$	$\phi_2=0°$ or $\phi_2=\pm180°$	$\phi_2=\pm90°$

Table 5. Wrist Joint Variables in the Most Commonly Encountered Sequences

For all the groups of six-joint manipulators considered in this chapter, there are two types of inverse kinematic solution, namely the *closed-form* (CF) solution and the *parametrized joint variable* (PJV) solution where one of the joint variables is temporarily treated as if it is a known parameter. For many of the six-joint manipulators, a closed-form solution can be obtained if the wrist point location equation and the end-effector orientation equation are separable, i.e. if it is possible to write the wrist point equation as

$$\bar{r} = \bar{r}(q_1, q_2, q_3) \tag{78}$$

where q_k denotes the kth joint variable, which is either θ_k or s_k. Since there are three unknowns in the three scalar equations contained in Equation (78), the unknowns q_1, q_2, q_3 can be directly obtained from those equations. The end-effector orientation variables q_4, q_5, q_6 are then obtained by using the equation for $\hat{C}$.

In general, the necessity for a PJV solution arises when a six-joint manipulator has a non-zero value for one or more of the structural length parameters a_4, a_5, and d_5. However, in all of the manipulators that are considered here, only d_5 exists as an offset at the wrist. In this case, $\bar{r}$ will be a function of four variables as

$$\bar{r} = \bar{r}(q_1, q_2, q_3, q_4) \tag{79}$$

Since, there are more unknowns now than the three scalar equations contained in Equation (79), the variables q_1, q_2, q_3, and q_4 can not be obtained directly. So, one of the joint variables is parametrized and the remaining five joint variables are obtained as functions of this parametrized variable from five of the six scalar equations contained in the $\hat{C}$ and $\bar{r}$ equations. Then, the remaining sixth scalar equation is solved for the parametrized variable using a suitable numerical method. Finally, by substituting the numerically found value of this joint variable into the previously obtained expressions of the remaining joint variables, the complete solution is obtained.

There may also be a situation that a six-joint manipulator can have non-zero values for the structural parameters d_5 and a_5 so that

$$\bar{r} = \bar{r}(q_1, q_2, q_3, q_4, q_5) \tag{80}$$

In this case, two joint variables can be chosen as the parametrized joint variables and the remaining four joint variables are obtained as functions of these two parametrized variables. Then, using a suitable numerical method, the remaining two equations are solved for the parametrized joint variables. Afterwards, the inverse kinematic solution is completed similarly as described above. However, if desired, it may also be possible to reduce the remaining two equations to a single but rather complicated univariate polynomial equation by using methods similar to those in (Raghavan & Roth, 1993; Manseur & Doty, 1996; Lee et al, 1991).

Although the analytical or semi-analytical solution methods are necessarily dependent on the specific features of the manipulator of concern, the procedure outlined below can be used as a general course to follow for most of the manipulators considered here.

1. The wrist location equation is manipulated symbolically to obtain three scalar equations using the simplification tools given in Appendix A.

2. The three scalar equations are worked on in order to cast them into the forms of the trigonometric equations considered in Appendix C, if they are not so already.

3. As a sufficient condition for a CF solution, if there exists a scalar equation containing only one joint variable, or if such an equation can be generated by combining the other available equations, it can be solved for that joint variable to start the solution.

4. If such an equation does not exist or cannot be generated, then the PJV method is used. Thus, except the parametrized joint variable, there will again be a single equation with a single unknown to start the solution.

5. The two remaining scalar equations pertaining to the wrist location are then used to determine the remaining two of the arm joint variables again by using the appropriate ones of the trigonometric equation solutions given in Appendix C.

6. Once the arm joint variables are obtained, the solution of the orientation equation for the three wrist joint variables is straightforward since it will be the same as the solution pattern of one of the commonly encountered rotation sequences, such as 1-2-3, 3-2-3, etc, which are shown in Table 5.

When the manipulators in Table 4 are considered from the viewpoint of the solution procedure described above, they are accompanied by the designations CF (having inverse kinematic solution in closed-form) and PJV (having inverse kinematic solution using a parametrized joint variable). As noted, almost all the manipulators in Table 4 are designated exclusively either with CF or PJV. Exceptionally, however, the subgroups 4.3 and 4.4 have both of the designations. This is because the solution type can be either CF or PJV depending on whether $\theta_3 = 0°$ or $\theta_3 = 90°$, respectively.

In this section, the inverse kinematic solutions for the subgroups 1.1 (e.g. KUKA IR 662/10), 1.7 (e.g. GMF S-3 L or R) and 4.4 (e.g. Unimate 4000) are given in order to demonstrate the solution procedure described above. As indicated in Table 4, the subgroup 1.1 can have the inverse kinematic solution in closed-form, whereas the subgroup 1.7 necessitates a PJV solution. On the other hand, for the subgroup 4.4, which has a prismatic joint, the inverse kinematic solution can be obtained either in closed-form or by using the PJV method depending on whether the structural parameter θ_3 associated with the prismatic joint is 0° or 90°. Although $\theta_3 = 0°$ for the enlisted industrial robot of this subgroup, the solution for $\theta_3 = 90°$ is also considered here for sake of demonstrating the application of the PJV method to a robot with a prismatic joint as well. It should be noted that, the subgroups 1.1 and 1.7 are examples to robots with only revolute joints and the subgroup 4.4 is an example to robots with revolute and prismatic joints. The subgroups 1.1 and 1.7 are considered particularly because the number of industrial robots is high within these categories. As an additional example, the ABB IRB2000 industrial robot is also considered to demonstrate the applicability of the method to manipulators containing closed kinematic chains. However, the solutions for the other subgroups or a new manipulator with a different kinematic structure can be obtained easily by using the same systematic approach.

4.1 Inverse Kinematics of Subgroups 1.1 and 1.7

For all the subgroups of the main group 1, the orientation matrix is

$$\hat{C}_1 = e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_{23}} e^{\tilde{u}_3\theta_4} e^{\tilde{u}_2\theta_5} e^{\tilde{u}_3\theta_6} \tag{81}$$

Since all the subgroups have the same $\hat{C}_1$ matrix, they will have identical equations for θ_4, θ_5 and θ_6. In other words, these variables can always be determined from the following equation, after finding the other variables somehow from the wrist location equations of the subgroups:

$$e^{\tilde{u}_3\theta_4} e^{\tilde{u}_2\theta_5} e^{\tilde{u}_3\theta_6} = \hat{M}_1 \tag{82}$$

Here, $\hat{M}_1 = e^{-\tilde{u}_2\theta_{23}} e^{-\tilde{u}_3\theta_1} \hat{C}_1$ and $\theta_{23} = \theta_2 + \theta_3$. Since the sequence in Equation (82) is 3-2-3 , using Table 5, the angles θ_4, θ_5 and θ_6 are obtained as follows, assuming that θ_1 and θ_{23} have already been determined as explained in the next subsection:

$$\theta_4 = \text{atan2}\,(\sigma_5 m_{23},\ \sigma_5 m_{13}) \tag{83}$$

$$\theta_5 = \text{atan2}\,(\sigma_5 \sqrt{1 - m_{33}^2},\ m_{33}) \tag{84}$$

$$\theta_6 = \text{atan2}\,(\sigma_5 m_{32},\ -\sigma_5 m_{13}) \tag{85}$$

Here, $\sigma_5 = \pm 1$ and $m_{ij} = \bar{u}_i^T \hat{M}_1 \bar{u}_j$.

Note that this 3-2-3 sequence becomes singular if $\theta_5 = 0$ or $\theta_5 = \pm 180°$, but the latter case is not physically possible. This is the first kind of singularity of the manipulator, which is called *wrist singularity*. In this singularity with $\theta_5 = 0$, the axes of the fourth and sixth joints become aligned and Equation (82) degenerates into

$$e^{\tilde{u}_3\theta_4} e^{\tilde{u}_3\theta_6} = e^{\tilde{u}_3(\theta_4 + \theta_6)} = e^{\tilde{u}_3\theta_{46}} = \hat{M}_1 \tag{86}$$

This equation implies that, in the singularity, θ_4 and θ_6 become arbitrary and they cannot be determined separately although their combination $\theta_{46} = \theta_4 + \theta_6$ can still be determined as $\theta_{46} = \text{atan2}\,(m_{21},\ m_{11})$. This means that one of the fourth and sixth joints becomes redundant in orienting the end-effector, which in turn becomes underivable about the axis normal to the axes of the fifth and sixth joints.

4.1.1 Inverse Kinematics of Subgroup 1.1

The wrist point position vector of this subgroup given in Equation (40) can be written again as follows by transposing the leading exponential matrix on the right hand side to the left hand side:

$$e^{-\tilde{u}_3\theta_1} \bar{r}_{11} = a_2 e^{\tilde{u}_2\theta_2} \bar{u}_1 + d_4 e^{\tilde{u}_2\theta_{23}} \bar{u}_3 \tag{87}$$

Premultiplying both sides of Equation (87) by $\bar{u}_1^T$, $\bar{u}_2^T$, $\bar{u}_3^T$ and using the simplification tool E.8 in Appendix A, the following equations can be obtained.

$$r_1 \cos\theta_1 + r_2 \sin\theta_1 = a_2 \cos\theta_2 + d_4 \sin\theta_{23} \tag{88}$$

$$r_2 \cos \theta_1 - r_1 \sin \theta_1 = 0 \tag{89}$$

$$r_3 = -a_2 \sin \theta_2 + d_4 \cos \theta_{23} \tag{90}$$

Here, r_1, r_2 and r_3 are the base frame components of the wrist position vector, $\bar{r}_{11}$.

From Equation (89), θ_1 can be obtained as follows by using the trigonometric equation T1 in Appendix C, provided that $r_2^2 + r_1^2 \neq 0$:

$$\theta_1 = \text{atan2} \left(\sigma_1 r_2, \sigma_1 r_1 \right) \quad \text{and} \quad \sigma_1 = \pm 1 \tag{91}$$

If $r_2^2 + r_1^2 = 0$, i.e. if $r_2 = r_1 = 0$, i.e. if the wrist point is located on the axis of the first joint, the second kind of singularity occurs, which is called *shoulder singularity*. In this singularity, Equation (89) degenerates into $0 = 0$ and therefore θ_1 cannot be determined. In other words, the first joint becomes ineffective in positioning the wrist point, which in turn becomes underivable in the direction normal to the arm plane (i.e. the plane formed by the links 2 and 3).

To continue with the solution, let

$$\rho_1 = r_1 \cos \theta_1 + r_2 \sin \theta_1 \tag{92}$$

Thus, Equation (88) becomes

$$\rho_1 = a_2 \cos \theta_2 + d_4 \sin \theta_{23} \tag{93}$$

Using Equations (90) and (93) in accordance with T6 in Appendix C, θ_3 can be obtained as follows, provided that $-1 \leq \rho_2 \leq 1$:

$$\theta_3 = \text{atan2} \left(\rho_2, \sigma_3 \sqrt{1 - \rho_2^2} \right) \quad \text{and} \quad \sigma_3 = \pm 1 \tag{94}$$

Here,

$$\rho_2 = \frac{(\rho_1^2 + r_3^2) - (a_2^2 + d_4^2)}{2 a_2 d_4} \tag{95}$$

Note that the constraint on ρ_2 implies a working space limitation on the manipulator, which can be expressed more explicitly as

$$(a_2 - d_4)^2 \leq \rho_1^2 + r_3^2 \leq (a_2 + d_4)^2 \tag{96}$$

Expanding $\sin \theta_{23}$ and $\cos \theta_{23}$ in Equation (90) and (93) and rearranging the terms as coefficients of $\sin \theta_2$ and $\cos \theta_2$, the following equations can be obtained.

$$\rho_1 = \rho_3 \cos \theta_2 + \rho_4 \sin \theta_2 \qquad (97)$$

$$r_3 = \rho_4 \cos \theta_2 - \rho_3 \sin \theta_2 \qquad (98)$$

Here,

$$\rho_3 = a_2 + d_4 \sin \theta_3 \qquad (99)$$

$$\rho_4 = d_4 \cos \theta_3 \qquad (100)$$

According to T4 in Appendix C, Equations (97) and (98) give θ_2 as follows, provided that $\rho_3^2 + \rho_4^2 \neq 0$:

$$\theta_2 = \text{atan2}\left(\rho_4 \rho_1 - \rho_3 r_3 \, , \, \rho_4 r_3 - \rho_3 \rho_1\right) \qquad (101)$$

If $\rho_3^2 + \rho_4^2 = 0$, i.e. if $\rho_3 = \rho_4 = 0$, the third kind of singularity occurs, which is called *elbow singularity*. In this singularity, both of Equations (97) and (98) degenerate into $0 = 0$. Therefore, θ_2 cannot be determined. Note that, according to Equations (99) and (100), it is possible to have $\rho_3 = \rho_4 = 0$ only if $a_2 = d_4$ and $\theta_3 = \pm 180^\circ$. This means that the elbow singularity occurs if the upper and front arms (i.e. the links 2 and 3) have equal lengths and the front arm is folded back onto the upper arm so that the wrist point coincides with the shoulder point. In this configuration, the second joint becomes ineffective in positioning the wrist point, which in turn becomes underivable neither along the axis of the second joint nor in a direction parallel to the upper arm.

As seen above, the closed-form inverse kinematic solution is obtained for the subgroup 1.1 as expressed by Equations (83)-(85) and (91)-(101). The completely analytical nature of the solution provided all the multiplicities (indicated by the sign variables σ_1, σ_2, etc), the singularities, and the working space limitations alongside with the solution.

4.1.2 Inverse Kinematics of Subgroup 1.7

The wrist point position vector of this subgroup is given in Equation (46). From that equation, the following scalar equations can be obtained as done previously for the subgroup 1.1:

$$r_1 \cos\theta_1 + r_2 \sin\theta_1 = a_2 \cos\theta_2 + d_4 \sin\theta_{23} - d_5 \cos\theta_{23} \sin\theta_4 \qquad (102)$$

$$r_2 \cos\theta_1 - r_1 \sin\theta_1 = d_5 \cos\theta_4 \qquad (103)$$

$$r_3 = -a_2 \sin\theta_2 + d_4 \cos\theta_{23} + d_5 \sin\theta_{23} \sin\theta_4 \qquad (104)$$

Here, r_1, r_2 and r_3 are the components of the wrist position vector, $\bar{r}_{17}$. Note that Equations (102)-(104) contain four unknowns (θ_1, θ_2, θ_3, θ_4). Therefore, it now becomes necessary to use the PJV method. That is, one of these four unknowns must be parametrized. On the other hand, Equation (103) is the simplest one of the three equations. Therefore, it will be reasonable to parametrize either θ_1 or θ_4. As it is shown in (Balkan et al., 1997, 2000), the solutions obtained by parametrizing θ_1 and θ_4 expose different amounts of explicit information about the multiple and singular configurations of the manipulators belonging to this subgroup. The rest of the information is concealed within the equation to be solved numerically. It happens that the solution obtained by parametrizing θ_4 reveals more information so that the critical shoulder singularity of the manipulator can be seen *explicitly* in the relevant equations; whereas the solution obtained by parametrizing θ_1 conceals it. Therefore, θ_4 is chosen as the parametrized joint variable in the solution presented below. As the starting step, θ_1 can be obtained from Equation (103) as follows by using T3 in Appendix C, provided that $r_2^2 + r_1^2 > 0$ and $r_2^2 + r_1^2 \geq \rho_5^2$:

$$\theta_1 = \text{atan2}\,(-r_2, r_1) + \sigma_1 \text{atan2}\,(\sqrt{r_1^2 + r_2^2 - \rho_5^2}, \rho_5) \quad \text{and} \quad \sigma_1 = \pm 1 \qquad (105)$$

Here,

$$\rho_5 = d_5 \cos\theta_4 \qquad (106)$$

If $r_2^2 + r_1^2 = 0$, which necessitates that $\rho_5 = 0$ or $\theta_4 = \pm 90°$, the shoulder singularity occurs. In that case, Equation (103) degenerates into $0 = 0$ and therefore θ_1 becomes arbitrary. The consequences are the same as those of the subgroup 1.1.

On the other hand, the inequality constraint $r_2^2 + r_1^2 \geq \rho_5^2$ indicates a working space limitation on the manipulator.

Equations (102) and (104) can be arranged as shown below:

$$x_1 = a_2 \cos\theta_2 + d_4 \sin\theta_{23} - \rho_6 \cos\theta_{23} \qquad (107)$$

$$r_3 = -a_2 \sin\theta_2 + d_4 \cos\theta_{23} + \rho_6 \sin\theta_{23} \tag{108}$$

Here,

$$\rho_6 = d_5 \sin\theta_4 \tag{109}$$

According to T9 in Appendix C, Equations (107) and (108) give θ_3 as follows, provided that $\rho_1^2 + d_4^2 \geq \rho_7^2$:

$$\theta_3 = \text{atan2}\,(d_4, -\rho_6) + \sigma_3 \text{atan2}\,(\sqrt{\rho_1^2 + d_4^2 - \rho_7^2},\, \rho_7) \quad \text{and} \quad \sigma_3 = \pm 1 \tag{110}$$

Here,

$$\rho_7 = \frac{(\rho_1^2 + r_3^2) - (a_2^2 + d_4^2 + \rho_6^2)}{2a_2} \tag{111}$$

As noted, the inequality constraint $\rho_1^2 + d_4^2 \geq \rho_7^2$ constitutes another limitation on the working space of the manipulator.

Expanding $\sin\theta_{23}$ and $\cos\theta_{23}$ in Equation (107) and (108) and collecting the relevant terms as coefficients of $\sin\theta_2$ and $\cos\theta_2$, the following equations can be obtained.

$$\rho_1 = \rho_8 \cos\theta_2 + \rho_9 \sin\theta_2 \tag{112}$$

$$r_3 = \rho_9 \cos\theta_2 - \rho_8 \sin\theta_2 \tag{113}$$

Here,

$$\rho_8 = a_2 + d_4 \sin\theta_3 - \rho_6 \cos\theta_3 \tag{114}$$

$$\rho_9 = d_4 \cos\theta_3 + \rho_6 \sin\theta_3 \tag{115}$$

According to T4 in Appendix C, Equation (112) and (113) give θ_2 as follows, provided that $\rho_8^2 + \rho_9^2 \neq 0$:

$$\theta_2 = \text{atan2}\,(\rho_9\rho_1 - \rho_3 r_3\,,\, \rho_9 r_3 - \rho_8\rho_1) \tag{116}$$

If $\rho_8^2 + \rho_9^2 = 0$, the elbow singularity occurs. Then, θ_2 becomes arbitrary with the same consequences as those of the subgroup 1.1.

Note that the matrix $\hat{M}_1 = e^{-\tilde{u}_2\theta_{23}} e^{-\tilde{u}_3\theta_1} \hat{C}_1$ of this subgroup comes out to be a function of θ_4 because the angles θ_1 and $\theta_{23} = \theta_2 + \theta_3$ are determined above as functions of θ_4. Therefore, the equation for the parametrized joint variable θ_4 is nothing but Equation (83), which is written here again as

$$\theta_4 = f_4(\theta_4) = \text{atan2}\,[\sigma_5 m_{23}(\theta_4),\, \sigma_5 m_{13}(\theta_4)] \quad \text{and} \quad \sigma_5 = \pm 1 \tag{117}$$

As noticed, Equation (117) is a highly complicated equation for the unknown θ_4 and it can be solved only with a suitable numerical method. However, after it is solved for θ_4, by substituting θ_4 into the previously derived equations for the other joint variables, the complete solution is obtained. Here, it is worth to mention that, although this solution is not completely analytical, it is still capable of giving the multiple and singular configurations as well as the working space limitations.

Although the PJV method is demonstrated above as applied to the subgroup 1.7, it can be applied similarly to the other subgroups that require it. For example, as a detailed case study, its quantitatively verified application to the FANUC ArcMate Sr. robot of the subgroup 1.9 can be seen in (Balkan et al. 1997 and 2000).

4.2 Inverse Kinematics of Subgroup 4.4

The inverse kinematic solution for the subgroup 4.4 is obtained in a similar manner and the related equations are given in Table 6 indicating the multiple solutions by $\sigma_i = \pm 1$. The orientation matrix $\hat{C}_4$ is simplified using the kinematic properties of this subgroup and denoted as $\hat{C}_{44}$. Actually, the Unimate 4000 manipulator of this subgroup does not have two versions with $\theta_3 = 0°$ and $\theta_3 = 90°$ as given below. It has simply $\theta_3 = 0°$ and the other configuration is a fictitious one. However, aside from constituting an additional example for the PJV method, this fictitious manipulator also gives a design hint for choosing the structural parameters so that the advantage of having a closed-form inverse kinematic solution is not lost.

Orientation Matrix, $\hat{C}_{44}$	$\check{C}_{44} = e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_{24}} e^{\tilde{u}_3\theta_3} e^{\tilde{u}_2\theta_6} e^{-\tilde{u}_1\pi/2}$ for $\theta_3 = 0°$ $\hat{C}_{44} = e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} e^{-\tilde{u}_1\theta_4'} e^{-\tilde{u}_2\theta_5'} e^{\tilde{u}_3\theta_6}$ for $\theta_3 = 90°$ Here, $\theta_4' = \theta_4 + 90°$, $\theta_5' = \theta_5 + 90°$	
Modified Orientation Matrix, $\hat{M}_{44}$	$e^{-\tilde{u}_3\theta_1} \hat{C}_{44} e^{\tilde{u}_1\pi/2} = \hat{M}_{44} = e^{\tilde{u}_2\theta_{24}} e^{\tilde{u}_3\theta_5} e^{\tilde{u}_2\theta_6}$ for $\theta_3 = 0°$ $e^{-\tilde{u}_2\theta_2} e^{-\tilde{u}_3\theta_1} \hat{C}_{44} = \hat{M}_{44} = e^{-\tilde{u}_1\theta_4'} e^{-\tilde{u}_2\theta_5'} e^{\tilde{u}_3\theta_6}$ for $\theta_3 = 90°$	
θ_3 Selection	$\theta_3 = 0°$	$\theta_3 = 90°$
Solution Method	CF	PJV with θ_4
Wrist Orientation Sequence	2-3-2	1-2-3
Solved Joint Variables from $\hat{M}_{44}$	$\theta_{24} = \text{atan2}(\sigma_5 m_{32}, -\sigma_5 m_{12})$ $\theta_5 = \text{atan2}(\sigma_5\sqrt{1-m_{22}^2}, m_{22})$ $\theta_6 = \text{atan2}(\sigma_5 m_{23}, \sigma_5 m_{21})$	$\theta_4' = \text{atan2}(\sigma_5 m_{23}, \sigma_5 m_{33})$ $\theta_5' = \text{atan2}(-m_{13}, \sigma_5\sqrt{1-m_{13}^2})$ $\theta_6 = \text{atan2}(-\sigma_5 m_{12}, \sigma_5 m_{11})$
Wrist Singularity	$\theta_5 = 0$	$\theta_5 = \pm90°$
Wrist Point Position Vector $\bar{r}_{44}$	$\bar{r}_{44} = a_2 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} \bar{u}_1 + s_3 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} \bar{u}_3 + d_5 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} e^{\tilde{u}_3\theta_3} e^{\tilde{u}_2\theta_4} \bar{u}_3$ $\bar{r}_{44} = a_2 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} \bar{u}_1 + s_3 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} \bar{u}_3 + d_5 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_{24}} \bar{u}_3$ for $\theta_3 = 0°$ $\bar{r}_{44} = a_2 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} \bar{u}_1 + s_3 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} \bar{u}_3 - d_5 e^{\tilde{u}_3\theta_1} e^{\tilde{u}_2\theta_2} e^{-\tilde{u}_1\theta_4'} \bar{u}_2$ for $\theta_3 = 90°$	
Scalar Wrist Point Equations	$r_1\cos\theta_1 + r_2\sin\theta_1 = \begin{cases} a_2\cos\theta_2 + s_3\sin\theta_2 + d_5\sin\theta_{24} & \text{for } \theta_3 = 0° \\ a_2\cos\theta_2 + (s_3 + d_5\cos\theta_4)\sin\theta_2 & \text{for } \theta_3 = 90° \end{cases}$ $r_2\cos\theta_1 - r_1\sin\theta_1 = \begin{cases} 0 & \text{for } \theta_3 = 0° \\ d_5\sin\theta_4 & \text{for } \theta_3 = 90° \end{cases}$ $r_3 = \begin{cases} -a_2\sin\theta_2 + s_3\cos\theta_2 + d_5\cos\theta_{24} & \text{for } \theta_3 = 0° \\ -a_2\sin\theta_2 + (s_3 + d_5\cos\theta_4)\cos\theta_2 & \text{for } \theta_3 = 90° \end{cases}$	
θ_3 Selection	$\theta_3 = 0°$	$\theta_3 = 90°$
Used T# Equations of Appendix C	T1 for θ_1, T2 for s_3, and T4 for θ_2	T3 for θ_1, T2 for s_3, and T4 for θ_2
Solved Joint Variables	$\theta_1 = \text{atan2}(\sigma_1 r_2, \sigma_1 r_1)$ $s_3 = \sqrt{\rho_{10}^2 + \rho_{11}^2 - a_2^2}$ $(s_3 > 0)$ $\theta_2 = \text{atan2}(\rho_2', \rho_2)$ $\rho_2 = s_3\rho_{11} + a_2\rho_{10})$ $\rho_2' = s_3\rho_{10} - a_2\rho_{11}$ $\theta_4 = \theta_{24} - \theta_2$ $\rho_{10} = \rho_1 - d_5\sin\theta_{24}$ $\rho_{11} = r_3 - d_5\cos\theta_{24}$ $\rho_1 = r_1\cos\theta_1 + r_2\sin\theta_1$	$\theta_1 = \text{atan2}(-r_1, r_2)$ $+ \sigma_1\text{atan2}(\sqrt{r_1^2 + r_2^2 - \rho_{12}^2}, \rho_{12})$ $s_3 = \sigma_3\sqrt{\rho_1^2 + r_3^2 - a_2^2} - \rho_{13}$ $(s_3 > 0)$ $\theta_2 = \text{atan2}(\rho_2', \rho_2)$ $\rho_2 = (s_3 + \rho_{13})r_3 + a_2\rho_1$ $\rho_2' = (s_3 + \rho_{13})\rho_1 - a_2 r_3$ Here, $\rho_{12} = d_5\sin\theta_4$ $\rho_{13} = d_5\cos\theta_4$
Shoulder Singularity	$r_1 = r_2 = 0$	$r_1 = r_2 = 0$ together with $\theta_4 = 0$ or $\theta_4 = \pm180°$

Table 6. Inverse Kinematic Solution for Subgroup 4.4

4.3 Inverse Kinematics of Manipulators with Closed Kinematic Chains

The method of inverse kinematics presented here is not limited to the serial manipulators only. It can also be applied to robots with a main open kinematic chain supplemented with auxiliary closed kinematic chains for the purpose of motion transmission from the actuators kept close to the base. As a typical example, the ABB IRB2000 industrial robot is selected here in order to demonstrate the application of the method to such manipulators. The kinematic sketch of this manipulator with its four-link transmission mechanism is shown in Figure 2. It can be seen from the kinematic sketch that the four-link mechanism can be considered in a sense as a satellite of the manipulator's main open kinematic chain. In other words, its relative position with respect to the main chain is determined completely by the angle θ_3. Once θ_3 is found by the inverse kinematic solution, the angular position of the third joint actuator ϕ_3 can be determined in terms of θ_3 as follows by considering the kinematics of the four-link mechanism:

$$\phi_3 = \psi_2 + \theta_2 \tag{118}$$

Here,

$$\psi_2 = \text{atan2}(b, a) + \sigma_3 \text{atan2}(\sqrt{a^2 + b^2 + c^2}, c) \quad \text{and} \quad \sigma_3 = \pm 1 \tag{119}$$

$$a = a_2 + b_4 \sin\theta_3, \quad b = b_4 \cos\theta_3, \quad c = \frac{a_2^2 + b_2^2 + b_4^2 - b_3^2}{2b_2} - \frac{a_2 b_4}{b_2} \sin\theta_3 \tag{120}$$

a) Complete Kinematic Structure   b) Closed Kinematic Chain Details
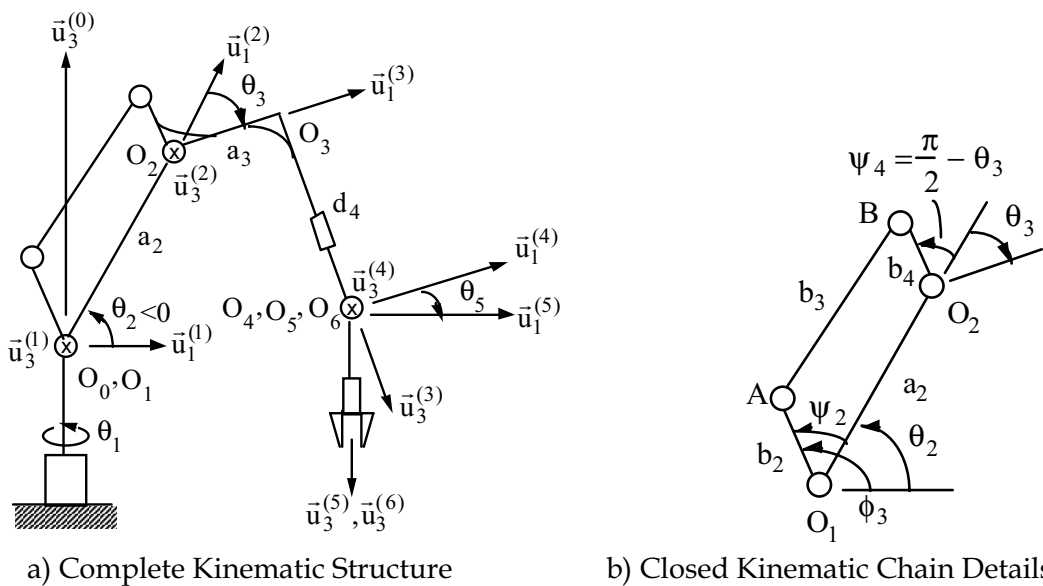
Figure 2. Kinematic Sketch of the ABB IRB2000 Manipulator

However, in this particular manipulator, the four-link mechanism happens to be a parallelogram mechanism so that $\psi_2 = \psi_4$ and $\phi_3 = \theta_2 + \dfrac{\pi}{2} - \theta_3$. Note that, if the auxiliary closed kinematic chain is separated from the main open kinematic chain, then this manipulator becomes a member of the subgroup 1.4 and the pertinent inverse kinematic solution can be obtained in closed-form similarly as done for the subgroup 1.1.

4.4 Comments on the Solutions

The inverse kinematic solutions of all the subgroups given in Table 4 are obtained. In main group 1, subgroup 1.2 has parameter d_2 in excess when compared to subgroup 1.1. This has an influence only in the solution of θ_1. The remaining joint variable solutions are the same. Similarly subgroup 1.3 has parameter a_1 and subgroup 1.5 has d_{23} ($d_2 + d_3$) in excess when compared to subgroup 1.1. Considering subgroup 1.3, only the solutions of θ_2 and θ_3 are different than the solution of subgroup 1.1, whereas the solution of subgroup 1.5 is identical to the solution of subgroup 1.2 except that d_{23} is used in the formulas instead of d_2. Subgroup 1.6 has parameter a_1 in excess compared to subgroup 1.4. Thus, the solutions of θ_2 and θ_3 are different than the solution of subgroup 1.4. Subgroup 1.8 has parameter d_2 and subgroup 1.9 has a_1 and a_3 in excess when compared to subgroup 1.7. Considering subgroup 1.8, only the solution of θ_1 is different. For subgroup 1.9, a_1 and a_3 causes minor changes in the parameters defined in the solutions of θ_2 and θ_3. The last subgroup, that is subgroup 1.10 has the parameters a_1, a_3 and d_3 in excess when compared to subgroup 1.7. θ_1, θ_2 and θ_3 have the same form as they have in the solution of subgroup 1.7, except the minor changes in the parameters defined in the solutions. It can be concluded that d_2 affects the solution of θ_2 and a_1 affects the solutions of θ_2 and θ_3 through minor changes in the parameters defined in the solution. In main group 2, subgroup has parameter d_2 in excess when compared to subgroup 2.3 and thus the solution of θ_1 has minor changes. Subgroup 2.5 has parameter a_1 in excess when compared to subgroup 2.4 and the solutions of θ_2 and θ_3 have minor changes. Subgroup 2.6 has parameter d_4 in excess and the term including it is identical to the term including d_2 in subgroup 2.5 except θ_1 which includes d_4 instead of d_2. In main group 8, subgroup 8.2 has the parameter a_2 in excess compared to subgroup 8.1. This leads to a minor change in the solutions of θ_1 and θ_2 through the parameters defined in the solution.

For main group 1, if θ_1 is obtained analytically θ_4, θ_5 and θ_6 can be solved in closed-form. Any six-joint manipulator belonging to main group 2 can be solved in closed-form provided that θ_1 is obtained in closed-form. Using θ_1, θ_{234} can easily be determined using the orientation matrix equation. Since θ_4 appears in the terms including a_4, d_5 and a_5 as θ_{234}, this lead to a complete

closed-form solution. In main group 3, directly obtaining θ_1 and θ_2 analytically results in a complete closed-form solution provided that the kinematic parameter $a_5 = 0$. In main group 6, even there is the offset d_5, a complete closed-form solution can be obtained since the term of d_5 does not include θ_4. Moreover, if θ_1 can be obtained in closed-form and a_5 is a nonzero kinematic parameter, simply solving θ_{345} from $\hat{C}$ leads to a complete closed-form solution. Among all the main groups, main group 5 has the least complicated equations. Joint variable θ_{1234} is directly obtained from $\hat{C}$ and if d_5 or a_4 are kinematic parameters of a manipulator belonging to this main group, the whole solution will be in closed-form. The offset in main group 9 does not lead to a PJV solution since it does not include θ_4 in the term including it. Also θ_1 even if a_5 is a nonzero kinematic parameter, a closed-form solution can be obtained using θ_{45} provided that θ_1 is obtained analytically. Since α_4 of main groups 6 and 9 is 0°, θ_4 does not appear in the term including offset d_5. The kinematic parameter a_5 does not appear in any of the subgroups, so it can be concluded that this parameter might appear only in some very specific six-joint manipulators. On the other hand, the more the number of prismatic joints, the easier the solution is, since the joint angles become constant for prismatic joints.

5. Conclusion

In this chapter, a general approach is introduced for a classification of the six-joint industrial robotic manipulators based on their kinematic structures, and a complete set of compact kinematic equations is derived according to this classification. The algebraic tools based on the properties of the exponential rotation matrices have been very useful in simplifying the kinematic equations and obtaining them in compact forms. These compact kinematic equations can be used conveniently to obtain the inverse kinematic solutions either analytically in closed forms or semi-analytically using parametrized joint variables. In either case, the singular and multiple configurations together with the working space limitations are also determined easily along with the solutions.

Moreover, both types of these inverse kinematic solutions provide much easier programming facilities and much higher on-line application speeds compared to the general manipulator-independent but purely numerical solution methods.

On the other hand, although the classification based method presented here is naturally manipulator-dependent, it is still reasonable and practical to use for the inverse kinematic solutions, because a manipulator having all non-zero kinematic parameters does not exist and it is always possible to make a considerable amount of simplification on the kinematic equations before attempting to solve them.

6. References

Balkan, T.; Özgören, M. K., Arıkan, M. A. S. & Baykurt M. H. (1997). A Method of Inverse Kinematics Solution for a Class of Robotic Manipulators, *CD-ROM Proceedings of the ASME 17th Computers in Engineering Conference*, Paper No: DETC97/CIE-4283, Sacramento, CA.

Balkan, T.; Özgören, M. K., Arıkan, M. A. S. & Baykurt, M. H. (1999). A Kinematic Structure Based Classification of Six-DOF Industrial Robots and a Method of Inverse Kinematic Solution, *CD-ROM Proceedings of the ASME 1999 Design Engineering Technical Conference*, Paper No: DETC99/DAC-8672, Las Vegas, Nevada.

Balkan, T.; Özgören, M. K., Arıkan, M. A. S. & Baykurt, M. H. (2000). A Method of Inverse Kinematics Solution Including Singular and Multiple Configurations for a Class of Robotic Manipulators, *Mechanism and Machine Theory*, Vol. 35, No. 9, pp. 1221-1237.

Balkan, T.; Özgören, M. K., Arıkan, M. A. S. & Baykurt, M. H. (2001). A Kinematic Structure-Based Classification and Compact Kinematic Equations for Six-DOF Industrial Robotic Manipulators, *Mechanism and Machine Theory*, Vol. 36, No. 7, pp. 817-832.

Denavit, J. & Hartenberg, R. S. (1955). A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices, *ASME Journal of Applied Mechanics*, pp. 215-221.

Hunt, K. H. (1986). The Particular or the General (Some Examples from Robot Kinematics), *Mechanism and Machine Theory*, Vol. 21, No. 6, pp. 481-487.

Lee H. Y.; Woernle, C. & Hiller M. (1991). A Complete Solution for the Inverse Kinematic Problem of the General 6R Robot Manipulator, *Transactions of the ASME, Journal of Mechanical Design*, Vol. 13, pp. 481-486.

Manseur, R. & Doty, K. L. (1996). Structural Kinematics of 6-Revolute-Axis Robot Manipulators, *Mechanism and Machine Theory*, Vol. 31, No. 5, pp. 647-657.

Özgören, M. K. (1987). Application of Exponential Rotation Matrices to the Kinematic Analysis of Manipulators, *Proceedings of the 7th World Congress on the Theory of Machines and Mechanisms*, pp. 1187-1190, Seville, Spain.

Özgören, M. K. (1994). Some Remarks on Rotation Sequences and Associated Angular Velocities, *Mechanism and Machine Theory*, Vol. 29, No. 7, pp. 933-940.

Özgören, M. K. (1995). Position and Velocity Related Singularity Analysis of Manipulators, *Proceedings of the 9th World Congress on the Theory of Machines and Mechanisms*, Milan, Italy.

Özgören, M. K. (1999). Kinematic Analysis of a Manipulator with its Position and Velocity Related Singular Configurations, *Mechanism and Machine Theory*, Vol. 34, No. 7, pp. 1075-1101.

Özgören, M. K. (2002). Topological Analysis of 6-Joint Serial Manipulators and Their Inverse Kinematic Solutions, *Mechanism and Machine Theory*, Vol. 37, pp. 511-547.

Pieper D. L. & Roth B. (1969). The Kinematics of Manipulators under Computer Control, *Proceedings of the 2nd International Congress for the Theory of Machines and Mechanisms*, pp. 159–168, Zakopane, Poland.

Raghavan, B. & Roth, B. (1993). Inverse Kinematics of the General 6R Manipulator and Related Linkages, *Transactions of the ASME, Journal of Mechanical Design*, Vol. 115, pp. 502-508.

Wampler, C. W. & Morgan, A. (1991). Solving the 6R Inverse Position Problem Using a Generic-Case Solution Methodology, *Mechanism and Machine Theory*, Vol. 6, Vol. 1, pp. 91-106.

Wu, C. H. & Paul, R. P. (1982). Resolved Motion Force Control of Robot Manipulator, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC 12, No. 3, pp. 266-275.

Appendix A

Exponential Rotation Matrix Simplification Tools
(Özgören, 1987-2002; Balkan et al., 2001)

E.1 : $(e^{\tilde{u}_i \theta_k})^{-1} = e^{-\tilde{u}_i \theta_k}$

E.2 : $e^{\tilde{u}_i \theta_k} \bar{u}_i = \bar{u}_i$

E.3 : $\bar{u}_i{}^T e^{\tilde{u}_i \theta_k} = \bar{u}_i{}^T$

E.4 : $e^{\tilde{u}_i \theta_j} e^{\tilde{u}_i \theta_k} = e^{\tilde{u}_i (\theta_j + \theta_k)} = e^{\tilde{u}_i \theta_{jk}}$

E.5 : $e^{\tilde{u}_i \pi} = e^{-\tilde{u}_i \pi}$

E.6 : $e^{\tilde{u}_i \pi/2} e^{\tilde{u}_j \theta} e^{-\tilde{u}_i \pi/2} = e^{\tilde{n}_{ij} \theta}$ where $\bar{n}_{ij} = \tilde{u}_i \bar{u}_j$.

E.7 : $e^{\tilde{u}_i \theta_k} e^{\tilde{u}_j \pi} = e^{\tilde{u}_j \pi} e^{-\tilde{u}_i \theta_k}$

E.8 : $e^{\tilde{u}_i \theta_k} \bar{u}_j = \bar{u}_j \cos\theta + (\tilde{u}_i \bar{u}_j)\sin\theta$ for $i \neq j$

E.9 : $\bar{u}_j{}^T e^{\tilde{u}_i \theta_k} = \bar{u}_j{}^T \cos\theta + (\tilde{u}_j \bar{u}_i)^T \sin\theta$ for $i \neq j$

E.10 : $e^{-\tilde{u}_i \theta_k} e^{\tilde{u}_i \theta_k} = e^{\tilde{u}_i 0} = \hat{I}$

Appendix B

List of Six-joint Industrial Robots Surveyed (Balkan et al., 2001)

Robot Name	Non-Zero Link Parameters	Subgroup	Robot Name	Non-Zero Link Parameters	Subgroup
ABB IRB 1400	a_1, a_2, a_3, d_4	1.6	Kawasaki UT120	a_1, a_2, a_3, d_4	1.6
ABB IRB 1400H	a_1, a_2, a_3, d_4	1.6	Kawasaki UT150	a_1, a_2, a_3, d_4	1.6
ABB IRB 2400	a_1, a_2, a_3, d_4	1.6	Kawasaki UX100	a_2, a_3, d_4	1.4
ABB IRB 2400L	a_1, a_2, a_3, d_4	1.6	Kawasaki UX120	a_2, a_3, d_4	1.4
ABB IRB 4400	a_1, a_2, a_3, d_4	1.6	Kawasaki UX150	a_2, a_3, d_4	1.4
ABB IRB 4400L	a_1, a_2, a_3, d_4	1.6	Kawasaki UX70	a_2, a_3, d_4	1.4
ABB IRB 6400	a_1, a_2, a_3, d_4	1.6	KUKA IR 662/10	a_2, d_4	1.1
ABB IRB4400FS	a_1, a_2, a_3, d_4	1.6	Limanat RT-280	a_2, a_3, d_4	2.1
ABB IRB6400PE	a_1, a_2, a_3, d_4	1.6	Mack BASE Rob	$a_1, a_2, s_1, s_2, s_3, d_4$	4.1
Acrobe AG-4	a_2, a_3, a_4	2.2	Motoman J10	a_2, d_4	1.1
AKR 3000	a_2, d_4, d_5	1.7	Motoman L3	a_2, a_3, d_3	2.3
Altınay ASR-60	a_2, a_3, d_4	1.4	Motoman SK-10	a_1, a_2, a_3, d_4	1.6
Altınay EG-6	$a_1, a_2, a_3, a_4, d_4, d_5$	9.1	Motoman SK-16	a_1, a_2, a_3, d_4	1.6
Altınay EV-6	a_1, a_2, a_3, d_4	1.6	Motoman SK-30	a_1, a_2, a_3, d_4	1.6
Altınay HSR-6	a_2, d_4	1.1	Motoman SK-45	a_1, a_2, a_3, d_4	1.6
Anorad-Anorobot	$a_1, a_3, s_1, s_2, s_3, d_4$	3.2	Motoman SK-6	a_1, a_2, a_3, d_4	1.6
ATI Robo-Master	a_2, a_3, a_4	2.2	Motoman SK120	a_1, a_2, a_3, d_4	1.6
Bendix AA670	a_1, a_2, d_4	1.3	MotomanSK16-6	a_1, a_2, a_3, d_4	1.6
Binks 88-800	a_2, d_4	1.1	MTS 200A	a_2, a_3, d_4	1.4
CM T3-566	a_2, a_3, a_4	2.2	Nachi Robot8601	a_2, d_4, d_5	1.7
CM T3-856	a_2, a_3, a_4	2.2	Northwest Tech.	a_2, a_3, a_4	2.2
CMV Inst. DDaII	a_1, a_2, d_3, d_4	8.2	Nordson Robot	a_2, d_4, d_5	1.7
Comau SmartS-2	$a_1, a_2, a_3, d_3, d_4, d_5$	1.10	Numan I&II	s_2, d_4	4.2
CRS A255	a_2, a_3, a_4	2.2	Pharemme BMR	a_1, a_2, d_4	1.3
CRS A465	a_2, d_4	1.1	Portque 80	s_1, s_2, s_3	3.1
Cyclomat Merlin	a_2, d_2, d_4	1.2	PUMA 200	a_2, d_2, d_3, d_4	1.5
Daihen LK	a_2, a_3, d_5	2.3	PUMA 260	a_2, d_2, d_3, d_4	1.5
FANUC S-420	a_1, a_2, a_3, d_4	1.6	PUMA 560	a_2, d_2, d_4	1.2
FanucArcMateSR	a_1, a_2, a_3, d_4, d_5	1.9	PUMA 562	a_2, d_2, d_4	1.2
GCA DKP 1000	a_2, d_4	1.1	PUMA 700	a_2, d_2, d_4	1.2
GCP 132	a_2, d_4	1.1	REIS RR625	a_1, a_2, d_3, d_4	8.2
GMF S-2L	a_2, d_4	1.1	Rhino Charger	a_2, d_4	1.1
GMF S-3L	a_2, d_4, d_5	1.7	SCARA Robot	a_1, a_2, s_3	5.1
GMF S-3R	a_2, d_4, d_5	1.7	Scheinman	a_2, s_3, d_5	3.4
GMF S-LR	a_2, d_4	1.1	Scorbot ERIX	a_1, a_2, a_3, d_2, d_5	2.5
Griffin	a_2, d_4	1.1	Scorbot PerfMK2	a_1, a_2, a_3, d_4, d_5	2.6
Intelledex 660	a_3, a_4	6.1	SEF-SR10	a_1, a_2, d_4	1.3
Kawasaki EE10	a_2, d_3, d_4, d_5	1.8	Stanford JPL	d_2, s_3	3.3
Kawasaki EH120	a_1, d_3, d_4	8.1	Staubli RX-line	a_2, d_4	1.1
Kawasaki JA-5	a_2, d_4	1.1	Staubli RX130	a_2, d_4	1.1
Kawasaki JC-5	a_2, d_4	1.1	Staubli RX60	a_2, d_4	1.1
Kawasaki JS-10	a_2, d_4, d_5	1.7	Staubli RX90	a_2, d_4	1.1
Kawasaki JS-30	a_2, d_4	1.1	TH8	a_2, s_2, s_3	7.1
Kawasaki JS-40	a_2, d_4	1.1	Thermwood 6	a_2, d_4	1.1
Kawasaki JS-5	a_2, d_4	1.1	Thermwood PM	a_2, d_4	1.1
Kawasaki JS-6	a_2, d_4	1.1	Toshiba TSR 500	a_2, a_3, d_2, d_5	2.4
Kawasaki U2100	a_2, a_3, d_4	1.4	Unimate 4000	a_2, s_1, d_5	4.4
Kawasaki U2120	a_2, a_3, d_4	1.4	Unimate Robot	s_3, d_5	4.3
Kawasaki U2150	a_2, a_3, d_4	1.4	V80	a_2, d_4	1.1
Kawasaki UT100	a_1, a_2, a_3, d_4	1.6	6P01	a_2, d_4	1.1

Appendix C

Solutions to Some Trigonometric Equations Encountered in Inverse Kinematic Solutions

T0a T0b T0c	$\sin\theta_i = a$ (alone) $\cos\theta_i = b$ (alone) $\sin\theta_i = a$, $\cos\theta_i = b$ (together)	$\theta_i = \text{atan2}\,(a,\, \sigma_i\sqrt{1-a^2})$; $\sigma_i = \pm 1$ $\theta_i = \text{atan2}\,(\sigma_i\sqrt{1-b^2},\, b)$; $\sigma_i = \pm 1$ $\theta_i = \text{atan2}\,(a,\, b)$
T1	$a\cos\theta_i - b\sin\theta_i = 0$	$\theta_i = \text{atan2}\,(\sigma_i a,\, \sigma_i b)$; $\sigma_i = \pm 1$ if $a = b = 0$ singularity occurs
T2	$s_j\cos\theta_j = b$, $s_j\sin\theta_j = a$	$s_i = \sigma_i\sqrt{a^2+b^2}$; $\sigma_i = \pm 1$ $\theta_i = \text{atan2}\,(\sigma_i a,\, \sigma_i b)$ if $s_i = 0$ singularity occurs
T3	$a\cos\theta_i + b\sin\theta_i = c$	$\theta_i = \text{atan2}\,(b,a) + \sigma_i\text{atan2}\,(\sqrt{a^2+b^2-c^2},\, c)$ $\sigma_i = \pm 1$; if $a = b = 0$ singularity occurs
T4	$a\cos\theta_i - b\sin\theta_i = c$ $a\sin\theta_i + b\cos\theta_i = d$	$\theta_i = \text{atan2}\,(ad-bc,\, ac+bd)$ provided that $a^2+b^2 = c^2+d^2$ and $a^2+b^2 \neq 0$ if $a^2+b^2 = 0$ singularity occurs
T5	$a\cos\theta_i + b\cos\theta_{ij} = c$ $a\sin\theta_i + b\sin\theta_{ij} = d$	$\theta_j = \text{atan2}\,(\sigma_j\sqrt{1-e^2},\, e)$; $\sigma_j = \pm 1$ $e = \dfrac{(c^2+d^2)-(a^2+b^2)}{2ab}$ and $\theta_{ij} = \theta_i + \theta_j$
T6	$a\cos\theta_i + b\sin\theta_{ij} = c$ $-a\sin\theta_i + b\cos\theta_{ij} = d$	$\theta_j = \text{atan2}\,(e,\, \sigma_j\sqrt{1-e^2})$; $\sigma_j = \pm 1$ $e = \dfrac{(c^2+d^2)-(a^2+b^2)}{2ab}$ and $\theta_{ij} = \theta_i + \theta_j$
T7	$a\cos\theta_i + b\sin\theta_i = c + d\cos\theta_j$ $b\cos\theta_i - a\sin\theta_i = d\sin\theta_j$	$\theta_j = \text{atan2}\,(\sigma_j\sqrt{1-e^2},\, e)$; $\sigma_j = \pm 1$ $e = \dfrac{(a^2+b^2)-(c^2+d^2)}{2cd}$
T8	$a\cos\theta_i + b\sin\theta_i = c - d\sin\theta_j + e\cos\theta_j$ $b\cos\theta_i - a\sin\theta_i = d\sin\theta_j + e\cos\theta_j$	$\theta_j = \text{atan2}\,(-d,\, e) + \sigma_j\text{atan2}\,(\sqrt{d^2+e^2-f^2},\, f)$ $f = \dfrac{(a^2+b^2)-(c^2+d^2+e^2)}{2c}$; $\sigma_j = \pm 1$
T9	$a\sin\theta_{ij} + b\cos\theta_{ij} + d\cos\theta_i = e$ $-a\cos\theta_{ij} + b\sin\theta_{ij} + d\sin\theta_i = f$	$\theta_j = \text{atan2}\,(a,b) + \sigma_j\text{atan2}\,(\sqrt{a^2+b^2-g^2},\, g)$ $g = \dfrac{(e^2+f^2)-(a^2+b^2+d^2)}{2d}$; $\sigma_j = \pm 1$

The two-unknown trigonometric equations T5-T8 and T9 become similar to T4 and T0c respectively once θ_j is determined from them as indicated above. Then, θ_i can be determined as described for T4 or T0c.

Inverse Position Procedure for Manipulators with Rotary Joints

Ibrahim A. Sultan

1. Introduction

Industrial robot manipulators are essentially spatial linkages that consist of rigid bodies connected by joints. Even though many types of joints (which are also known as kinematic pairs) are available for use in mechanical linkages, only two types are employed for robot manipulators. These are the revolute, or rotary, joints (referred to in literature as R) and the prismatic, or sliding, joints (referred to as P). These specific types allow a single degree of freedom relative movement between adjacent bodies; and are easier to drive and control than other kinematic pairs. Normally every joint on the manipulator is independently driven by a dedicated motor. It is central to kinematic control of manipulators to calculate the sets of joint-motor displacements which correspond to a desired pose (i.e. position and orientation) at the end-effector. The mathematical procedure which is followed to achieve this purpose is often referred to as, *Inverse Position Analysis*. This analysis presents a special difficulty in the field of Robotics as it is associated with the use of intricate spatial geometry techniques. The complexity of the analysis increases substantially with the number of rotary joints on the manipulator structure. For this reason a considerable part of the published literature is mainly concerned with the revolute-joint manipulators.

Published literature reveals that various methods have been proposed to solve the inverse position problem of manipulators. These methods range from Jacobian-based iterative techniques to highly sophisticated levels of equation-manipulation intended to reduce the whole model into a polynomial with thousands of mathematical terms. However, most industrial robots are designed with geometric features (such as parallelism and perpendicularity) to make it possible for simple inverse position solutions to be obtained in closed forms suitable for real time control. Another geometric aspect that leads to simplified inverse solutions is the spherical wrist design, which entails that the last three joints on the manipulator structure intersect at one point. This usually suggests that these three joints (also known as the wrist joints) have the main task of orienting (rather than placing) the end-effector in space. In this

case it should be possible to regard the manipulator as consisting of two separate parts where the first part (referred to as the arm) consists of the first three joints, counting from the stationary base, on the structure. The task of the arm is to place the end-effector origin (i.e. the point of intersection of the last three joint axes) at a defined point in space. The solution for this first part can be obtained separately before proceeding to find the angles of the last three joints which will result in giving the end-effector its desired spatial orientation. The work presented in this paper adopts this strategy to propose a mathematical procedure, for the arm inverse solution, based on assigning local coordinates at every joint, and utilising the properties of rotation to relate these coordinates. A model manipulation technique is then employed to obtain the arm inverse solution in terms of one polynomial. A kinematic synthesis discussion is then presented for the arm structure in terms of local coordinates to reflect on the number of solutions expected from the polynomial. It will be shown that the concept of intersecting spatial circles offers a good ground to comprehend the kinematics of revolute-joint manipulators. Moreover, models are presented for the wrist structure to obtain a full inverse kinematic solution for the robot manipulator. A solved example is demonstrated to prove the validity of the method presented.

2. Literature Survey

Published literature reveals that the homogeneous transformation matrix which was developed as far back as 1955 has extensively been employed for the analysis of robot manipulators. The matrix involves the use of four parameters, usually referred to as the DH-parameters, intended to perform transformation between two spatial Cartesian coordinate systems (Denavit and Hartenberg, 1955). Recently, other kinematic models have been proposed by researchers to deal with the drawbacks of the DH presentation (Sultan and Wager, 1999). This is particularly important if the model is going to be implemented for robot calibration purposes. The theory of dual-number algebra was introduced into the field of kinematics back in the 1960's (Yang and Freudenstein, 1964); and it did appeal to researchers in the field of robot kinematics (Pennock and Yang, 1985; Gu and Luh, 1987; Pardeep *et al*, 1989). In addition to these approaches, which are based on matrices, vector methods were also employed in the field of kinematic analysis of robots (Duffy, 1980; Lee and Liang, 1988A and 1988B).

Many industrial robots possess parallel and intersecting joint-axes and their direct-position models can be inverted analytically such that closed-form solutions may be obtained for the joint-displacements (Gupta, 1984; Pennock and Yang, 1985; Pardeep *et al*, 1989; Wang and Bjorke, 1989).

Spherical-wrist manipulators have their last three joint-axes intersecting at a common point. For these manipulators the position of the end-effector in space is determined only by the displacements performed about the first three joint-axes. This concept is often referred to as the position-orientation decoupling; and has been utilised to produce a closed form solution, for the inverse position problem of simple structure robots, efficient enough to be implemented for computer control (Pieper and Roth, 1969). Inverse position techniques have been proposed to utilise the position-orientation decoupling of industrial robot of arbitrarily directed axes (Sultan, 2000; Sultan and Wager, 2001). As such these techniques do not rely on any particular spatial relations (e.g. parallelism or perpendicularity) between the successive joint-axes. In fact, approaches which utilise these particular geometric features to produce the model equations are likely to produce positioning errors when used for robot control since the actual structures always deviate from their intended ideal geometry.

Iterative techniques have been employed for the inverse position analysis of general robot manipulators. Many of these techniques involve the computation of a Jacobian matrix which has to be calculated and inverted at every iteration. The solution in this case may be obtained by a Newton-Raphson technique (Hayati and Reston, 1986) or a Kalman filter approach (Coelho and Nunes, 1986). However, the inversion of the system Jacobian may not be possible near singular configurations (where the motion performed about one joint-axis produces exactly the same effect, at the end-effector, as the motion performed about another axis, hence resulting in loss of one or more degrees of freedom). Therefore, a singularity avoidance approach has been reported where the technique of damped least-squares is used for the analysis (Chiaverini *et al*, 1994). However, this technique seems to be rather sluggish near singular points where extra computational procedure may have to be involved.

Optimisation techniques have also been employed to solve the inverse-position problem of manipulators whereby a six-element error vector was implemented for the analysis (Goldenberg *et al*, 1985). The vector combines the current spatial information (position and orientation) of the robot hand and compares it to the desired pose to produce error values. Published literature in the area of optimisation report a technique by which the robot is moved about one joint at a time to close an error gap (Mahalingam and Sharan, 1987; Wang and Chen, 1991; Poon and Lawrence, 1988). More recent research effort demonstrates valuable inputs form such areas as neural networks (Zhang *et al*, 2005) and fuzzy techniques (Her *et al*, 2002) to the field of robot inverse kinematics.

 It has been shown that the kinematic behaviour of robots can be described in terms of a set of polynomials that can be solved iteratively (Manseur and Doty 1992a, 1992b and 1996). One such method features a set of eight polynomials

which were solved numerically to obtain different possible solutions to the inverse position problem; it could therefore be concluded that the maximum number of meaningful solutions to the inverse position problem of a general robotic structure is 16 (Tsai and Morgan 1985), rather than 32 as had previously been suggested (Duffy and Crane, 1980). However it has been pointed out that a manipulator with 16 different real inverse position solutions can seldom be found in real life (Manseur and Doty, 1989). In reality most manipulators are designed to possess up to 8 solutions of which only one or two can be physically attained.

It is possible to express the inverse position problem of robots in terms of a 16 degree polynomial in the tan-half-angle of a joint-displacement (Lee and Liang 1988a & 1988b; Raghavan and Roth 1989). However it has been argued that the coefficients of such a polynomial are likely to contain too many terms which may render such a tack impractical to use (Smith and Lipkin 1990). Also, these high order polynomials are obtained by evaluating the eliminants of hyper-intricate determinants which may be impossible to handle symbolically in the first place. This may have motivated some researchers (Manocha and Canny 1992; Kohli and Osvatic 1993) to reformulate the solutions in terms of eigenvalue models in order to simplify the analysis and avoid numerical complications. However, a numerical technique has been introduced to obtain the inverse solutions without having to expand the system characteristic determinant (Sultan, 2002).

The procedure introduced here for the inverse position analysis of robot manipulators is described in the rest of this paper.

3. Rotation of Vectors

The unit vector $\hat{\mathbf{z}}_i$ in Figure (1) represents an axis of rotation in a spatial mechanism. It is required to obtain the new *rotated* vector, $\mathbf{v}_{ir}$, which results from rotating the *original* vector $\mathbf{v}_{io}$ (where $\mathbf{v}_{io} \times \hat{\mathbf{z}}_i \neq \mathbf{0}$) by an angle $\theta_i \hat{\mathbf{z}}_i$ In order to do so, the Cartesian system $\hat{\mathbf{x}}_i \hat{\mathbf{y}}_i \hat{\mathbf{z}}_i$ may be introduced as follows,

$$\hat{\mathbf{x}}_i \circ \hat{\mathbf{z}}_i = 0$$

$$\tag{1}$$

where $\left\| \hat{\mathbf{x}}_i \right\| = 1$.

Then

$$\hat{\mathbf{y}}_i = \hat{\mathbf{z}}_i \times \hat{\mathbf{x}}_i \tag{2}$$

The original vector, $\mathbf{v}_{io}$, and the rotated vector $\mathbf{v}_{ir}$, can both be expressed with respect to the $\hat{\mathbf{x}}_i \hat{\mathbf{y}}_i \hat{\mathbf{z}}_i$-frame in terms of local coordinates, *n, m* and *l* as follows,

$$\left.\begin{array}{l} \mathbf{v}_{io} = n_{io}\hat{\mathbf{x}}_i + m_{io}\hat{\mathbf{y}}_i + l_{io}\hat{\mathbf{z}}_i \\ \mathbf{v}_{ir} = n_{ir}\hat{\mathbf{x}}_i + m_{ir}\hat{\mathbf{y}}_i + l_{ir}\hat{\mathbf{z}}_i \end{array}\right\} \quad (3)$$
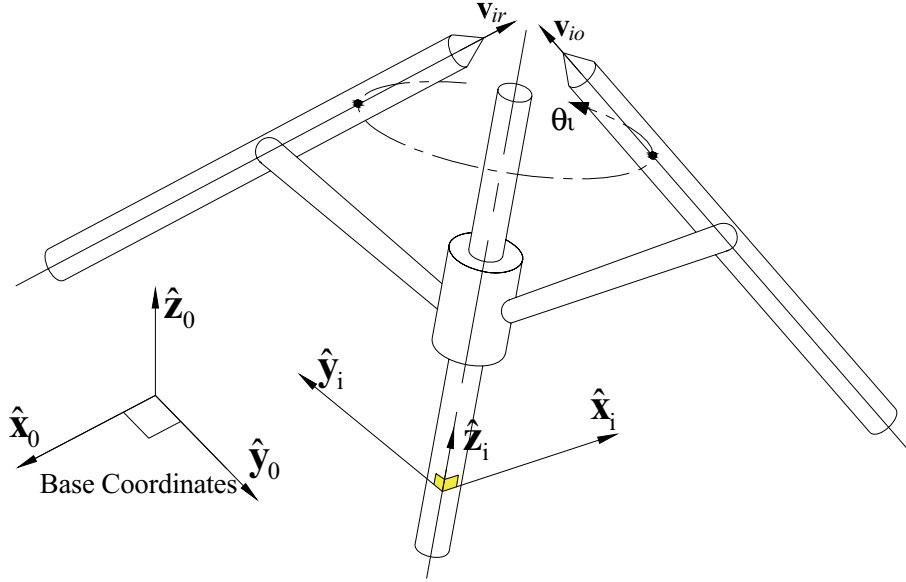
Figure 1. Rotation of Vectors.

where the local coordinates are given as follows;

$$\left.\begin{array}{l} n_{io} = \mathbf{v}_{io} \circ \hat{\mathbf{x}}_i \\ m_{io} = \mathbf{v}_{io} \circ \hat{\mathbf{y}}_i \\ l_{io} = \mathbf{v}_{io} \circ \hat{\mathbf{z}}_i \end{array}\right\} \quad (4)$$

$$\text{And} \quad \left.\begin{array}{l} n_{ir} = n_{io}\cos\theta_i - m_{io}\sin\theta_i \\ m_{ir} = m_{io}\cos\theta_i + n_{io}\sin\theta_i \\ l_{ir} = l_{io} \end{array}\right\} \quad (5)$$

The inverse of this problem is encountered when $\hat{\mathbf{z}}_i$, $\mathbf{v}_{io}$ and $\mathbf{v}_{ir}$ are all known and it is required to obtain the corresponding value of θ_i. With the values of the local coordinates known, θ_i could be obtained as follows,

$$\theta_i = \text{atan2}(m_{ir}n_{io} - n_{ir}m_{io}, n_{ir}n_{io} + m_{ir}m_{io}) \quad (6)$$

where the function atan2(y,x) is available in many computer algebra packages and compilers to compute the angle θ_i (over the range of the whole circle) when its sine and cosine are both given. In this paper, the concepts mentioned above are used together with the suitable conditions of rotation to perform the inverse position analysis of the manipulator arm and wrist. The proposed analysis for the arm is given in the next section.

4. Inverse Kinematics of the Arm

The arm, which is the largest kinematic part of the manipulator, consists of three revolute joints connected through rigid links. Each joint, as shown in Figure (2), is represented by the spatial pose of its axis. The first joint-axis has a fixed location and orientation in space as it represents the connection between the whole manipulator and the fixed frame. Any other joint-axis number i can float in space as it rotates about the joint-axis number i–1.

In the current context, the main function of the arm is to displace a certain spatial point from an initial known location to a required final position. In spherical-wrist manipulators, this point is at the intersection of the wrist axes. In a calibrated (non-spherical-wrist) manipulator, it may represent a point on the sixth axis as close as possible to the fifth joint-axis. In Figure (2), the arm is required to displace point pi to a final position pf. The position vectors, $\mathbf{pi}_b$ and $\mathbf{pf}_b$ respectively, of these two points are known with respect to the base coordinate system.

As per Appendix A, any joint-axis $\hat{\mathbf{z}}_i$ is related to the successive axis, $\hat{\mathbf{z}}_{i+1}$, through a common normal, $\hat{\mathbf{x}}_{i+1}$. This common normal is used to construct a local frame at the axis $\hat{\mathbf{z}}_{i+1}$ using the relation, $\hat{\mathbf{y}}_{i+1} = \hat{\mathbf{z}}_{i+1} \times \hat{\mathbf{x}}_{i+1}$. The shortest distance, a_{i+1}, between the axes, $\hat{\mathbf{z}}_i$ and $\hat{\mathbf{z}}_{i+1}$, is measured along $\hat{\mathbf{x}}_{i+1}$ which intersects $\hat{\mathbf{z}}_i$ at the point $\mathbf{p}_i$ and $\hat{\mathbf{z}}_{i+1}$ at the point $\mathbf{p}_{i(i+1)}$.

At the zero initial position which is shown in Figure (2), the axis $\hat{\mathbf{x}}_1$ is chosen to coincide with $\hat{\mathbf{x}}_2$. In this figure, the position vectors, $\mathbf{pi}_{3o}$ and $\mathbf{pf}_{1r}$, of points pi and pf respectively with respect to the frames $\hat{\mathbf{x}}_3\hat{\mathbf{y}}_3\hat{\mathbf{z}}_3$ and $\hat{\mathbf{x}}_1\hat{\mathbf{y}}_1\hat{\mathbf{z}}_1$ may be numerically calculated as follows,

$$\left.\begin{array}{l} \mathbf{pf}_{1r} = \mathbf{pf}_b - \mathbf{p}_1 \\ \mathbf{pi}_{3o} = \mathbf{pi}_b - \mathbf{p}_{32} \end{array}\right\} \tag{7}$$

where $\mathbf{p}_1$ and $\mathbf{p}_{32}$ are the position vectors of the axes-attached, points $\mathbf{p}_1$ and $\mathbf{p}_{32}$, respectively as measured from the origin of the base coordinates. Accord-

ing to the concepts in (4) and (5), $\mathbf{pf}_{1r}$ can be described with respect to the $\hat{\mathbf{x}}_1\hat{\mathbf{y}}_1\hat{\mathbf{z}}_1$-frame in terms of known local coordinates (n_{1r}, m_{1r} and l_{1r}). Also, $\mathbf{pi}_{30}$ can be described with respect to the $\hat{\mathbf{x}}_3\hat{\mathbf{y}}_3\hat{\mathbf{z}}_3$-frame in terms of known local coordinates (n_{3o}, m_{3o} and l_{3o}).
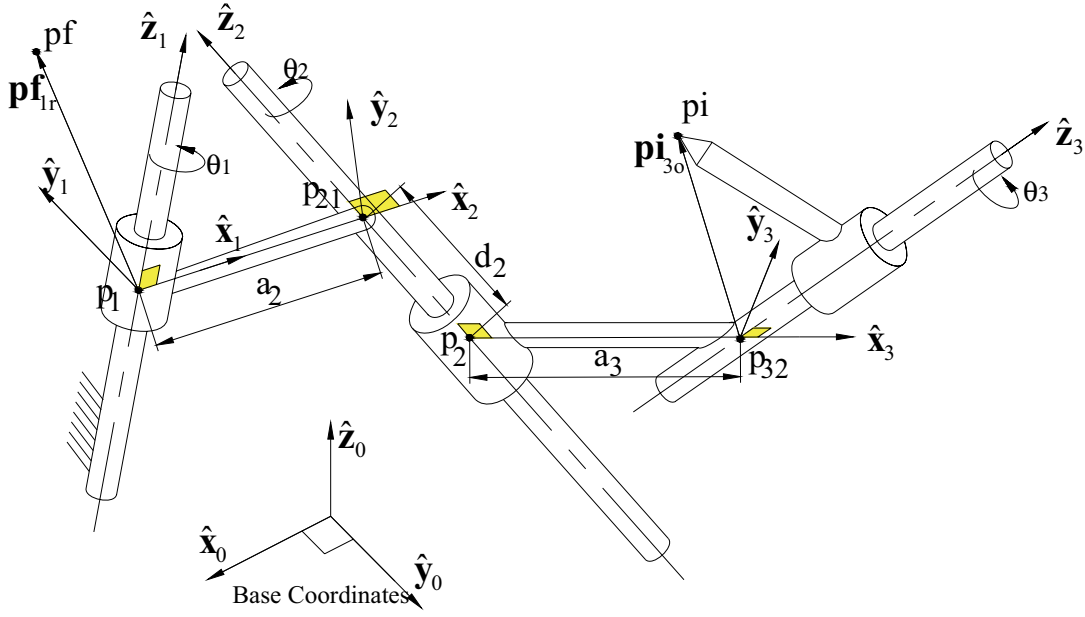
Figure 2. A General View of a 3R Manipulator Arm at Its Zero Position.

It is understood that the vector $\mathbf{pf}_{1r}$ resulted from rotating another vector $\mathbf{pf}_{1o}$ about the $\hat{\mathbf{z}}_i$ axis by an angle, θ_1 (i.e. a $\theta_1\hat{\mathbf{z}}_1$-type rotation). The original vector, $\mathbf{pf}_{1o}$, can be expressed with respect to the $\hat{\mathbf{x}}_1\hat{\mathbf{y}}_1\hat{\mathbf{z}}_1$-frame in terms of local coordinates (n_{1o}, m_{1o} and l_{1o}). Also, during the positioning process the vector $\mathbf{pi}_{3o}$ will perform a $\theta_3\hat{\mathbf{z}}_3$-type rotation to evolve into $\mathbf{pi}_{3r}$ which can be expressed with respect to the $\hat{\mathbf{x}}_3\hat{\mathbf{y}}_3\hat{\mathbf{z}}_3$-frame in terms of local coordinates (n_{3r}, m_{3r} and l_{3r}). Therefore the two vectors, $\mathbf{pf}_{1o}$ and $\mathbf{pi}_{3r}$ can be written as follows;

$$\left. \begin{array}{l} \mathbf{pf}_{1o} = n_{1o}\hat{\mathbf{x}}_1 + m_{1o}\hat{\mathbf{y}}_1 + l_{1o}\hat{\mathbf{z}}_1 \\ \mathbf{pi}_{3r} = n_{3r}\hat{\mathbf{x}}_3 + m_{3r}\hat{\mathbf{y}}_3 + l_{3r}\hat{\mathbf{z}}_3 \end{array} \right\} \tag{8}$$

where the two equations above have four unknowns that need to be determined. These four unknowns are n_{1o}, m_{1o}, n_{3r} and m_{3r}. The numerical values of the l-type local coordinates are calculated as follows;

$$
\left.\begin{aligned}
l_{3r} &= \mathbf{pi}_{3o} \circ \hat{\mathbf{z}}_3 \\
l_{1o} &= \mathbf{pf}_{1r} \circ \hat{\mathbf{z}}_1
\end{aligned}\right\}
\tag{9}
$$

In fact the value of l_{3r} is calculated, and stored in a data file, once the manipulator has been calibrated and an initial position has been nominated; however, l_{1o} has to be calculated for every new desired end-effector position. Moreover, the end-effector positions which are defined by the vectors $\mathbf{pf}_{1o}$ and $\mathbf{pi}_{3r}$ can be used to study the rotation about the middle joint-axis, $\hat{\mathbf{z}}_2$. These same positions can be expressed relative to a point, $\mathbf{p}_2$, attached to $\hat{\mathbf{z}}_2$, using the two respective vectors, $\mathbf{p}_{2r}$ and $\mathbf{p}_{20}$ as follows;

$$
\left.\begin{aligned}
\mathbf{p}_{2r} &= \mathbf{pf}_{1o} + d_2\hat{\mathbf{z}}_2 - a_2\hat{\mathbf{x}}_2 \\
\mathbf{p}_{2o} &= \mathbf{pi}_{3r} + a_3\hat{\mathbf{x}}_3
\end{aligned}\right\}
\tag{10}
$$

where $d_2 = (\mathbf{p}_{21} - \mathbf{p}_2) \circ \hat{\mathbf{z}}_2$

It may be noted that $\mathbf{p}_{2r}$ and $\mathbf{p}_{20}$ are separated by a single rotation, $\theta_2\hat{\mathbf{z}}_2$. The properties of this rotation may be utilised to show that,

$$
\mathbf{pi}_{2o} \circ \hat{\mathbf{z}}_2 = \mathbf{pi}_{2r} \circ \hat{\mathbf{z}}_2
\tag{11}
$$
and
$$
\mathbf{pi}_{2o} \circ \mathbf{pi}_{2o} = \mathbf{pi}_{2r} \circ \mathbf{pi}_{2r}
\tag{12}
$$

Equations (7) to (12) may then be manipulated to obtain the following two linear equations,
$$
m_{1o}\hat{\mathbf{y}}_1 \circ \hat{\mathbf{z}}_2 = m_{3r}\hat{\mathbf{y}}_3 \circ \hat{\mathbf{z}}_2 + l_{3r}\hat{\mathbf{z}}_3 \circ \hat{\mathbf{z}}_2 - l_{1o}\hat{\mathbf{z}}_1 \circ \hat{\mathbf{z}}_2 - d_2
\tag{13}
$$

and

$$
m_{1o}d_2\hat{\mathbf{y}}_1 \circ \hat{\mathbf{z}}_2 - a_2 n_{1o} = a_3 n_{3r} + \frac{1}{2}\left(\mathbf{pi}_{3i} \circ \mathbf{pi}_{3i} - \mathbf{pf}_{1r} \circ \mathbf{pf}_{1r} + a_3^2 - a_2^2 - d_2^2 - 2l_{1o}d_2\hat{\mathbf{z}}_1 \circ \hat{\mathbf{z}}_2\right)
\tag{14}
$$

The concept in equations (3) and (5) may be employed to express the x_2-, y_2- and z_2-components of a rotated vector $\mathbf{p}_{2o}^r$ which results from performing a $\theta_2\hat{\mathbf{z}}_2$ rotation on $\mathbf{p}_{20}$. Then the coincidence of $\mathbf{p}_{2o}^r$ and $\mathbf{p}_{2r}$ may be described by,

$$\mathbf{p}_{2_o}^r \circ \hat{\mathbf{x}}_2 = \mathbf{p}_{2_r} \circ \hat{\mathbf{x}}_2 \tag{15}$$

and

$$\mathbf{p}_{2_o}^r \circ \hat{\mathbf{y}}_2 = \mathbf{p}_{2_r} \circ \hat{\mathbf{y}}_2 \tag{16}$$

where $\mathbf{p}_{2_o}^r \circ \hat{\mathbf{z}}_2 = \mathbf{p}_{2_r} \circ \hat{\mathbf{z}}_2$ is already described in equation (13) , and the expanded forms of the (15) and (16) are given respectively as follows;

$$
\begin{aligned}
n_{1_o} - a_2 &= \left(\hat{\mathbf{x}}_3 \circ \hat{\mathbf{x}}_2 c_2 - \hat{\mathbf{x}}_3 \circ \hat{\mathbf{y}}_2 s_2 \right) n_{3r} + \left(\hat{\mathbf{y}}_3 \circ \hat{\mathbf{x}}_2 c_2 - \hat{\mathbf{y}}_3 \circ \hat{\mathbf{y}}_2 s_2 \right) m_{3r} \\
&\quad + \left(l_{3r} \hat{\mathbf{z}}_3 \circ \hat{\mathbf{x}}_2 + a_3 \hat{\mathbf{x}}_3 \circ \hat{\mathbf{x}}_2 \right) c_2 - \left(l_{3r} \hat{\mathbf{z}}_3 \circ \hat{\mathbf{y}}_2 + a_3 \hat{\mathbf{x}}_3 \circ \hat{\mathbf{y}}_2 \right) s_2
\end{aligned}
\tag{17}
$$

and

$$
\begin{aligned}
m_{1_o} \hat{\mathbf{y}}_1 \circ \hat{\mathbf{y}}_2 + l_{1_o} \hat{\mathbf{z}}_1 \circ \hat{\mathbf{y}}_2 &= \left(\hat{\mathbf{x}}_3 \circ \hat{\mathbf{x}}_2 s_2 + \hat{\mathbf{x}}_3 \circ \hat{\mathbf{y}}_2 c_2 \right) n_{3r} + \left(\hat{\mathbf{y}}_3 \circ \hat{\mathbf{x}}_2 s_2 + \hat{\mathbf{y}}_3 \circ \hat{\mathbf{y}}_2 c_2 \right) m_{3r} \\
&\quad + \left(l_{3r} \hat{\mathbf{z}}_3 \circ \hat{\mathbf{y}}_2 + a_3 \hat{\mathbf{x}}_3 \circ \hat{\mathbf{y}}_2 \right) c_2 + \left(l_{3r} \hat{\mathbf{z}}_3 \circ \hat{\mathbf{x}}_2 + a_3 \hat{\mathbf{x}}_3 \circ \hat{\mathbf{x}}_2 \right) s_2
\end{aligned}
\tag{18}
$$

where c_2 and s_2 stand for $\cos\theta_2$ and $\sin\theta_2$ respectively.

The four linear equations, (13), (14), (17) and (18) represent the mathematical core of the kinematic model introduced in the present work for the inverse position analysis of the arm module. A symbolic solution for these equations can be obtained such that, n_{3r} and m_{3r} are expressed in the following forms,

$$n_{3r} = f_1 / f \tag{19}$$

and

$$m_{3r} = f_2 / f \tag{20}$$

where f, f_1 and f_2 are linear functions of s_2 and c_2.
Noting the properties of rotation about $\hat{\mathbf{z}}_3$ the following may be deduced,

$$f_1^2 + f_2^2 = f^2 \left(\mathbf{pi}_{3_o} \circ \mathbf{pi}_{3_o} - l_{3r}^2 \right) \tag{21}$$

This last equation is a polynomial of s_2, c_2, s_2^2, c_2^2 and $s_2 c_2$; and can be re-expressed in the following form,

$$\sum_{k=0}^{4} b_{4-k} t^{4-k} = 0 \tag{22}$$

where the coefficients are calculated symbolically from the model presented above. The parameters which constitute these coefficients reflect the kinematic relations between the successive arm axes and they can be calculated and stored for run-time use once the arm is calibrated. These parameters are all constant for a given arm except $\mathbf{pf}_{1r} \circ \mathbf{pf}_{1r}$ and l_{1o} which depend on the desired final location of the end-effector as described above. The fact that only two parameters need to be calculated highlights the computational efficiency of the described approach.

In (22), t is the tangent of half θ_2 and it is used to replace c_2 and s_2 as follows,

$$
\left.
\begin{aligned}
c_2 &= \frac{1-t^2}{1+t^2} \\[2mm]
s_2 &= \frac{2t}{1+t^2}
\end{aligned}
\right\}
\tag{23}
$$

Equation (22), which is a fourth degree polynomial, can be solved using a systematic non-iterative technique (Tranter, 1980). The resulting roots can successively be plugged back in equations (23) to work out the corresponding values of c_2 and s_2. These values are then employed to obtain the joint-displacement θ_2 using the atan2 function referred to above. The values of the local coordinates, n_{3r} and m_{3r}, may be calculated by using equations (19) and (20).

A numerically stable method to obtain m_{1o} and n_{1o} is to use equation (17) for n_{1o} and then obtain m_{1o} from the following equation,

$$
m_{1o} = \left(\mathbf{p}_{2o}^r - d_2 \hat{\mathbf{z}}_2 \right) \circ \hat{\mathbf{y}}_1
\tag{24}
$$

Finally, n_{3r}, m_{3r}, n_{3o}, m_{3o} are employed in equation (6) to obtain the corresponding values of θ_3. Similarly, n_{1r}, m_{1r}, n_{1o}, m_{1o} are used to obtain the corresponding values of θ_1.

As revealed by the polynomial in (22), the maximum number of arm configurations, N_{arm}, which correspond to a given end-effector position is four. In some cases, however, the geometrical relationships between the consecutive axes as well as the required position of p_f allow for the inverse position problem to be solved through the use of quadratic, rather quartic or higher, polynomials. Arms which exhibit this sort of simplification are said to have simple structures. Some of these cases are outlined in the next section.

5. Kinematic Synthesis of the Arm Mechanism

Most industrial robots are designed to have their successive axes either parallel or perpendicular to make a simplified closed form inverse position solution achievable. Researchers have repeatedly assigned the term, *simple structure*, to these robotic arms. The word "*simple*" usually implies that a non-iterative solution can be obtained for the inverse position problem of this particular structure. However, as the discussion in the previous section reveals, a non-iterative solution can still be obtained even for arms with arbitrarily positioned and directed joint-axes. A definition has been proposed for this term in the light of the conics theory (Smith and Lipkin, 1990). In the present section, a consistent simplified geometrical definition is introduced.

To gain understanding of the results obtainable from the fourth-degree polynomial equation (22), equations (13) and (14) along with the following two equations may be considered,

$$n_{3r}^2 + m_{3r}^2 = \mathbf{pi}_{3o} \circ \mathbf{pi}_{3o} - l_{3r}^2 \tag{25}$$

$$n_{1o}^2 + m_{1o}^2 = \mathbf{pf}_{1r} \circ \mathbf{pf}_{1r} - l_{1o}^2 \tag{26}$$

The four equations, (13), (14), (25) and (26), together are useful in studying the kinematic behaviour of the arm mechanism.

Essentially, the inverse position problem of the arm structure may be depicted as shown in Figure (3). In the figure, points p_f and p_i assume their local circular paths about the rotary axes, $\hat{\mathbf{z}}_1$ and $\hat{\mathbf{z}}_3$, creating two spatial circles Cz_1 and Cz_3, respectively, in two planes perpendicular to $\hat{\mathbf{z}}_1$ and $\hat{\mathbf{z}}_3$ with their centres located on the axes. Thus, a solution exists if a circle, Cz_2, that intersects both Cz_1 and Cz_3 simultaneously, can be drawn in a plane normal to $\hat{\mathbf{z}}_2$ with its centre located along it. As the analysis given in the previous section suggests, if the three axes are located and directed arbitrarily in space, a maximum of four different circles can be drawn about $\hat{\mathbf{z}}_2$ to satisfy this condition. Each circle intersects each of Cz_1 and Cz_3 at **one** point and hence, the four possible solutions.
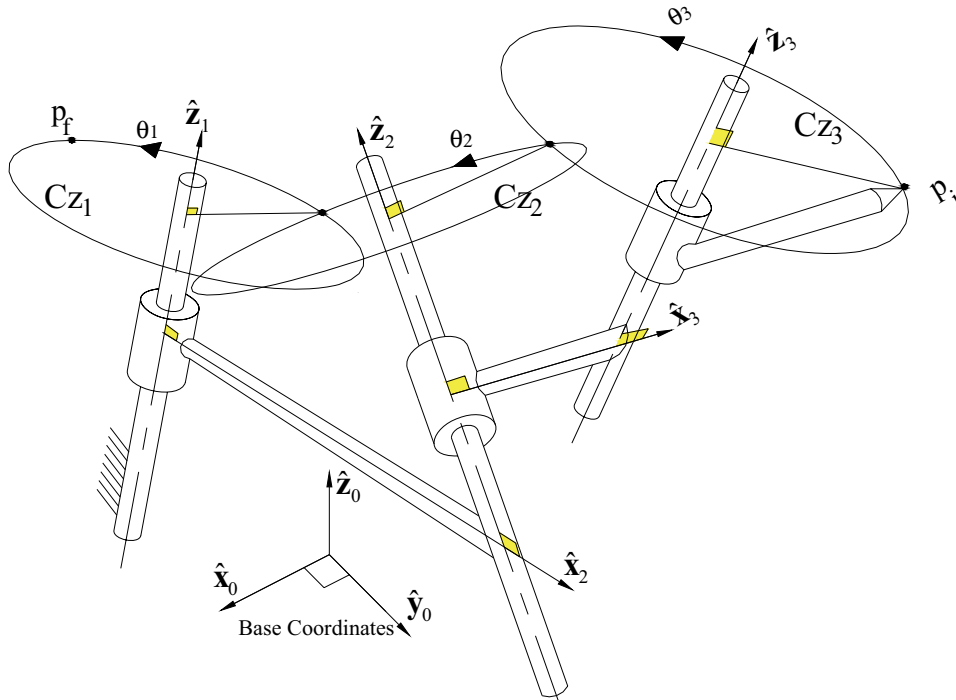
Figure 3. The Kinematic Behaviour of the Arm Joints.

As established in Appendix B, any two spatial circles may intersect at two points if, and only if, their corresponding axes lie in one and the same plane (this is the plane which perpendicularly halves the line connecting the two points of intersection). Therefore, in arms with ideal (non-calibrated) structures where $\hat{\mathbf{z}}_2$ lies in the same plane with either $\hat{\mathbf{z}}_1$ or $\hat{\mathbf{z}}_3$, the number of middle circles, Cz_2, becomes **two**. In such a case, the complex mathematical aspects associated with the inverse position problem of the arm disappear and the solution can easily be obtained by using equations (13), (14), (25) and (26). For example, if $\hat{\mathbf{z}}_1$ and $\hat{\mathbf{z}}_2$ lie in one plane where $a_2 = 0$, the following procedure may be adopted for the solution;

- i. use equations (13) and (14) to express n_{3r} and m_{3r} as functions of m_{1o}.

- ii. use these functions in equation (25) to obtain the two roots of m_{1o}: $m_{1o}|_1$ and $m_{1o}|_2$,

- iii. use equation (26) to obtain the four corresponding values of n_{1o}: $n_{1o}|_1$, $-n_{1o}|_1$, $n_{1o}|_2$ and $-n_{1o}|_2$,

- iv. use equations (13) to obtain the two corresponding values of m_{3r}: $m_{3r}|_1$ and $m_{3r}|_2$; then use (14) to obtain the corresponding values of n_{3r}:. $n_{3r}|_1$ and $n_{3r}|_2$,

- v. combine the roots in the following order to obtain the required solutions,

- $(n_{1o}|_1, \ m_{1o}|_1, \ n_{3r}|_1, \ m_{3r}|_1)$, $(-n_{1o}|_1, \ m_{1o}|_1, \ n_{3r}|_1, \ m_{3r}|_1)$, $(n_{1o}|_2, \ m_{1o}|_2, \ n_{3r}|_2, \ m_{3r}|_2)$ and $(-n_{1o}|_2, \ m_{1o}|_2, \ n_{3r}|_2, \ m_{3r}|_2)$.

The corresponding four values of θ_2 may be obtained by solving equations (17) and (18) simultaneously for $\cos\theta_2$ and $\sin\theta_2$. Also, θ_1 and θ_3 are obtained by using equation (5).

The above mathematical procedure can be performed symbolically such that, closed form expressions are obtained for the three joint-displacements.

Similar simplified mathematical procedure may be used in cases with $\hat{\mathbf{z}}_1$ parallel to $\hat{\mathbf{z}}_2$. It may be noted that in designs where $\hat{\mathbf{z}}_2$ lies in one plane with $\hat{\mathbf{z}}_1$ and in another plane with $\hat{\mathbf{z}}_3$, the number of middle circles, Cz2, becomes **one** and the solution can be simplified even further. In such a case, the middle circle, Cz2, intersects both Cz1 and Cz3 at two points to produce the four possible solutions. An example may be sought in PUMA-type robots, whose nominal structures possess the following kinematic features,

$$a_2 = 0, \qquad\qquad \hat{\mathbf{y}}_1 \circ \hat{\mathbf{z}}_2 = -1 \qquad \text{and} \qquad \hat{\mathbf{y}}_3 \circ \hat{\mathbf{z}}_2 = 0$$

This makes it possible to obtain the solution for a non-calibrated PUMA arm substructure using the following procedure,

- i. obtain m_{1o} from equation (13)
- ii. obtain n_{3r} from equation (14)
- iii. obtain $\pm m_{3r}$ from equation (25)
- iv. obtain $\pm n_{1o}$ from equation (26)

Thus, the four possible configurations of the arm are given by the following root combinations,

$(n_{1o}, m_{1o}, n_{3r}, m_{3r})$, $(n_{1o}, m_{1o}, n_{3r}, -m_{3r})$, $(-n_{1o}, m_{1o}, n_{3r}, m_{3r})$ and $(-n_{1o}, m_{1o}, n_{3r}, -m_{3r})$.

Based on the above discussion it can be concluded that the middle axes $\hat{\mathbf{z}}_2$ must lie in one plane with either $\hat{\mathbf{z}}_1$ or $\hat{\mathbf{z}}_3$ for a simplified mathematical procedure to be realisable. Once this condition is satisfied, the four equations, (13), (14), (25) and (26) can be readily employed to obtain the inverse solution and therefore the arm structure can be described as *simple*.

In the next section, the procedure which is presented for the inverse position analysis of the wrist substructure is explained.

6. Inverse Kinematics of the Wrist

In the current context, the main task of the first two wrist joints (namely the fourth and fifth joints on the manipulator structure) is to displace the axis of the last joint (i.e. the sixth joint) from a given known orientation to a new desired direction in space.

Figure (4) depicts an arrangement of two revolute joints with their axes $\hat{\mathbf{z}}_4$ and $\hat{\mathbf{z}}_5$ pointing in the directions calculated using any suitable direct kinematic procedure featuring three consecutive rotations, $\theta_3\hat{\mathbf{z}}_3$, $\theta_2\hat{\mathbf{z}}_2$ and $\theta_1\hat{\mathbf{z}}_1$. At this specific pose, the axis of the sixth joint, $\hat{\mathbf{z}}_6^i\big|_{5o}$, is also calculated using the same consecutive rotations, and it is now required to be orientated in the direction of $\hat{\mathbf{z}}_6^f$. In the figure, the common normal $\hat{\mathbf{x}}_5$ is directed from $\hat{\mathbf{z}}_4$ to $\hat{\mathbf{z}}_5$ (where $\hat{\mathbf{x}}_5 = \hat{\mathbf{z}}_4 \times \hat{\mathbf{z}}_5$). At zero position $\hat{\mathbf{x}}_4$ is selected to coincide with $\hat{\mathbf{x}}_5$ such that two Cartesian coordinate systems $\hat{\mathbf{x}}_4\hat{\mathbf{y}}_4\hat{\mathbf{z}}_4$ and $\hat{\mathbf{x}}_5\hat{\mathbf{y}}_5\hat{\mathbf{z}}_5$ can be established. According to the concepts in (4) and (5), $\hat{\mathbf{z}}_6^f$ can be described with respect to the $\hat{\mathbf{x}}_4\hat{\mathbf{y}}_4\hat{\mathbf{z}}_4$-frame in terms of local coordinates (n_{4r}, m_{4r} and l_{4r}). Also, $\hat{\mathbf{z}}_6^i\big|_{5o}$ can be described with respect to the $\hat{\mathbf{x}}_5\hat{\mathbf{y}}_5\hat{\mathbf{z}}_5$-frame in terms of known local coordinates (n_{5o}, m_{5o} and l_{5o}).
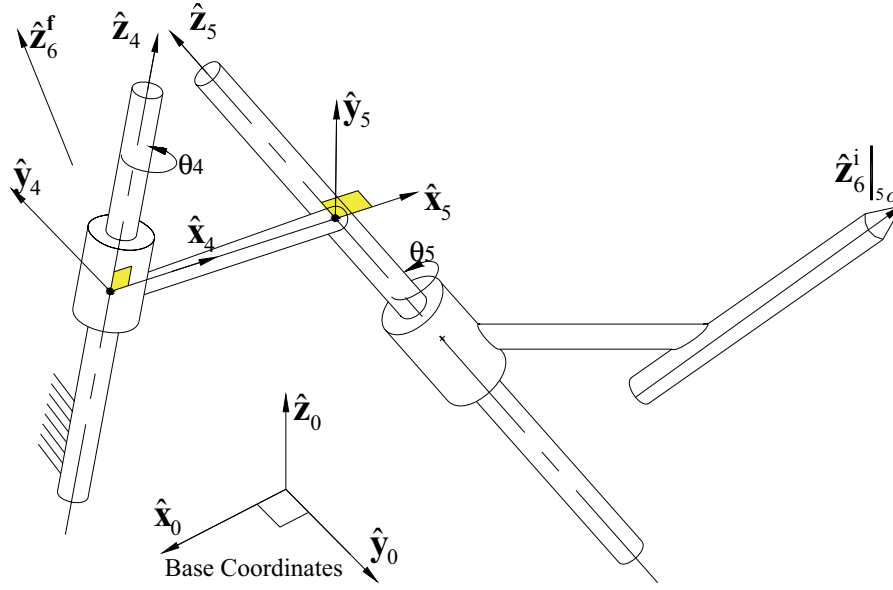
Figure 4. A 2R Arrangement Used for Orienting Vectors in Space.

It is understood that the vector $\hat{\mathbf{z}}_6^{\mathrm{f}}$ resulted from rotating another vector $\hat{\mathbf{z}}_6^{\mathrm{f}}\big|_{4o}$ about the $\hat{\mathbf{z}}_4$ axis by an angle, θ_4 (i.e. a $\theta_4\hat{\mathbf{z}}_4$-type rotation). The original vector, $\hat{\mathbf{z}}_6^{\mathrm{f}}\big|_{4o}$, can be expressed with respect to the $\hat{\mathbf{x}}_4\hat{\mathbf{y}}_4\hat{\mathbf{z}}_4$-frame in terms of local coordinates (n_{4o}, m_{4o} and l_{4o}), where n_{4o} and m_{4o} are unknowns to be worked out and l_{4o} is numerically obtained from $l_{4o} = l_{4r} = \hat{\mathbf{z}}_6^{\mathrm{f}} \circ \hat{\mathbf{z}}_4$.

The vector $\hat{\mathbf{z}}_6^{\mathrm{i}}\big|_{5r}$ which results from rotating $\hat{\mathbf{z}}_6^{\mathrm{i}}\big|_{5o}$ by an angle $\theta_5\hat{\mathbf{z}}_5$ may be expressed in the following form,

$$\hat{\mathbf{z}}_6^{\mathrm{i}}\big|_{5r} = \left(n_{5o}c_5 - m_{5o}s_5\right)\hat{\mathbf{x}}_5 + \left(n_{5o}s_5 + m_{5o}c_5\right)\hat{\mathbf{y}}_5 + l_{5o}\hat{\mathbf{z}}_5 \tag{27}$$

where c_5 and s_5 stand for $\cos\theta_5$ and $\sin\theta_5$ respectively.

A property of rotation about $\hat{\mathbf{z}}_4$ may be stated as,

$$l_{4o} = \left(n_{5o}s_5 + m_{5o}c_5\right)\hat{\mathbf{y}}_5 \circ \hat{\mathbf{z}}_4 + l_{5o}\hat{\mathbf{z}}_5 \circ \hat{\mathbf{z}}_4 \tag{28}$$

This last expression (28) is a linear equation in s_5 and c_5. This equation may be re-expressed in a polynomial form as follows,

$$\sum_{k=0}^{2} b_{2-k} t^{2-k} = 0 \tag{29}$$

where t is the tangent of half θ_5 and b_j is the coefficient of the j$^{\text{th}}$ power term.

It could be concluded from equation (29), which is a second degree polynomial, that the number of the wrist configurations, N_{wrist}, which correspond to the required orientation of $\hat{\mathbf{z}}_6^{\text{f}}$ is ≤ 2.

Once θ_5 is obtained, m_{4o} and n_{4o} can be worked out as follows;

$$
\left.\begin{aligned}
n_{4o} &= n_{5o}c_5 - m_{5o}s_5 \\
m_{4o} &= \left(n_{5o}s_5 + m_{5o}c_5\right)\hat{\mathbf{y}}_5 \circ \hat{\mathbf{y}}_4 + l_{5o}\hat{\mathbf{z}}_5 \circ \hat{\mathbf{y}}_4
\end{aligned}\right\} \tag{30}
$$

Finally, n_{4r}, m_{4r}, n_{4o} and m_{4o} are employed in equation (6) to obtain the corresponding values of θ_4.

From the analysis presented in this and the previous sections, it can be concluded that the maximum number of configurations of a spherical-wrist manipulator structure which correspond to any given position and orientation at the end-effector is eight. The actual number of configurations, N, is calculated by,

$$
N = N_{arm}N_{wrist} \tag{31}
$$

In spherical-wrist manipulators, each arm configuration corresponds to two possible wrist configurations as indicated by equation (31).

7. Completing the Full Pose

Once the first five joints on the manipulator structure have performed consecutive rotations ($\theta_i\hat{\mathbf{z}}_i$, where $i = 1,2,...5$) to place the sixth joint axis at its desired position and orientation, one final rotation ($\theta_6\hat{\mathbf{z}}_6$), will be performed to align any side axis on the end-effector with its desired direction. The term "side axis" here refers to any axis, on the end-effector Cartesian frame, whose direction is influenced by rotations performed about $\hat{\mathbf{z}}_6$. This final part of the inverse kinematic procedure is a straight forward application of the model presented in equations (3) to (6) to calculate the angle of rotation. However, it worth noting here that this final step of the analysis is preceded by a direct kinematic procedure to calculate the updated direction of the side axis after five consecutive rotations, $\theta_5\hat{\mathbf{z}}_5, \theta_4\hat{\mathbf{z}}_4, \theta_3\hat{\mathbf{z}}_3, \theta_2\hat{\mathbf{z}}_2$ and $\theta_1\hat{\mathbf{z}}_1$, have been performed.

8. The Inverse Solution Procedure

Figure (5) depicts a flow chart that has been designed to explain the procedure proposed here for the inverse position analysis of manipulators. For spherical-wrist manipulators, the procedure produces eight sets of solutions in a non-iterative fashion. However, for calibrated robotic structures, the eight solutions are obtained in a simple iterative approach which does not involve any Jacobian matrix computations. By virtue of the concepts presented, the various solutions may be calculated simultaneously if parallel computing facilities are available.

In the present approach, the arm is assigned the task of positioning any point on the sixth joint-axis at its required spatial location. The closest point on the sixth-joint axis to the fifth joint-axis may be conveniently selected for this purpose. This point will be referred to in the following discussion as p_0^i. The four joint-displacement solutions which correspond to this positioning task are therefore obtained using the models presented above and saved in four three-element vectors, $\mathbf{v}_j$, where j=1,2,3 and 4.

At arm configuration number j, the wrist joints align the sixth joint-axis with its required final orientation, as previously described, and the two corresponding solutions are accordingly obtained and saved in a pair of two-elements vectors, $\mathbf{w}_{jk}$, where k may assume the values of 1 or 2. To this end, a set of eight joint-displacement solutions have been obtained. If the robot was of the spherical-wrist type these solutions should accurately represent the required joint-displacements and no iterations would be required.

Calibrated robots, however, are not likely to have their last three joint-axes intersecting at a common point (i.e. the spherical-wrist property is lost), the motions performed by the wrist joints will displace the point which was previously positioned by the arm to eight new locations, p_0^{jk}, corresponding to the wrist solutions obtained.

At location number jk, the instantaneous position vector, $\mathbf{p}_0^{jk}$, of the displaced point may be calculated, using a suitable direct kinematic procedure, and compared to the required position vector $\mathbf{p}_0^n$ where the net radial error, e_{jk}, is calculated as follows,

$$e_{jk} = \left\| \mathbf{p}_0^n - \mathbf{p}_0^{jk} \right\| \tag{32}$$
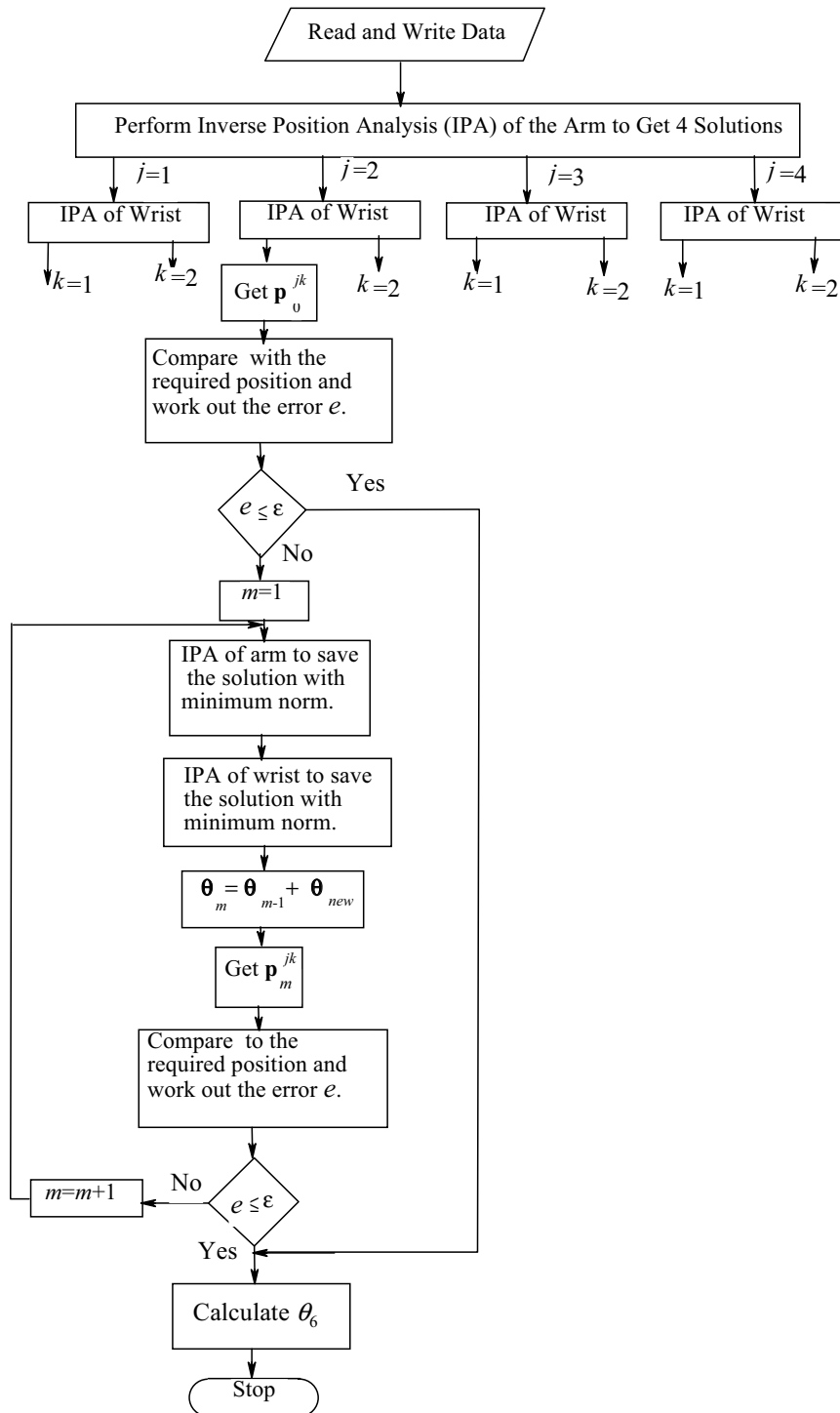
Figure 5. Inverse Position Analysis of Robots Using Elementary Motions.

If the calculated value for e_{jk} does not fall within an allowable error zone, the calculations proceed such that at iteration number m, the arm sets out from the most updated configuration number $jk(m$-$1)$ to position point p_{m-1}^{jk} in the required location, $\mathbf{p}_0^n$. The four solutions obtained may be stored in four three-

element vectors whose norms are subsequently calculated and compared. Only the vector which corresponds to minimum norm, $\mathbf{v}_{jk}^{m}$, may be saved in the memory and the other solutions would be discarded. This vector is referred to here as the **arm elementary-motions vector** because it contains fractional quantities of elementary joint-displacements.

The two corresponding wrist solutions may then be obtained and stored in a pair of two-element vectors whose norms will also be calculated and compared. The vector with minimum norm, $\mathbf{w}_{jk}^{m}$, is subsequently saved while the other vector may be disposed of. In the current context, $\mathbf{w}_{jk}^{m}$ is designated as the **wrist elementary-motions vector** because it contains small values of joint-displacements.

The new displaced location of the positioned point may then be calculated and compared with the required location as per equation (32). When the radial error is small enough, the final joint-displacement vector, $\mathbf{v}_{jk}^{n}$, of the arm group which corresponds to solution number jk may be calculated as follows,

$$\mathbf{v}_{jk}^{n} = \mathbf{v}_{j} + \sum_{m=1}^{M} \mathbf{v}_{jk}^{m} \qquad (33)$$

where M is the corresponding number of iterations.

The vector, $\mathbf{w}_{jk}^{n}$, which corresponds to the jk-solution of the wrist is calculated as;

$$\mathbf{w}_{jk}^{n} = \mathbf{w}_{jk} + \sum_{m=1}^{M} \mathbf{w}_{jk}^{m} \qquad (34)$$

Once the jk-solution for the first five joint-displacements has been obtained, the corresponding displacement of the last joint may simply be calculated.

The iterative technique presented here utilises the physical kinematic behaviour of manipulator joints and therefore fast and singularity-proof convergence may be assured. The technique does not require initial guesses introduced into the model.

In the next section a numerical example is given where the inverse position solutions will be produced for a PUMA-type robot of both calibrated and ideal structures.

9. Numerical Example

A PUMA-like manipulator with six revolute joints is selected for the example. The dimensions of the spherical-wrist structure of the manipulator are given in Table (1). The dimensions of the non-spherical-wrist version of the same manipulator are similar to those given in Table (1) except for the locations of the fourth and fifth joint-axes which were displaced to (-128.0, 818.51 and 205.04 mm) and (-130.5, 802.0 and 180.4 mm) respectively.

Axes	Direction Cosines of Joint-axes			Axes Locations (mm)		
	z_x	z_y	z_z	p_x	p_y	p_z
z_1	-0.0871557	0.02255767	0.9848077	-1.0	-9.0	8.0
z_2	-0.9961946	0.0001274	-0.0871557	5.0	-5.0	198.0
z_3	-0.9961947	0.05233595	0.0696266	-68.0	438.0	195.0
z_4	0.02233595	-0.9993908	0.02681566	-130.5	808.5	177.0
z_5	0.99975050	-0.0223359	0.00009744	-130.5	808.5	177.0
z_6	0.02489949	0.9996253	0.00012081	-130.0	808.5	177.0

Table 1. Cartesian Dimensions of a Spherical-wrist Manipulator.

In both cases, the initial and final locations of the Tool Centre Point (TCP) of the end-effector are given with respect to the base coordinates as, -120.54, 1208.36 and 175.095 and, –400.0, –400.0 and 1009.0 mm respectively. The initial and final orientations of the end-effector are given in terms of an Euler ZYZ-system as: 88.5733, 89.9604 and 89.722 and, 120.0, –20.0 and 150.0 degrees respectively.

The models proposed in this paper were used to calculate the inverse position solutions for both the spherical-wrist and general manipulator structures and the results are displayed in Tables (2) and (3) respectively. The angular displacements given in these tables are in degrees.

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Sol. No. 1	-34.45	-163.09	64.67	86.12	-36.06	-130.97
Sol. No. 2	-34.45	-163.09	64.67	-85.75	31.73	58.63
Sol. No. 3	-47.14	-104.39	-72.4	19.01	-84.81	154.74
Sol. No. 4	-47.14	-104.39	-72.4	-160.03	80.47	-20.04
Sol. No. 5	104.39	-72.90	64.57	24.26	85.22	2.38
Sol. No. 6	104.39	-72.90	64.57	-156.27	-89.56	177.24
Sol. No. 7	117.07	-11.05	-72.30	68.50	28.82	68.23
Sol. No. 8	117.07	-11.05	-72.30	-120.58	-33.16	-122.18

Table 2. Inverse Position Solutions for the Spherical-wrist Robot.

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Sol. No. 1	-39.20	-162.93	66.79	86.5	-36.82	-125.19
Sol. No. 2	-30.78	-162.92	70.95	-73.54	32.47	68.51
Sol. No. 3	-48.12	-106.90	-68.18	19.18	-83.65	156.37
Sol. No. 4	-46.46	-103.38	-65.35	-158.64	72.85	-17.25
Sol. No. 5	105.11	-73.14	67.49	24.37	87.28	0.42
Sol. No. 6	103.60	-70.98	72.08	-154.89	-98.19	173.67
Sol. No. 7	120.24	-11.00	-68.57	65.31	29.99	60.59
Sol. No. 8	114.00	-11.34	-62.96	-133.30	-37.94	-134.91

Table 3. Inverse Position Solutions for the Non-spherical-wrist Robot.

The solutions obtained for the spherical-wrist manipulator did not involve iterations at all. However, a maximum of 4 iterations were used for the non-spherical-wrist manipulator. In most cases the number of iterations was 2 except for the second and first solutions were this number was 3 and 4 respectively. This demonstrates the numerical efficiency of the proposed models.

10. Conclusions

The work presented in this paper introduces a technique for inverse position analysis of revolute-joint manipulators. The analysis developed results in simplified solutions for both the arm and the wrist subassemblies. These solutions are obtained in form of polynomials whose coefficients can be simply calculated for a given manipulator structure. The technique can be used to obtain inverse kinematic solutions for both spherical-wrist and calibrated manipulator structures.

The technique results in obtaining multiple sets of the joint-motor displacements which correspond to a given pose at the end-effector. This enables the trajectory designer to the select the joint-trajectory which best fits a desired manipulator task.

Appendix A

To relate a pair of successive axes on a manipulator structure, the direction co-
sines of the two axes are given (with respect to a Cartesian base frame), to-
gether with a position vector describing a point on each axis. These spatial
particulars are defined in Figure (A) as $\hat{\mathbf{z}}_i$ and $\mathbf{p}_{0i}$ for the joint axis number i,
and $\hat{\mathbf{z}}_{i+1}$ and $\mathbf{p}_{0(i+1)}$ for joint-axis number i+1. The procedure kicks off by cal-
culating the common normal, $\mathbf{x}_{i+1}$ as follows;

$$\mathbf{x}_{i+1} = \hat{\mathbf{z}}_i \times \hat{\mathbf{z}}_{i+1} \tag{A.1}$$

where the following condition is employed;

$$if \ \left\| \hat{\mathbf{z}}_i \times \hat{\mathbf{z}}_{i+1} \right\| = 0 \quad then \quad \mathbf{x}_{i+1} = \left(\mathbf{p}_{0(i+1)} - \mathbf{p}_{0i} \right) - \hat{\mathbf{z}}_i \left[\left(\mathbf{p}_{0(i+1)} - \mathbf{p}_{0i} \right) \circ \hat{\mathbf{z}}_i \right] \tag{A.2}$$

The unit vector, $\hat{\mathbf{x}}_{i+1}$, is then calculated as follows;

$$\hat{\mathbf{x}}_{i+1} = \frac{\mathbf{x}_{i+1}}{\left\| \mathbf{x}_{i+1} \right\|} \tag{A.3}$$
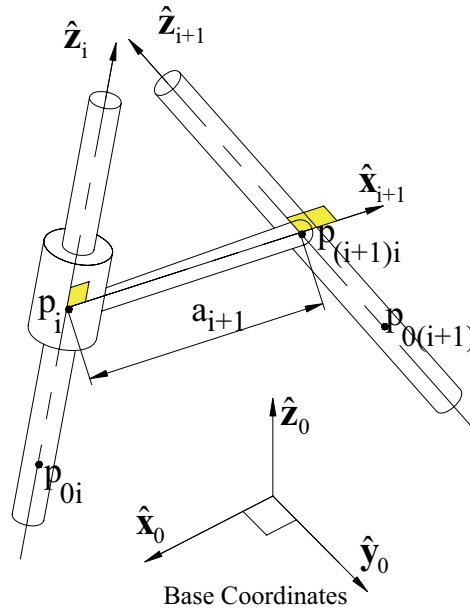
Figure A. Relating Successive Axes with a Common Normal.

The shortest distance, a_{i+1}, separating the two axes is calculated as follows;

$$a_{i+1} = \left(\mathbf{p}_{0(i+1)} - \mathbf{p}_{0i} \right) \circ \hat{\mathbf{x}}_{i+1} \tag{A.4}$$

The intersection of $\hat{\mathbf{x}}_{i+1}$ with $\hat{\mathbf{z}}_i$ is defined by a position vector $\mathbf{p}_i$, which is obtained from;

$$\mathbf{p}_i = \mathbf{p}_{0i} + b_i \hat{\mathbf{z}}_i \tag{A.5}$$

where

$$b_i = \frac{\left(\hat{\mathbf{z}}_{i+1} \times \hat{\mathbf{x}}_{i+1}\right) \circ \left(\mathbf{p}_{0(i+1)} - \mathbf{p}_{0i}\right)}{\left(\hat{\mathbf{z}}_{i+1} \times \hat{\mathbf{x}}_{i+1}\right) \circ \hat{\mathbf{z}}_i} \tag{A.6}$$

which is subject to the condition, $if \ \left(\hat{\mathbf{z}}_{i+1} \times \hat{\mathbf{x}}_{i+1}\right) \circ \hat{\mathbf{z}}_i = 0 \ \ then \ \ b_i = 0$ (A.7)

The intersection of $\hat{\mathbf{x}}_{i+1}$ with $\hat{\mathbf{z}}_{i+1}$ is defined by a position vector $\mathbf{p}_{i(i+1)}$, which is calculated from;

$$\mathbf{p}_{i(i+1)} = \mathbf{p}_i + a_{i+1} \hat{\mathbf{x}}_{i+1} \tag{A.8}$$

Appendix B

Claim:
Any two spatial circles intersect at two points if, and only if, their axes lie in one and the same plane.

Proof:
Figure (B) depicts two spatial circles, C_1 and C_2, and their axes, $\hat{\mathbf{z}}_1$ and $\hat{\mathbf{z}}_2$ respectively. The circles intersect one another at two points, s_1 and s_2. To prove that $\hat{\mathbf{z}}_1$ and $\hat{\mathbf{z}}_2$ must lie in one and the same plane, the centres of the two circles, Pc1 and Pc2, are connected to the point, s_3, which divides the line $\overline{s_1 s_2}$ into two equal parts.
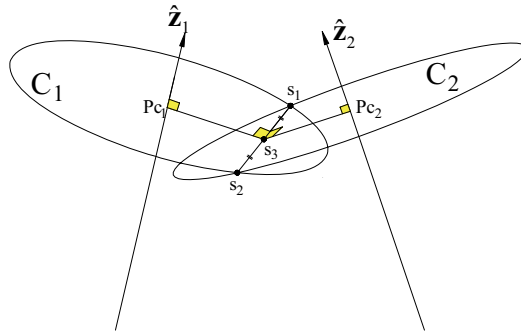
Figure B. Two-point Intersection of Spatial Circles.

- $\overline{s_1 s_2}$ lies in a plane perpendicular to $\hat{\mathbf{z}}_1$ and therefore $\overline{s_1 s_2}$ is perpendicular to $\hat{\mathbf{z}}_1$.

- From planar geometry, $\overline{s_1 s_2}$ is perpendicular to the line $\overline{P_{c_1} s_3}$.

- Therefore $\overline{s_1 s_2}$ is perpendicular to the plane which contains the two intersecting lines, $\hat{\mathbf{z}}_1$ and $\overline{P_{c_1} s_3}$. Let this plane be referred to as PN1.

- Similarly, it could be established that $\overline{s_1 s_2}$ is also perpendicular to the plane which contains the two intersecting lines, z_2 and $\overline{P_{c_2} s_3}$. This plane may be referred to as PN2.

- A general conclusion may now be drawn that, PN1 is parallel to PN2.

- However, PN1 and PN2 share one common point, s_3.

- Therefore, the two planes coincide and $\hat{\mathbf{z}}_1$, $\overline{P_{c_1} s_3}$, $\hat{\mathbf{z}}_2$ and $\overline{P_{c_2} s_3}$ must all lie in one and the same plane.

11. References

Chiaverini, S., Siciliano, B. and Egeland, O. (1994), Review of the Damped Least-Squares Inverse kinematics with Experiments on an Industrial Robot Manipulator, IEEE Trans. Control Syst. Technol., Vol. 2, No. 2, pp. 123-134.

Coelho, P. H. G. and Nunes, L. F. A. (1986), Application of Kalman Filtering to Robot Manipulators, In: *Recent Trends in Robotics: Modelling, Control and Education*. Jamshidi, M., Luh, L. Y. S. and Shahinpoor, M. (Ed.), pp. 35-40, Elsevier Science Publishing Co., Inc.

Denavit, J. and Hartenberg, R. S. (1955), A Kinematic Notation for Low Pair Mechanisms Based on Matrices, *J. Appl. Mech.-Trans. ASME*, Vol. 22, June 1955, pp. 215-221.

Duffy, J. and Crane C. (1980), A Displacement Analysis of the General Spatial 7-Link, 7R Mechanism, *Mech. Mach. Theory*, Vol. 15, No. 3-A, pp. 153-169.

Goldenberg, A. A., Benhabib, B. and Fenton, R. G. (1985), A Complete Generalised Solution to the Inverse Kinematics of Robots, *IEEE. Trans. Robot Autom*, Vol. RA-1, No. 1, pp. 14-20.

Gu, Y.-L. and Luh, J. Y. S. (1987), Dual-Number Transformation and Its Application to Robotics, *IEEE. Trans. Robot Autom*, Vol. RA-3, No. 6, pp. 615-623.

Gupta, K. (1984), A Note on Position Analysis of Manipulators, *Mech. Mach. Theory*, Vol. 19, No. 1, pp. 5-8.

Hayati, S. and Roston, G. (1986), Inverse Kinematic Solution for Near-Simple Robots and Its Application to Robot Calibration, In: *Recent Trends in Robotics: Modelling, Control and Education.*, Jamshidi, M., Luh, L. Y. S. and Shahinpoor, (Ed.), M. Elsevier Science Publishing Co., Inc, pp. 41-50.

Her, M.-G, Yen, C.-Y., Hung, Y.-C. and Karkoub, M. (2002), Approximating a Robot Inverse Kinematics Solution Using Fuzzy Logic Tuned by Genetic Algorithms, *Int. J. Adv. Manuf. Technol*, Vol. 20, pp. 372-380.

Kohli, D. and Osvatic, M. (1993), Inverse Kinematics of General 6R and 5R,P Spatial Manipulators, *ASME J. Mech. Des.*, Vol. 115, Dec. 1993, pp. 922-930.

Lee, H.-Y. and Liang, C. G. (1988a), A New Vector Theory for the Analysis of Spatial Mechanisms, *Mech. Mach. Theory*, Vol. 23, No. 13, pp. 209-217.

Lee, H.-Y. and Liang, C. G. (1988b), Displacement Analysis of the General Spatial 7-Link 7R Mechanism, *Mech. Mach. Theory*, Vol. 23, No. 13, pp. 219-226.

Lee, H.-Y., Reinholtz, C. F. (1996), Inverse Kinematics of Serial-Chain Manipulators, *J. Mech. Des.-Trans. ASME*, Vol. 118, Sept. 1996, pp. 396-404.

Mahalingam, S. and Sharan (1987), A., The Nonlinear Displacement Analysis of Robotic Manipulators Using the Complex Optimisation Method, *Mech. Mach. Theory*, Vol. 22, No. 1, pp. 89-95.

Manocha, D. and Canny, J. F. (1992), Real Time Inverse Kinematics for General 6R Manipulators, *Proceedings of IEEE Conf. on Robotics and Automation*, pp. 383-389, Nice-France, May 1992.

Manseur, R. and Doty, K. (1992a), A Complete Kinematic Analysis of Four-Revolute-Axis Robot Manipulators, *Mech. Mach. Theory*, Vol. 27, No. 5, pp. 575-586.

Manseur, R. and Doty, K. (1992b), Fast Inverse Kinematics of Five-Revolute-Axis Robot Manipulators, *Mech. Mach. Theory*, Vol. 27, No. 5, pp. 587-597.

Manseur, R. and Doty, K. (1989), A Robot Manipulator with 16 Real Inverse Kinematic Solution Sets, *Int. J. Robot Res*, Vol. 8, No. 5, pp. 75-79.

Manseur, R. and Doty, K. (1988), Fast Algorithm for Inverse Kinematic Analysis of Robot Manipulators, *Int. J. Robot Res*, Vol. 7, No. 3, pp. 52-63.

Manseur, R. and Doty, K. (1996), Structural Kinematics of 6-Revolute-Axis Robot Manipulators, *Mech. Mach. Theory*, Vol. 31, No. 5, pp. 647-657.

Pieper, D. L. and Roth, B. (1969), The Kinematics of Manipulators Under Computer Control, *Proceedings of 2nd Int. Congress on the Theory of Machines and Mechanisms*, Vol. 2, Zakopane, Poland, pp. 159–168.

Poon, J. K. and Lawrence, P. D. (1988), Manipulator Inverse Kinematics Based on Joint Functions, *Proceedings of IEEE Conf. on Robotics and Automation*, Philadelphia, pp. 669-674.

Pradeep, A. K., Yoder, P. J. and Mukundan, R. (1989), On the Use of Dual-Matrix Exponentials in Robotic Kinematic, *Int. J. Robot Res*, Vol. 8, No. 54, pp. 57-66.

Raghavan, M. and Roth, B. (1989), Kinematic Analysis of the 6R Manipulator of General Geometry, *Proceedings of the 5th Int. Symposium on Robotics Research*, Tokyo, pp. 263-269.

Smith, D. R. and Lipkin, H. (1990), Analysis of Fourth Order Manipulator Kinematics Using Conic Sections, *Proceedings of IEEE Conf. on Robotics and Automation*, Cincinnati, pp. 274-278.

Sultan, I. A. (2000), On the Positioning of Revolute-Joint Manipulators, *J. of Robot Syst*, Vol. 17, No. 8, pp 429-438.

Sultan, I. A. and Wager, J. G. (1999), User-Controlled Kinematic Modelling, *Adv. Robot.*, Vol. 12, No. 6, pp 663-677.

Sultan, I. A. and Wager, J. G. (2001), A ϕ-Model Solution for the Inverse Position Problem of Calibrated Robots Using Virtual Elementary Motions, *Inverse Probl. Eng.*, Vol. 9, No. 3, pp. 261-285.

Sultan, I. A. (2002), A Numerical Solution for Determinantal Polynomials with Application to Robot Kinematics, *Proceedings of the 4th Int. Conf. on Modelling and Simulation*, Melbourne.

Tsai, L.-W. and Morgan, A. P. (1985), Solving the Kinematics of the Most General Six- and Five-Degree-of-Freedom Manipulators by Continuation Methods, *ASME J. Mech, Transm and Autom Des*, Vol. 107, June 1985, pp. 189-199.

Tranter, C. J. (1980), *Techniques of Mathematical Analysis.* UNIBOOKS, Hodder and Stoughton, London.

Wang, K. and Bjorke, O. (1989), An Efficient Inverse Kinematic Solution with a Closed Form for Five-Degree-of-Freedom Robot Manipulators with a Non-Spherical Wrist, *Annals of CIRP*, Vol. 38, pp. 365-368.

Wang., L. T. and Chen, C. C. (1991), A Combined Optimisation Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators, *IEEE. Trans. Robot Autom*, Vol. 7, No. 4, pp. 489-499.

Yang, A. T. and Freudenstein, F. (1964), Application of Dual-Numbers Quaternion Algebra to the Analysis of Spatial Mechanisms, *J. Appl. Mech.-Trans. ASME*, June 1964, pp. 300-308.

Zhang, P.-Y., Lu, T.-S. and Song, L.- B. (2005), RBF networks-based inverse kinematics of 6R manipulator, *Int J Adv Manuf Technol.*, Vol. 26, pp. 144–147

Cable-based Robot Manipulators with Translational Degrees of Freedom

Saeed Behzadipour and Amir Khajepour

1. Introduction

Cable-based robots build upon mechanisms that not only use rigid links in their structures but also utilize unilateral force elements such as cables to deliver the desired motion. Cables may be either connected to active winches to provide a variable length and hence to actuate the mechanism or may be only to provide a kinematic constraint to eliminate an undesired motion of the end-effector. Manipulators in which the cables have variable lengths are usually called cable-driven or wire-driven manipulators.

Cable-based manipulators posses several advantages over conventional serial/parallel link manipulators including:

1. Large workspace: An active winch can provide a large range of length change on the cables at a low cost. This facilitates building manipulators for very large working spaces which cannot be obtained by other robots.

2. Low inertia: Materials provide their highest strength-to-mass ratio when they are under tensile loading. Using cables, which can be only in tension, maximizes the use of material strength and therefore reduces the mass and inertia of the manipulator. Low inertia is desirable in many applications including high speed/acceleration robotics.

3. Simplicity in structure: Cables simplify the robot structure by utilizing bending flexibility as kinematic joints and reducing the fabrication cost by minimizing the machining process.

4. Reconfigurability and transportability: Winch assemblies can be simply relocated to reconfigure and adjust the workspace of a cable-driven manipulator. The ease of assembly/disassembly of these manipulators also facilitates their transportation and quick setup.

5. Fully remote actuation: Using a fully cable-driven manipulator, all the actuators and sensitive parts are located away from the end-effector and the actual working area. Such manipulators best suit harsh or hazardous environments.

It should be also noted that using cable structures in robot manipulators is accompanied by theoretical and technical difficulties. The unilateral force of cables complicates the workspace, kinematics and dynamics analysis. The constraint of tensile force in all cables should be incorporated into the design and control procedure otherwise, the manipulator will collapse. Also, the low stiffness of the cables compared to rigid links may result in undesired vibrations requiring compensation by a proper control scheme.

As it was mentioned before, maintaining positive tension (tensile force) in all the cables is an essential requirement for the rigidity of a cable-based manipulator and hence, this property should be studied thoroughly before the cable-based manipulator can be used in any real application. In other words, a cable-based manipulator can be treated as a rigid link manipulator only if all the cables are in tension. As a result, most of the researchers' efforts on this category of robot manipulators have been spent on analyzing and proving the rigidity of the cable-based structures.

The general problem of rigidity in cable-based manipulators has been studied in the literature using different approaches and terminologies such as controllable workspace (Verhoeven & Hiller, 2000), dynamic workspace (Barette & Gosselin, 2005), wrench closure (Gouttefarde & Gosselin, 2006), manipulability (Gallina & Rosati 2002), fully constraint configuration (Roberts et al. 1998) and tensionability (Landsberger & Shanmugasundram, 1992). General formulation of this problem can be found in the works by (Ming & Higuchi 1994), (Tadokoro et al., 1996), and (Verhoeven et al., 1998). They showed that for the rigidity of a cable-based manipulator, it is necessary but not sufficient to have either actuation redundancy or separate external loading sources to produce tension in all cables. Ming (Ming & Higuchi 1994a,b) calls the first group Completely Restrained Positioning Mechanisms, CRPM, in which all the cables can be made taut with no external load while in an IRPM (Incompletely Restrained Positioning Mechanism), the manipulator cannot maintain its own rigidity and hence needs external load to make all cables taut.

The useful workspace of a cable-based manipulator is a subset of its geometrical workspace in which the manipulator can be rigidified (either by actuation redundancy or external loading). Determination of this workspace is the most essential step in the design and operation of a cable-based manipulator and is usually done by numerical search after the synthesis of the manipulator is done. Examples of this approach can be found in (Kawamura et al., 1995; Ferraresi, 2004; Ogahara, 2003; So-Ryeok et al., 2005a,b; Pusey et al., 2004). In this approach, if the workspace found through the search does not satisfy the design requirements, the synthesis of the manipulator and the workspace determination should be repeated. As a result and in order to avoid trial and error in the design, it is desired to have cable-based manipulators that can be rigidified everywhere in their geometrical workspace or at least their workspace can be analytically expressed. In this regard, a geometrical explanation for the

workspace of a cable crane has been found (Landsberger & Shanmugasundram, 1992) which is an IRPM. An analytical study for the boundaries of the workspace in planar cable-based manipulators is also performed in (Barette & Gosselin, 2005), (Gouttefarde & Gosselin, 2006) and (Stump & Kumar, 2006).

In this article, a series of cable-based manipulators with translational motion (Behzadipour, 2005) is studied with focus on their rigidity study. In these designs, cables are used to drive the end-effector as well as to eliminate its rotation by proper kinematic constraints. The significance of these new manipulators is that their rigidity can be guaranteed everywhere in their geometrical workspace by a certain set of conditions enforced on the geometry of the manipulator. This will be proved in details for each manipulator and the conditions will be derived. By incorporating these conditions into the design and control, the cables will be always taut and hence the cable-based manipulator can be treated as its rigid link counterpart.

In Section 2, the general structure of these manipulators will be presented and the critical concepts of rigidity and tensionability will be defined. In Section 3, a theorem is given to simplify the study of tensionability in these manipulators. In Sections 4 and 5, two spatial cable-based manipulators are introduced and their rigidity are proved. In Section 6, two planar manipulators with translational motion are presented and their rigidity are thoroughly studied.

2. General Structure and Definitions

The general configuration of the cable-based manipulators studied in this paper is shown in Fig. 1.

The four main elements of these manipulators are:

1. **Base**: The fixed part of the manipulator to which the global system of coordinate $OXYZ$ is attached

2. **End-effector**: The moving body which carries the moving frame $O'X'Y'Z'$.

3. **Cables**: The flexible tendon elements with negligible mass and diameter connected from one end to the end-effector at points P_i $(i=1,2,...,m)$ and pulled from the other end at Q_i. The pulling actuator produces tension τ_i inside the cable and can be simply a winch which pulls and winds the cable or a separate mechanism that moves the cable's end (Q_i) without changing its length. Unit vectors $\hat{\mathbf{u}}_i$ $(i=1,2,...,m)$ determine the direction of the cables and point towards the base. Depending on the structure of the manipulator, there may be some extra pulleys to guide the cables. The number of cables, m, is equal to the dimension of the motion space of the end-effector. Therefore, m is three and six for planar and spatial mechanisms, respectively.
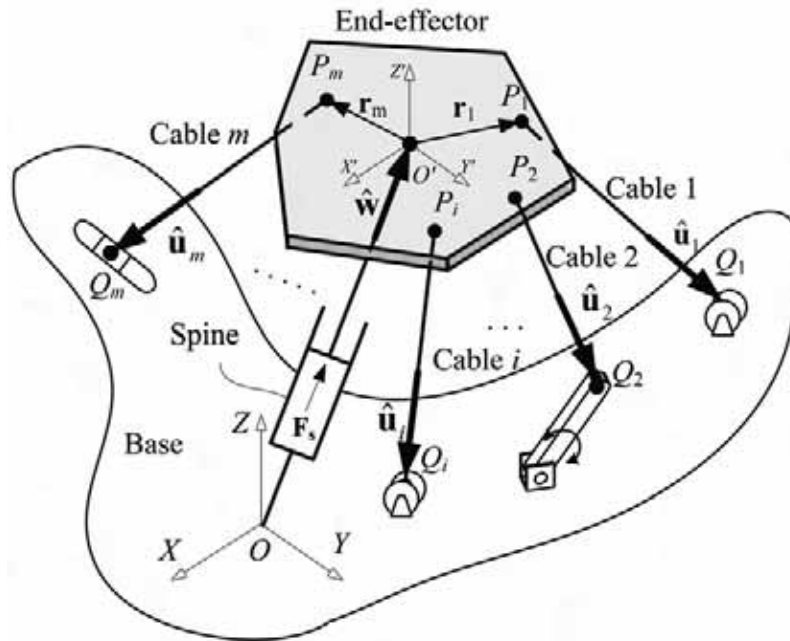
Figure 1. General configuration of the cable-based manipulators studied in this paper

4. **Spine**: The element that produces a force between the base and the end-effector in order to keep all the cables in tension. The spine can be an active element which generates a desired force. It can be also a passive element such as a cylinder energized by compressed air or a compressive spring designed properly to provide the sufficient force required to maintain tension in the cables. The direction of the spine is shown by unit vector $\hat{\mathbf{w}}$ pointing towards the end-effector.

For any cable-based manipulator, an equivalent rigid link counterpart can be found by replacing each cable by a rigid link and ball-and-socket joints at the ends. If the cable has a variable length, then a cylindrical element should be used to represent the cable in the rigid link manipulator. This analogy is valid as long as the cable-based manipulator is rigid according to the following definition:

Rigidity: A cable-based manipulator is rigid at a certain pose with respect to a given external load (including dynamic loads) and spine force if and only if all cables are in tension, $\tau_i \geq 0$ $i = 1, 2, ..., m$. A positive τ_i is considered as a tensile force in the cable.

It should be noted that the rigidity of a cable-based manipulator depends on the external load and therefore, dynamic forces should be also considered when the rigidity is evaluated. As a result, rigidity is not a property of the geometry only. The rigidity analysis requires the motion, inertia and all externally applied forces to be considered which complicates the process. To overcome this problem, another property called tensionability is defined and used which only depends on the geometry and expresses the potential of the manipulator for being rigid.

Tensionability: A cable-based manipulator is called tensionable at a given pose if and only if for any arbitrary external load there exists a finite spine force and a set of finite cable tensions to make the manipulator rigid.

Note that if a manipulator is tensionable and there is enough tensioning force available (by the spine and the cables), then the manipulator will be rigid under any external loading. In other words, tensionability and large enough tensioning force together provide a sufficient condition for the rigidity. The converse is that a manipulator may be rigid under a certain condition but not tensionable.

It is important to note that both rigidity and tensionability deal with the existence of the static equilibrium condition for the manipulator in which all the cables are in tension and hence, the manipulator does not collapse. However, they do not explain the nature of the equilibrium. Considering the stiffness of the manipulator, it may be rigid (meaning that it is in static equilibrium with all cables in tension) although the equilibrium might be an unstable one which implies that any small disturbance on the end-effector results in the collapse of the manipulator. It is known that the stability of the manipulator from the stiffness point of view is not specific to cable-based manipulators; however, it is shown in (Behzadipour & Khajepour, 2006) that the cable tensions may have a significant effect on the stiffness and even destabilization of the manipulator.

3. Tensionability

The goal of this section is to introduce an approach for the evaluation of tensionability in a cable-based manipulator. According to the definition, the tensionability of a manipulator must be evaluated for any arbitrary external load. In the following, a theorem is introduced which gives a sufficient condition for the manipulator to be tensionable.
The core idea of this theorem is to show that if positive tension (tensile force) can be generated in all the cables to any desired extent while the static equilib-

rium is satisfied in the absence of the external loads, then the manipulator can be rigidified under any arbitrary external load by having enough pretension in the cables.

Theorem 1.
A kinematically non-singular configuration of a cable-based manipulator is tensionable if for an arbitrary positive spine force $\mathbf{F}_s$ (compressive force), the static equilibrium equations of the manipulator have a solution with all positive cable tensions τ_i's.

$$\sum_i \tau_i \hat{\mathbf{u}}_i + \mathbf{F}_s = \mathbf{0} \tag{1}$$

$$\sum_i \mathbf{r}_i \times \tau_i \hat{\mathbf{u}}_i = \mathbf{0} \tag{2}$$

This theorem simply states that if the manipulator can stay rigid and statically balanced under an arbitrary compressive spine force, it is tensionable and thus can stay rigid for any external force and torque by choosing a large enough spine force.

Proof:
For the proof, it will be shown that such a manipulator can be made rigid for any arbitrary external load. The balance of forces for an arbitrary external force $\mathbf{F}_e$ applied at O' and moment $\mathbf{M}_e$ is:

$$\sum_i \tau_i \hat{\mathbf{u}}_i + \mathbf{F}_s + \mathbf{F}_e = \mathbf{0} \tag{3}$$

$$\sum_i \mathbf{r}_i \times \tau_i \hat{\mathbf{u}}_i + \mathbf{M}_e = \mathbf{0} \tag{4}$$

The above equations have a set of nontrivial solutions for τ_i's since the manipulator is assumed to be kinematically non-singular. Since the above set of equations is linear w.r.t. τ_i's, superposition can be applied to obtain the following two sets of equations:

$$\sum_i \tau_i^e \hat{\mathbf{u}}_i = -\mathbf{F}_e, \qquad \sum_i \mathbf{r}_i \times \tau_i^e \hat{\mathbf{u}}_i = -\mathbf{M}_e \tag{5}$$

$$\sum_i \tau_i^s \hat{\mathbf{u}}_i = -\mathbf{F}_s, \qquad \sum_i \mathbf{r}_i \times \tau_i^s \hat{\mathbf{u}}_i = \mathbf{0} \tag{6}$$

where $\tau_i^s + \tau_i^e = \tau_i$ for $i=1,2,...,m$. In this formulation, τ_i^s's are the cable forces to balance the spine force and are positive due to the assumption and τ_i^e's are the forces in the cables (positive or negative) due to the external force $\mathbf{F}_e$ and moment $\mathbf{M}_e$. If all τ_i^e's are positive, then τ_i's will be positive too and the cable-based manipulator is rigid. Otherwise, let $-\alpha^2 = \min_i(\tau_i^e)$ i.e. the most negative tension in the cables produced by the external load. Using the linearity of the static equilibrium equations in Eq. (6), cable tensions τ_i^s's can be increased by increasing f_s such that $\min_i(\tau_i^s) > \alpha^2$. As a result we have:

$$\min_i(\tau_i) \geq \min_i(\tau_i^s) + \min_i(\tau_i^e) > 0 \qquad (7)$$

Therefore, by increasing the spine force, the rigidity can be obtained and hence, the manipulator is tensionable.

The above theorem gives a sufficient condition for tensionability meaning that there might be situations in which the spine force cannot produce tension in all cables but the manipulator can be still rigidified. In those cases, sources other than spine may be used to generate tension in cables. An example of such cases will be studied in Section 5.1.

As a result from theorem 1, to evaluate the tensionability, instead of dealing with external load on the end-effector, we only need to show that the static equilibrium of the end-effector for an arbitrary spine force can be obtained by tensile forces in all of the cables. This will ensure that the manipulator is tensionable and thus can theoretically stand any external force and moment at the end-effector. By "theoretically" we mean that the required spine force and cable tensions are finite, although these forces may not be feasible due to the practical constraints. The above approach is used later in this paper to evaluate the tensionability of the new cable-based manipulators.

In the rest of this paper, some new designs of reduced DoF[1] cable-based manipulators are introduced. The target application of these manipulators is high-speed pick-and-place operations in which, small objects (less than 1kg) are moved with high speeds (more than 100 cycles per minute). High speed and acceleration requires low inertia which makes cable-based manipulators potential designs. However, most of the current spatial cable-based manipulators have 6 DoF while in pick-and-place applications, three translational axes of motion with a possible rotational DoF for reorienting the object are sufficient. In the designs presented in this work, cables are used to constrain the rotational motion of the end-effector in order to provide a pure translational mo-

---

[1] DoF: Degree of Freedom

tion. A more complete set of these designs can be found in (Khajepour et al., 2003). One of these designs, *DeltaBot*, has been prototyped at the University of Waterloo (Dekker et al. 2006). It can perform up to 150 cycles/minute of standard pick-and-place on small objects (less than 500gr).

4. BetaBot

In *BetaBot*, shown in Fig. 2, the upper triangle is the end-effector and the bottom one is the base. Three pairs of parallel cables are attached to the end-effector and wound by three winches after passing through guide holes on the winch frames. The winches are attached to the base. Each pair of cables forms a parallelogram such as *ABCD* as shown in the same figure. It is known that (Clavel, 1991), three parallelograms can eliminate the rotational motion of the end-effector. The spine is connected to the end-effector and base using two ball-and-sockets or one universal joint and one ball-and-socket. Therefore, the spine imposes no kinematic constraint on the end-effector.

The equivalent rigid link manipulator for *BetaBot* is obtained by replacing each cable with a slider with two ball-and-sockets at the ends. In this equivalent manipulator, there are 13 bodies, 12 ball-and-socket and 6 prismatic joints. The Gruebler equation gives $13 \times 6 - 12 \times 3 - 6 \times 5 = 12$ degrees of freedom. There are 6 trivial DoF's due to the twist of the sliders and there is also one constraint on each pair of sliders which forces their displacements to be the same (because each pair of cables is wound using a single winch). Therefore, the end-effector has $12 - 6 - 3 = 3$ DoF's which are translational.

Since the size of the end-effector plays no role in the kinematics of *BetaBot*, the end-effector can be shrunk to a point with three winches moving towards the center of the base accordingly. As a result, the kinematics of *BetaBot* becomes identical to that of a tripod (Mianowski & Nazarczuk, 1990), or the Tsai manipulator (Tsai, 1996).

The geometrical workspace of *BetaBot* is only limited by the maximum and minimum lengths of the spine assuming that there are no limitations on the cables' lengths and therefore, the shape of the workspace is a half sphere above the base whose radius is determined by the maximum length of the spine. It is clear that there is no possibility of any interference between the cables because of the non-rotating motion of the end-effector.
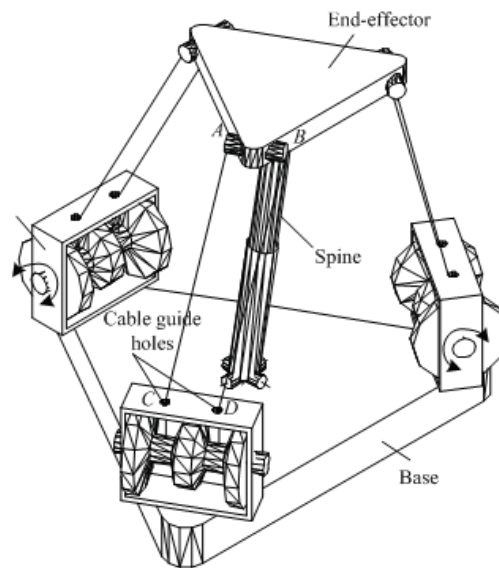
Figure 2. *BetaBot*: a 3 DoF cable-based manipulator with pure translational motion

4.1 Tensionability of *BetaBot*

BetaBot is tensionable everywhere in its workspace providing that certain conditions are enforced on the geometry of the manipulator as illustrated in Fig. 3:

Condition 1.   End-effector has a triangular shape as shown in Fig. 3. Each pair of the parallel cables is attached to one edge of the triangle,

Condition 2.   The guide holes on the winch frames are all on the same plane and form a triangle called Base triangle. This triangle is similar (and parallel) to the triangle of the end-effector but larger. As a result, the end-effector along with the cables form a convex region or a polyhedral

Condition 3.   Any two cables never become in-line

Condition 4.   The connection points of the spine, O and O' in Fig. 3, are on the base and end-effector triangles, respectively and have the same trilinear coordinates[2]. A direct result is that the spine never intersects with the faces of the polyhedral of Condition 2 (even if the spine and cables are extended from the base side).

---

[2] The ratio between their distances from the triangle vertices are the same
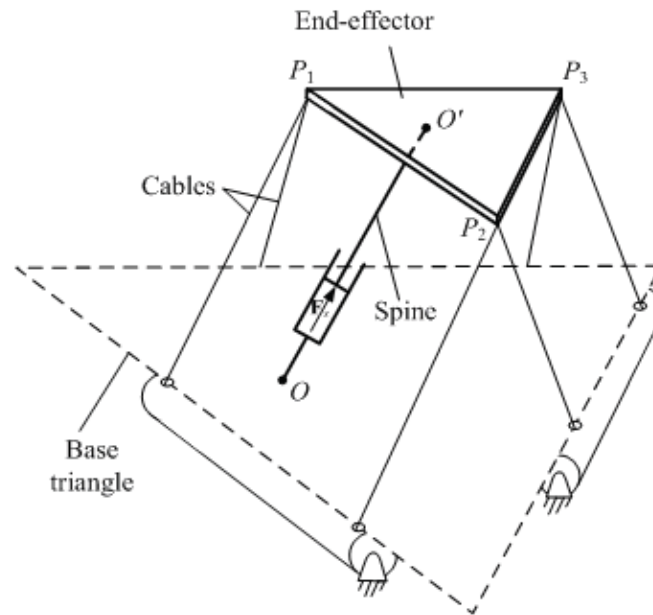
Figure 3. The prefered geometry of BetaBot which guarantees its tensionability

To prove the tensionability, we use the theorem given in the last section. Therefore, for *BetaBot* to be tensionable, an arbitrary positive spine force should be balanced by positive cable tensions. For the proof, a geometrical approach is employed which shows that the solution of static equilibrium equations consists of a set of positive tensions. Since the proof is lengthy, it is presented by four lemmas and one theorem.

Lemma 1.
If *BetaBot* meets the above mentioned conditions, then the three planes each of which formed by one pair of parallel cables intersect at one point such as R (see Fig. 4).

Proof.
In Fig. 4, consider the plane that includes P_1, P_2, B_1 and B_2: If B_1P_1 and B_2P_2 are extended, they meet at a point called R and form triangle $\Delta B_1 R B_2$ (otherwise they will be parallel which implies the end-effector and the base triangle are equal contradicting Condition 2). In this triangle we have:

$$P_1P_2 \| B_1B_2 \Rightarrow \frac{RP_2}{RB_2} = \frac{P_1P_2}{B_1B_2} \tag{8}$$
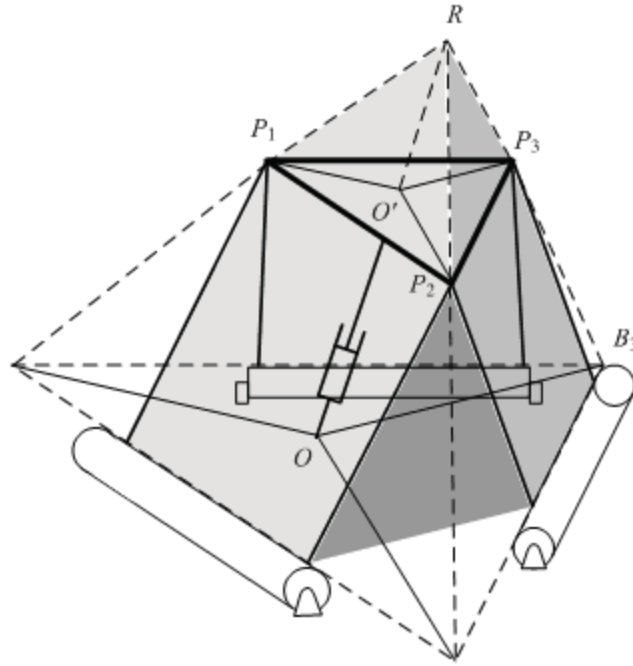
Figure 4. The three planes formed by parallel cables and the spine meet at a point called R.

Now, consider the second plane that includes P_2, P_3, B_2 and B_3: If B_2P_2 and B_3P_3 are extended, they meet at a point called R'. Note that R' and R are both on the line of P_2B_2. In triangle $\Delta B_2 R' B_3$ we have:

$$P_2P_3 \| B_2B_3 \Rightarrow \frac{R'P_2}{R'B_2} = \frac{P_2P_3}{B_2B_3} \tag{9}$$

Also, we know from Condition 2 that $\Delta P_1P_2P_3$ is similar to $\Delta B_1B_2B_3$ and hence:

$$\frac{P_1P_2}{B_1B_2} = \frac{P_2P_3}{B_2B_3} \tag{10}$$

which, by considering Eqs. (8, 9), results in:

$$\frac{R'P_2}{R'B_2} = \frac{RP_2}{RB_2} \tag{11}$$

Considering that R and R' are on the same line, Eq. (11) states that R and R' are coincident and a similar reasoning shows that the third plane also passes through R.

Lemma 2.
In *BetaBot*, the direction of the spine passes through point R as found in Lemma 1.

Proof.
Assume point R, as given in Fig. 4, is connected to O' and extended to intersect the base at point O'' (not shown in the figure). It is needed to show that O'' coincides with O. This is shown by the following relations:

$$\Delta RO''B_1: \quad O'P_1 \| O''B_1 \quad \Rightarrow \quad \frac{O'P_1}{O''B_1} = \frac{RP_1}{RB_1} \tag{12}$$

$$\Delta RB_1B_2: \quad P_1P_2 \| B_1B_2 \quad \Rightarrow \quad \frac{RP_1}{RB_1} = \frac{RP_2}{RB_2} \tag{13}$$

$$\Delta RO''B_2: \quad O'P_2 \| O''B_2 \quad \Rightarrow \quad \frac{O'P_2}{O''B_2} = \frac{RP_2}{RB_2} \tag{14}$$

As a result:

$$\frac{O'P_1}{O''B_1} = \frac{RP_1}{RB_1} = \frac{RP_2}{RB_2} = \frac{O'P_2}{O''B_2} \tag{15}$$

and using a similar approach on $\Delta R O'' B_3$, ΔRB_1B_3 and $\Delta R O'' B_1$, we have:

$$\frac{O'P_1}{O''B_1} = \frac{RP_1}{RB_1} = \frac{RP_3}{RB_3} = \frac{O'P_3}{O''B_3} \tag{16}$$

which results in:

$$\frac{O'P_1}{O''B_1} = \frac{O'P_2}{O''B_2} = \frac{O'P_3}{O''B_3} \tag{17}$$

However, Condition 4 states that:

$$\frac{O'P_1}{OB_1} = \frac{O'P_2}{OB_2} = \frac{O'P_3}{OB_3} \tag{18}$$

By comparing Eqs. (17) and (18), it is concluded that O' coincides with O''.

For the next two lemmas, Fig. 5 is used. In Fig. 5a, the region bounded by the cables is the polyhedral of Condition 2. The faces of this polyhedral are formed by the cables. This polyhedral has six faces S_{ij} (i,j=1,2,3 and i<j) each of which formed by two cable directions: $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{u}}_j$. In Fig. 5b, the force diagram of the end-effector is shown. $\mathbf{F}_s$ is the spine force and each force vector $\sigma_i \hat{\mathbf{u}}_i$ represents the force of two parallel cables and hence σ_i is the total tension of the two parallel cables along $\hat{\mathbf{u}}_i$. Now, the next two lemmas follow.
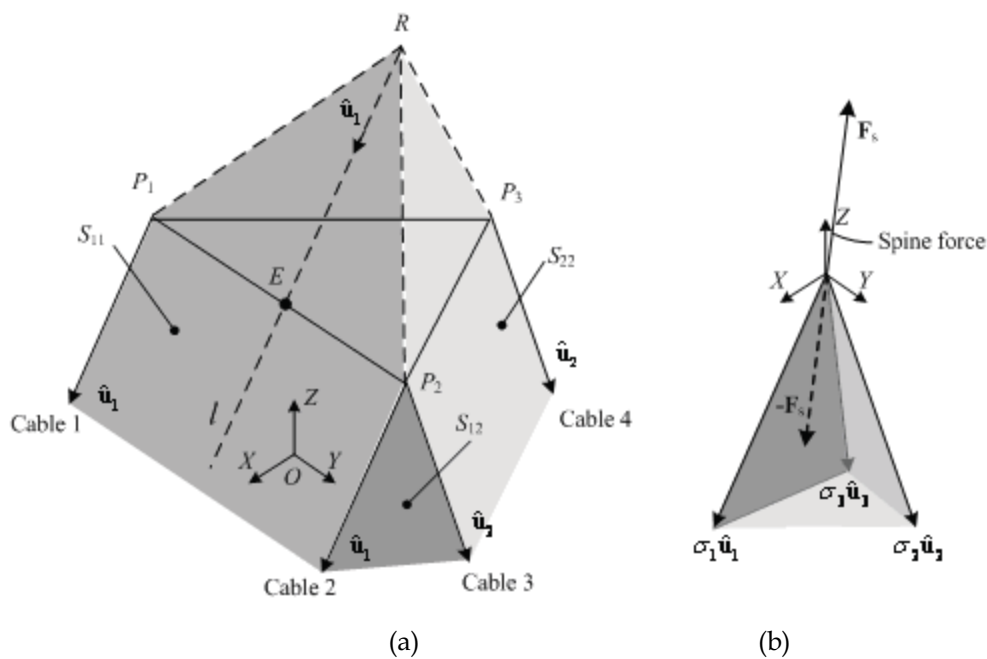
Figure 5. a) The polyhedra formed by the cables in *BetaBot* b) The cone of the cable forces

Lemma 3.

According to Fig. 5a, the half line Rl which starts from R in the direction of $\hat{\mathbf{u}}_1$ intersects line segment P_1P_2 at E.

Proof.

It is obvious that half line Rl intersects the line of P_1P_2 otherwise, they should be in parallel which implies that the first pair of cables are in-line contradicting Condition 3. The major effort of this proof is to show that the intersection occurs between P_1 and P_2 , i.e. $E \in P_1P_2$. For the proof, it is sufficient to show that E belongs to the polyhedral of Fig. 5a. Let $\mathbf{E}$ be the position vector of E with respect to $OXYZ$ reference frame. It is known that:

$$\mathbf{E} = \mathbf{R} + \alpha \hat{\mathbf{u}}_1 \tag{19}$$

where $\mathbf{R}$ is the position vector of R and a is a positive real number. The polyhedral of Fig. 5a is represented as a set of points in the Cartesian space by:

$$\mathbf{\Phi} = \left\{ \phi \in \mathsf{R}^3 \middle| \hat{\mathbf{n}}_{ij} \cdot \mathbf{\phi} \le d_{ij} \quad i, j = 1,2,3 \ i \le j \right\} \tag{20}$$

where the center of coordinate O is set to be inside the polyhedral, $\hat{\mathbf{n}}_{ij}$ is the unit vector normal to face S_{ij} pointing outward the polyhedral, and d_{ij} is the distance between face S_{ij} and the center of coordinates O.
In order to have $E \in \mathbf{\Phi}$, the following six inequalities should be satisfied:

$$\hat{\mathbf{n}}_{ij} \cdot \mathbf{E} \le d_{ij} \quad i, j = 1, 2, 3 \quad i \le j \tag{21}$$

For $i=j=1$, we have:

$$\begin{aligned}
\hat{\mathbf{n}}_{11} \cdot \mathbf{E} &= \hat{\mathbf{n}}_{11} \cdot \left(\mathbf{R} + \alpha \hat{\mathbf{u}}_1 \right) \\
&= \hat{\mathbf{n}}_{11} \cdot \mathbf{R} + \alpha \hat{\mathbf{n}}_{11} \cdot \hat{\mathbf{u}}_1
\end{aligned} \tag{22}$$

since $R \in S_{11}$ and $\hat{\mathbf{n}}_{11} \cdot \hat{\mathbf{u}}_1 = 0$, it is concluded that $\hat{\mathbf{n}}_{11} \cdot \mathbf{E} = d_{11} \le d_{11}$.
For $i=j=2$, we have:

$$\begin{aligned}
\hat{\mathbf{n}}_{22} \cdot \mathbf{E} &= \hat{\mathbf{n}}_{22} \cdot \left(\mathbf{R} + \alpha \hat{\mathbf{u}}_1 \right) \\
&= \hat{\mathbf{n}}_{22} \cdot \mathbf{R} + \alpha \hat{\mathbf{n}}_{22} \cdot \hat{\mathbf{u}}_1
\end{aligned} \tag{23}$$

it is known that $R \in S_{22}$ hence, we have:

$$\hat{\mathbf{n}}_{22} \cdot \mathbf{E} = d_{22} + \alpha \hat{\mathbf{n}}_{22} \cdot \hat{\mathbf{u}}_1 \overset{?}{\le} d_{22} \tag{24}$$

which requires $\alpha \hat{\mathbf{n}}_{22} \cdot \hat{\mathbf{u}}_1 \le 0$. To show this, an auxiliary point with position vector $\mathbf{P}_2 + \alpha \hat{\mathbf{u}}_1$ is considered. This point is on cable 2 according to Fig. 5a and hence belongs to polyhedral $\mathbf{\Phi}$. Therefore, one of the inequalities that this point should satisfy is:

$$\hat{\mathbf{n}}_{22} \cdot \left(\mathbf{P}_2 + \alpha \hat{\mathbf{u}}_1 \right) \le d_{22} \tag{25}$$

P_2 is a vertex of the end-effector triangle and thus, $P_2 \in S_{22}$ which implies that

$\hat{n}_{22} \cdot P_2 = d_{22}$. Using this in Eq. (25) results:

$$\hat{\mathbf{n}}_{22} \cdot \left(\mathbf{P}_2 + \alpha \hat{\mathbf{u}}_1 \right) = d_{22} + \alpha \hat{\mathbf{n}}_{22} \cdot \hat{\mathbf{u}}_1 \leq d_{22}$$
$$\Rightarrow \alpha \hat{\mathbf{n}}_{22} \cdot \hat{\mathbf{u}}_1 \leq 0$$
(26)

which was needed to satisfy inequality (24). Similar proof is used for $i=j=3$, by taking $\mathbf{P}_1 + \alpha \hat{\mathbf{u}}_1$ as the auxiliary point.

For $i=1$ and $j=2$, the inequality to be satisfied is:

$$\hat{\mathbf{n}}_{12} \cdot \mathbf{E} \leq d_{12}$$
$$\text{or} \quad \hat{\mathbf{n}}_{12} \cdot \left(\mathbf{R} + \alpha \hat{\mathbf{u}}_1 \right) \leq d_{12}$$
$$\text{or} \quad \hat{\mathbf{n}}_{12} \cdot \mathbf{R} + \alpha \hat{\mathbf{n}}_{12} \hat{\mathbf{u}}_1 \leq d_{12}$$
(27)

Again, since $R \in \boldsymbol{\Phi}$, we have $\hat{\mathbf{n}}_{12} \cdot \mathbf{R} \leq d_{12}$. Therefore for inequality (27) to hold true, it is sufficient to show that $\alpha \hat{\mathbf{n}}_{12} \hat{\mathbf{u}}_1 = 0$ which is obvious because $\hat{\mathbf{u}}_1$ is normal to $\hat{\mathbf{n}}_{12}$. A similar approach works for $i=1$, $j=3$. Finally, for $i=2$, $j=3$, we should have:

$$\hat{\mathbf{n}}_{23} \cdot \mathbf{E} = \hat{\mathbf{n}}_{23} \cdot \left(\mathbf{R} + \alpha \hat{\mathbf{u}}_1 \right) \leq d_{23}$$
(28)

Again, we have $R \in \boldsymbol{\Phi}$ and thus $\hat{\mathbf{n}}_{23} \cdot \mathbf{R} \leq d_{23}$. It would be sufficient to show that $\alpha \hat{\mathbf{n}}_{23} \hat{\mathbf{u}}_1 \leq 0$. For this purpose, consider the cone that is formed by extending S_{12}, S_{13} and S_{23}. This cone has a similar shape to the one of Fig. 5b. The edges of this cone are along $\hat{\mathbf{u}}_1$, $\hat{\mathbf{u}}_2$ and $\hat{\mathbf{u}}_3$. Let the apex of this cone be called W and take $\mathbf{W} + \alpha \hat{\mathbf{u}}_1$ as an auxiliary point which belongs to the cone and hence satisfies the following inequality:

$$\hat{\mathbf{n}}_{23} \cdot \left(\mathbf{W} + \alpha \hat{\mathbf{u}}_1 \right) \leq d_{23}$$
(29)

Now, since $\hat{\mathbf{n}}_{23} \cdot \mathbf{W} = d_{23}$, it is concluded that $\alpha \hat{\mathbf{n}}_{23} \hat{\mathbf{u}}_1 \leq 0$ which satisfies inequality (28) and hence $E \in \boldsymbol{\Phi}$, which completes the proof.

Lemma 4.
According to the force diagram of Fig. 5b, the spine force is balanced by positive σ_i's $i=1, 2, 3$ i.e.:

$$\sigma_1 \hat{\mathbf{u}}_1 + \sigma_2 \hat{\mathbf{u}}_2 + \sigma_3 \hat{\mathbf{u}}_3 = -\mathbf{F}_s$$
(30)

Proof.
We first show that the direction of spine force lies inside the cone formed by cables direction vectors: $\hat{\mathbf{u}}_1$, $\hat{\mathbf{u}}_2$ and $\hat{\mathbf{u}}_3$. For this purpose, we use Condition 4 which states the spine never intersects any faces of the polyhedral of cables (Fig. 5a). Considering that the faces of the cone of Fig. 5b are parallel to S_{12}, S_{13}

and S_{23} , respectively, it is understood that $\mathbf{F}_s$ never touches the faces of the cone and hence, is inside the cone. Based on the definition of a convex cone (Yamaguchi, 2002), the cone of Fig. 5b can be expressed as:

$$\Psi = \left\{ \psi \in \mathsf{R}^3 \, \middle| \, \psi = \sigma_1 \hat{\mathbf{u}}_1 + \sigma_2 \hat{\mathbf{u}}_2 + \sigma_3 \hat{\mathbf{u}}_3 \quad \sigma_1, \sigma_2, \sigma_3 \geq 0 \right\} \tag{31}$$

Now, since $\mathbf{F}_s$ is always inside the cone, it can be considered as the position vector for a point inside the cone. As a result, the definition of the cone given in Eq. (31) states that:

$$-\mathbf{F}_s = \sigma_1 \hat{\mathbf{u}}_1 + \sigma_2 \hat{\mathbf{u}}_2 + \sigma_3 \hat{\mathbf{u}}_3 \quad \sigma_1, \sigma_2, \sigma_3 \geq 0 \tag{32}$$

which completes the proof.

Theorem 2.
If *BetaBot* satisfies Conditions 1-4 as explained before, it is tensionable everywhere in its workspace.

Proof.
For the proof, we show that a positive spine force is statically balanced by positive tension in cables. For this purpose, first consider the force equilibrium of the end-effector (see Fig. 5b). According to Lemma 4, the force equilibrium on the end-effector is met by positive σ_1, σ_2 and σ_3, where for instance, σ_1 is the total tension of cables 1 (τ_1) and 2 (τ_2). Now, let:

$$\tau_1 = \frac{P_2 E}{P_1 P_2} \sigma_1 \qquad \tau_2 = \frac{P_1 E}{P_1 P_2} \sigma_1 \tag{33}$$

therefore, we have $\tau_1, \tau_2 > 0$, $\tau_1 + \tau_2 = \sigma_1$. Since E is on $P_1 P_2$ (Lemma 3) and using Eq. (33), the moments of $\tau_1 \hat{\mathbf{u}}_1$ and $\tau_2 \hat{\mathbf{u}}_1$ about E cancel each other. As a result, force vector $\sigma_1 \hat{\mathbf{u}}_1$ applied at E is an equivalent for cable forces $\tau_1 \hat{\mathbf{u}}_1$ and $\tau_2 \hat{\mathbf{u}}_1$. Note that the line of action of this equivalent force passes through R. Similarly, other cable forces can be replaced by equivalent forces whose lines of action pass through R. Now, based on Lemma 2, the direction of the spine also passes through the same point (R) and hence, the equilibrium of the moments is also satisfied which completes the proof.

5. DishBot

DishBot is another spatial cable-based manipulator with translational motion shown in Fig. 6. In *DishBot*, two independent sets of cables are used. The first set, called drive cables, moves the end-effector and the second set, named passive cables, eliminates the rotation. The spine, in this design, is composed of a bar sliding inside a collar along with a spring. The collar is connected to the center of the base by a universal joint. The spring is fixed to the collar from one end and to the bar from the other end. It produces a compressive force to maintain tension in the drive cables. A ball-and-socket is used to connect the end-effector to the spine.

As seen in Fig. 6b, drive cables are connected to the tip of the spine similar to the tripod design of Landsberger (Landsberger & Sheridan, 1985) or Mianowski's robot (Mianowski & Nazarczuk, 1990). Each cable is pulled and collected by a winch that is hinged to the base such that it can align itself with the cable direction.

In Fig. 6c, the passive cables are shown. Each passive cable starts from the end-effector on the top and proceeds down to the base where it bends around a pulley (such as pulley *i* in Fig. 6c) and goes towards the spine. Using another pulley (such as pulley *j*) on the collar to guide the cable, it reaches the bottom end of the spine bar. Each passive cable is in series with a tensile pre-tensioning spring to produce tensile force in the cable.

The length of the passive cables are constant which makes the end-effector stay parallel to the base. However, the perfect parallelness is only obtained if the radii of the two guiding pulleys are zero and the pulleys on the collar coincide with the center of the universal joint.

Fig. 7 shows how the passive cables make the end-effector stay parallel to the base in a similar 2D manipulator. In practice, complete parallelness is not achieved since the pulleys have non-zero radius and they cannot be placed at the center of the universal joint; however, the error can be minimized by proper sizing and positioning of the pulleys.

Assuming the passive cables are always taut and maintain the orientation of the end-effector, the kinematics of *DishBot* is identical to a tripod since it only depends on the drive cables. The inverse and direct kinematics have closed form solutions (Tsai, 1996). The geometrical workspace of *DishBot* is the half sphere above the base whose radius is the maximum length of the spine. Similar to *BetaBot*, the size of the workspace is only limited by the maximum and minimum length of the spine. There is no possibility of interference between the cables and spine as long as:

1. The spine is located inside the cone of drive cables,
2. The cone of drive cables is inside the prism of passive cables.

(a)

(b)   (c)
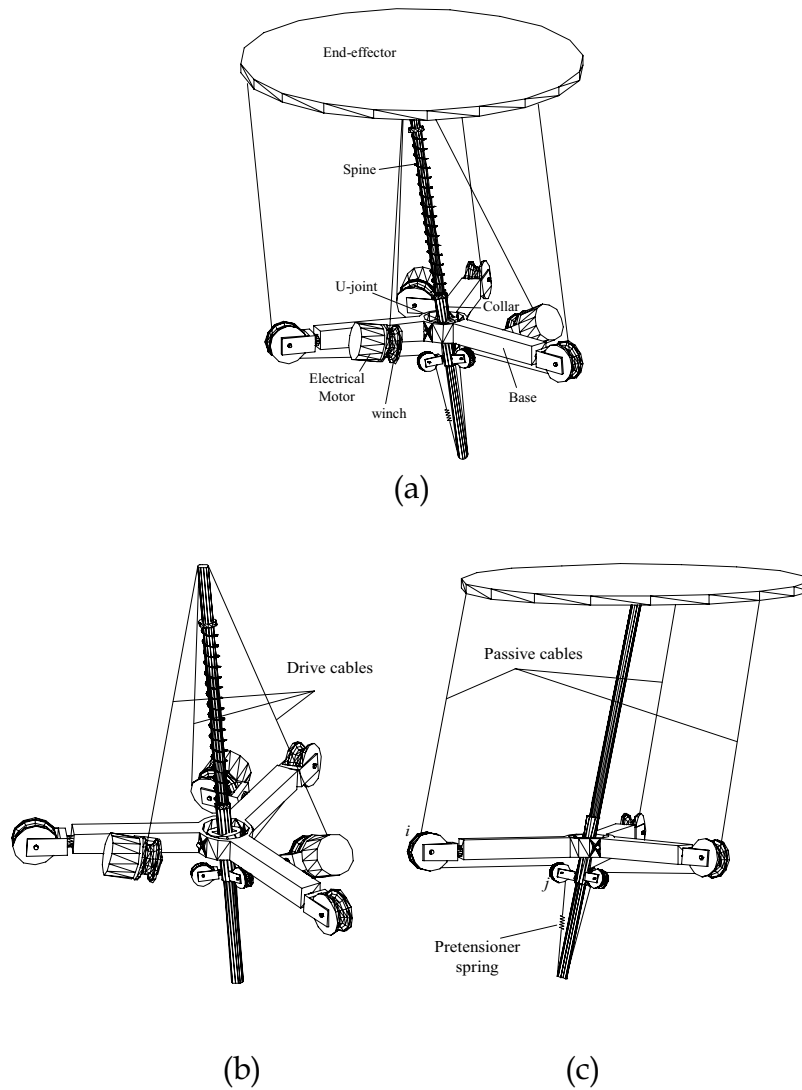
Figure 6. a) *DishBot*: a 3 DoF translational cable-based manipulator with two independent sets of cables. b) The configuration of the drive cables c) The configuration of the passive cables
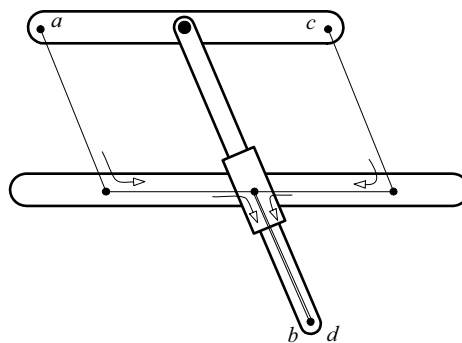
Figure 7. Assuming the lengths of the two cables *ab* and *cd* are equal, the end-effector always remains parallel to the base

5.1 Tensionability of *DishBot*

For the tensionability of *DishBot,* Theorem 1 given in Section 3 is not sufficient. The reason is that the spine force only affects the tension of drive cables and has no influence on the passive ones. As a result, it cannot leverage all the tensions. However, it should be noted that each passive cable has a pre-tensioning spring to maintain the tension. Therefore, the tensionability can be still proved based on its definition in Section 2. For the proof of tensionability, we use the idea of Theorem 1 (not the theorem itself), i.e. tension can be generated in the cables to any extent while the static equilibrium is satisfied.

The free body diagram of the end-effector is shown in Fig. 8. The passive cables are in parallel with the spine and their tensions are shown by a superscript p while for the tension of drive cables, a superscript d is used. The static equilibrium equations are found to be:

$$\sum \mathbf{F} = \tau_1^d \hat{\mathbf{u}}_1 + \tau_2^d \hat{\mathbf{u}}_2 + \tau_3^d \hat{\mathbf{u}}_3 + \left(f_s - \tau_1^d - \tau_2^d - \tau_3^d \right) \hat{\mathbf{w}} = \mathbf{0} \tag{34}$$

$$\sum \mathbf{M} = -\tau_1^p \left(\mathbf{r}_1 \times \hat{\mathbf{w}} \right) - \tau_2^p \left(\mathbf{r}_2 \times \hat{\mathbf{w}} \right) \hat{\mathbf{u}}_1 - \tau_3^p \left(\mathbf{r}_1 \times \hat{\mathbf{w}} \right) = \mathbf{0} \tag{35}$$

where $\tau_i^d \hat{\mathbf{u}}_i$'s are drive cable forces (i=1,2,3), $\tau_j^p \hat{\mathbf{w}}$'s are passive cable forces, $\mathbf{r}_j$'s are the position vectors of the anchor points of the passive cables (j=1,2,3) and $f_s \hat{\mathbf{w}}$ is the spine force.
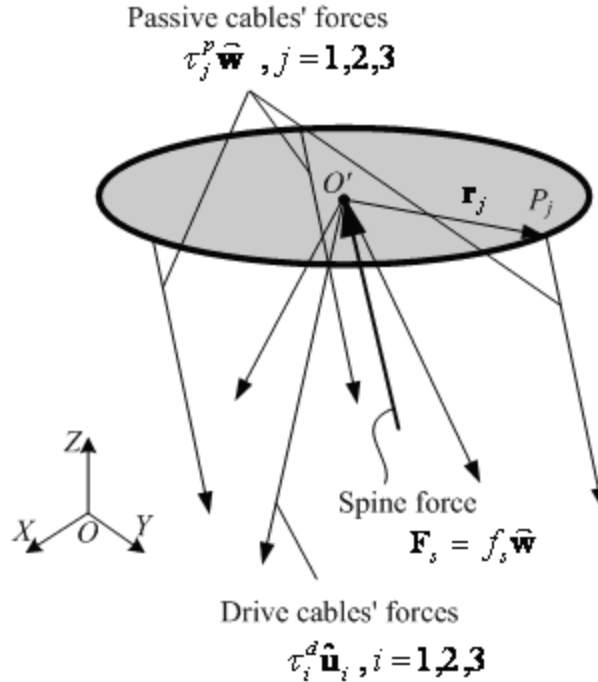
Figure 8. Free body diagram of DishBot's end-effector

A quick inspection of the above equations shows that Eq. (35), which is the equilibrium of the moments, is a set of homogenous equations independent from the spine force which, in general, results in zero tension for passive cables. The tension of the drive cables are found from Eq. (34). Note that the drive cables form a cone which contains the spine and hence, using Lemma 4, the drive cable tensions are positive as long as the equivalent spine force, $f_s - \tau_1^p - \tau_2^p - \tau_3^p$, is positive (compressive). As a conclusion, tension in the drive cables can be generated to any desired level by choosing a large enough spine force but it does not affect the tension in the passive cables.

In order for *DishBot* to be tensionable, we also need to show that the tension in the passive cables can be increased to any desired level. For this purpose, note that, in Fig. 8, if O' is the geometrical center of the three anchor points of the passive cables (P_1, P_2 and P_3) then Eq. (35) has non-zero solutions for passive cable tensions. In this case, the solution would be $\tau_1^p = \tau_2^p = \tau_3^p = \tau^p$ where τ^p is an arbitrary real value and thus can be positive. It is known that such a geometrical center coincides with the centroid of triangle $\Delta P_1 P_2 P_3$. As a result, if O' is the centroid of triangle $\Delta P_1 P_2 P_3$, positive equal tensions in passive cables are determined only by the pre-tensioning springs.

As a conclusion, *DishBot* is tensionable as long as the following conditions are met:

1. O' is the centroid of $\Delta P_1 P_2 P_3$,
2. The pretension of the pre-tensioning springs are equal (τ^p) and positive (tensile),
3. The spine lies inside the cone of the drive cables.
4. The spine force satisfies $f_s - \tau_1^p - \tau_2^p - \tau_3^p > 0$ which means $f_s > 3\tau^p$.

6. Planar cable-based manipulators with translational motion

Planar manipulators with translational motion (in XY plane) are sufficient for many industrial pick-and-place applications such as packaging and material handling. Simplicity of these manipulators compared to spatial ones further facilitates their applications where a two axis motion is sufficient (Chan, 2005). Two new designs of planar cables-based manipulators with translational motion are studied here that are tensionable everywhere in their workspace. Schematic diagrams of these manipulators are shown in Fig. 9. The spine is connected to the base and end-effector by revolute joints. The end-effector is constrained by three cables. Two of the cables form a parallelogram which eliminates the rotation of the end-effector as long as the cables are taut. As a result, the end-effector can only move in X and Y directions.

(a)

(b)
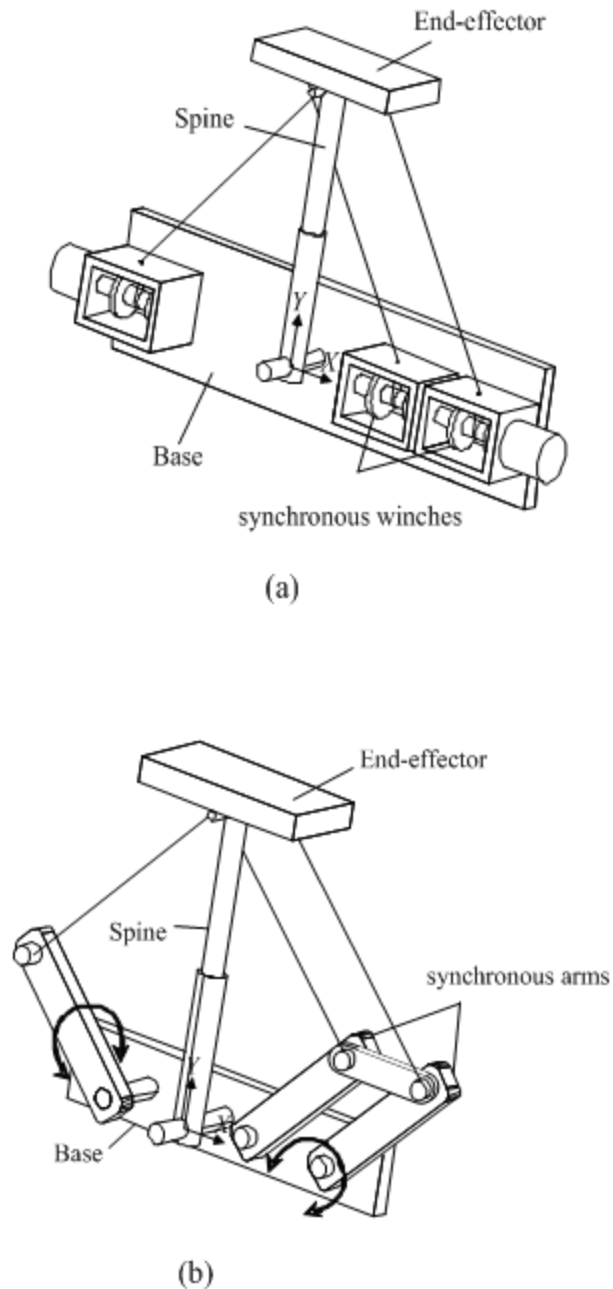
Figure 9. Planar cable-based manipulators with pure translational degrees of freedom

In the first design (Fig. 9a), the parallelogram is maintained by two winches with a common shaft which makes them move simultaneously and hence, keep the cable lengths equal. Similar to *BetaBot* and *DishBot*, the workspace of this manipulator is only limited by the minimum and maximum lengths of the

spine and hence it can theoretically span a half circle above the base. In the second design (Fig. 9b), a pair of synchronous rotating arms preserves the parallelogram without changing the length of the cables and therefore, possess a smaller workspace. The synchronization can be obtained by a pair of pulleys and a timing belt or a simple 4-bar parallelogram as seen in Fig. 9b.

The kinematics of these manipulators consist of a single planar cone and hence easy to formulate for both direct and inverse solutions. In this paper, however, our main focus is on the their tensionability and rigidity which is presented in the following.

6.1 Tensionability of Planar Manipulators

The planar manipulators of Fig. 9 are both tensionable everywhere in their workspaces. This can be proved using an approach similar to the one that was used for *BetaBot*. There are two geometrical conditions that should be met for the tensionability of these two manipulators. As depicted in Fig. 10a, these two conditions are as follows:
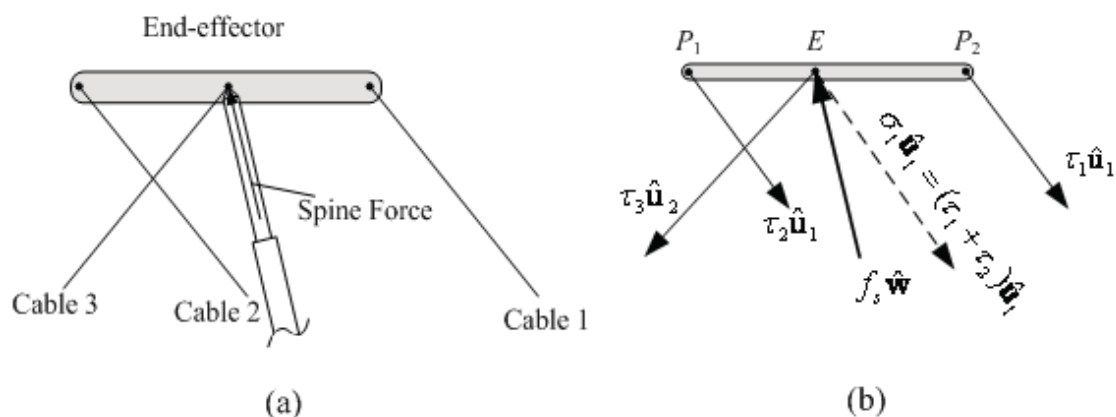
Figure 10. a) The configuration of the cables and spine in planar manipulators, b) The free body diagram of the end-effector

Condition 1.   Cable 1 (Fid. 10a) is always on the right and Cable 3 is always on the left side of the spine. This is obtained if the spine is hinged to the base at a proper point between the two sets of cables.

Condition 2.   On the end-effector, the spine and Cable 3 are concurrent at point E which is located somewhere between Cables 1 and 2.

To prove the tensionability, we show that a compressive spine force can be balanced by positive tensions in the cables. The proof is quite similar to the one of Theorem 2 and is briefly explained here.

We first consider the force equilibrium of the end-effector subject to a compressive spine force. According to the free body diagram shown in Fig. 10b, we have:

$$\sigma_1 \hat{\mathbf{u}}_1 + \tau_3 \hat{\mathbf{u}}_2 + f_s \hat{\mathbf{w}} = \mathbf{0} \quad \Rightarrow \quad \sigma_1 \hat{\mathbf{u}}_1 + \tau_3 \hat{\mathbf{u}}_2 = -f_s \hat{\mathbf{w}} \tag{36}$$

Due to Condition 1, the direction of the spine, $\hat{\mathbf{w}}$ is located between $\hat{\mathbf{u}}_1$ and $\hat{\mathbf{u}}_2$ (cable directions). Therefore, the projection of $-f_s \hat{\mathbf{w}}$ on $\hat{\mathbf{u}}_1$ and $\hat{\mathbf{u}}_2$ will be positive and hence $\sigma_1, \tau_3 > 0$. Now, let:

$$\tau_1 = \frac{P_1 E}{P_1 P_2} \sigma_1 \quad \text{and} \quad \tau_2 = \frac{P_2 E}{P_1 P_2} \sigma_1 \tag{37}$$

It is clear that $\tau_1 + \tau_2 = \sigma_1$. Since $\sigma_1 > 0$ and due to the distribution given in Eq. (37), the moment of $\tau_1 \hat{\mathbf{u}}_1$ about E cancels the one of $\tau_2 \hat{\mathbf{u}}_1$ and hence, these two forces can be replaced by $\sigma_1 \hat{\mathbf{u}}_1$ without violating the static equilibrium. Finally, since all three forces on the end-effector, $\sigma_1 \hat{\mathbf{u}}_1$, $\tau_1 \hat{\mathbf{u}}_1$ and $\tau_2 \hat{\mathbf{u}}_1$, are concurrent at E, the equilibrium of the moments is also met which completes the proof.

7. Conclusion

In this paper, several new cable-based manipulators with pure translational motion were introduced and their rigidity where thoroughly studied. The significance of these new designs can be summarized in two major advantages over the other cable-based manipulators:

1. Cables are utilized to provide kinematic constraints to eliminate rotational motion of the end-effector. In many industrial applications, reduced DoF manipulators are sufficient to do the job at a lower cost (less number of axes).

2. These manipulators can be rigidified everywhere in their workspace using a sufficiently large pretension in the cables.

In order to study the rigidity of these manipulators, the concept of tensionability was used and a theorem was given to provide a sufficient condition for tensionability. Using this theorem, tensionability of each manipulator was proved

using line geometry and static equilibrium in vector form. For each of these manipulators, it was shown that as long as certain conditions are met by the geometry of the manipulator, the tensionable workspace in which the manipulator can be rigidified, is identical to the geometrical workspace found from the kinematic analysis.

BetaBot and the planar manipulators are tensionable everywhere and can be rigidified only by a sufficiently large spine force. In *DishBot*, on top of the geometrical conditions, a relation between the spine force and pre-tensioning springs of passive cables should be also satisfied to maintain the rigidity of the manipulator.

8. References

Barrette G.; Gosselin C. (2005), Determination of the dynamic workspace of cable-   driven planar parallel mechanisms, *Journal of Mechanical Design*, Vol. 127, No.   3, pp. 242-248

Behzadipour S. (2005), *High-speed Cable-based Robots with Translational Motion*, PhD   Thesis, University of Waterloo, Waterloo, ON, Canada

Behzadipour S.; Khajepour A., (2006), Stiffness of Cable-based Parallel Manipulators   with Application to the Stability Analysis, *ASME Journal of Mechanical Design*,   Vol. 128, No. 1, 303-310

Chan E. (2005), *Design and Implementation of a High Speed Cable-Based Planar Parallel Manipulator*, MASc Thesis, University of Waterloo, Waterloo, ON, Canada.

Clavel R., (1991), *Conception d'un robot parallele rapide a 4 degres deliberate*, PhD Thesis, EPFL, Lausanne

Dekker R. & Khajepour A. & Behzadipour S. (2006), Design and Testing of an Ultra   High-Speed Cable Robot", *Journal of Robotics and Automation*, Vol. 21, No. 1,   pp. 25-34

Ferraresi C.; Paoloni M.; Pastorelli S.; Pescarmona F, (2004), A new 6-DOF parallel   robotic structure actuated by wires: the WiRo-6.3, *Journal of Robotics Systems,*   Vol. 21, No. 11, pp. 581-595

Gallina P; Rosati G., (2002), Manipulability of a planar wire driven haptic device,   *Mechanism and Machine Theory*, Vol. 37, pp. 215-228

Gouttefarde M.; Gosselin C. (2006), Analysis of the wrench-closure workspace of   planar parallel cable-driven mechanisms, *IEEE Transactions on Robotics*, Vol.   22, No. 3, pp. 434-445

Kawamura S.; Choe W.; Tanak S. Pandian S.R., (1995),Development of an ultrahigh speed robot FALCON usign wire driven systems, *Proceedings of IEEE   International Conference on Robotics and Automation*, pp. 215-220, IEEE, 1995

Khajepour A. & Behzadipour S. & Dekker R. & Chan E. (2003), Light Weight
Parallel Manipulators Using Active/Passive Cables, *US patent provisional
file No. 10/615,595*

Landsberger S.E.; Sheridan T.B., (1985), A new design for parallel link manipu-
lators, *Proceedings of the International Conference on Cybernetics and Society,*
pp. 81-88, Tuscon AZ, 1985

Landsberger S.E.; Shanmugasundram P. (1992), Workspace of a Parallel Link
Crane,
*Proceedings of IMACS/SICE International Symposium on Robotics, Mechatron-
ics and Manufacturing Systems*, pp. 479-486, 1992

Ming A.; Higuchi T. (1994a), Study on multiple degree-of-freedom positioning
mechanism using wires (Part1), *International Journal of Japan Society of
Precision Engineering*, Vol. 28, No. 2, pp. 131-138

Ming A.; Higuchi T. (1994b), Study on multiple degree-of-freedom positioning
mechanism using wires (Part2), *International Journal of Japan Society of
Precision Engineering*, Vol. 28, No. 3, pp. 235-242

Ogahara Y.; Kawato Y.; Takemura K.; Naeno T., (2003), A wire-driven minia-
ture five fingered robot hand using elastic elements as joints, *Pro-
ceedings of IEEE/RSJ International Conference on Intelligent Robots and Sys-
tems*, pp. 2672-2677, Las Vegas, Nevada, 2003

Oh s.; Makala K. K.; Agrawal S., (2005a) Dynamic modeling and robust con-
troller design of a two-stage parallel cable robot, *Multibody System Dy-
namics*, Vol. 13, No. 4, pp. 385-399

Oh s.; Makala K. K.; Agrawal S., Albus J. S. (2005b), A dual-stage planar cable
robot: Dynamic modeling and design of a robust controller with posi-
tive inputs, *ASME Journal of Mechanical Design*, Vol. 127, No. 4, pp. 612-
620

Pusey j.; Fattah A.; Agrawal S.; Messina E., (2004), Design and workspace
analysis of a 6-6 cable-suspended parallel robot, *Mechanisms and Ma-
chine Theory*, Vol. 39, No. 7, pp. 761-778

Robers G.R; Graham T.; Lippitt T. (1998), On the inverse kinematics, statics,
and fault tolerance of cable-suspended robots, *Journal of Robotic Sys-
tems*, Vol. 15, No. 10, pp. 581-597

Stump E.; Kumar V., (2006), Workspace of cable-actuated parallel manipula-
tors, *ASME Journal of Mechanical Design*, Vol. 128, No. 1, pp. 159-167

Tadokoro S.; Nishioka S.; Kimura T. (1996), On fundamental design of wire
configurations of wire-driven parallel manipulators with redundancy,
ASME Proceeding of Japan/USA Symposium on Flexible Automation, pp.
151-158

Tsai L-W., (1996), Kinematics of A Three DOF Platform With Three Extensible
Limbs, In *Recent Advances in Robot Kinematics*, Lenarcic J. and Parenti-
Castelli V., pp. 401-410, Kluwer Academic, Netherlands

Verhoeven R.; Hiller M.; Tadokoro S. (1998), Workspace, stiffness, singularities and classification of tendon-driven stewart platforms, In *Advances in Robot Kinematics Analysis and Control*, Lenarcic J. and Husty L., pp. 105-114, Kluwer Academic, Netherlands

Verhoeven R.; Hiller M. (2000), Estimating the controllable workspace of tendon-based Stewart platforms, In *Advances in Robot Kinematics*, Lenarcic J. and Stanisic M., pp. 277-284, Kluwer Academic, Netherlands

Yamaguchi F., (2002), *A Totally Four-dimensional Approach / Computer-Aided Geometric Design*, Springer-Verlag, Tokyo

A Complete Family of Kinematically-Simple Joint Layouts: Layout Models, Associated Displacement Problem Solutions and Applications

Scott Nokleby and Ron Podhorodeski

1. Introduction

Podhorodeski and Pittens (1992, 1994) and Podhorodeski (1992) defined a kinematically-simple (KS) layout as a manipulator layout that incorporates a spherical group of joints at the wrist with a main-arm comprised of successfully parallel or perpendicular joints with no unnecessary offsets or link lengths between joints. Having a spherical group of joints within the layouts ensures, as demonstrated by Pieper (1968), that a closed-form solution for the inverse displacement problem exists.

Using the notation of possible joint axes directions shown in Figure 1 and arguments of kinematic equivalency and mobility of the layouts, Podhorodeski and Pittens (1992, 1994) showed that there are only five unique, revolute-only, main-arm joint layouts representative of all layouts belonging to the KS family. These layouts have joint directions CBE, CAE, BCE, BEF, and AEF and are denoted KS 1 to 5 in Figure 2.
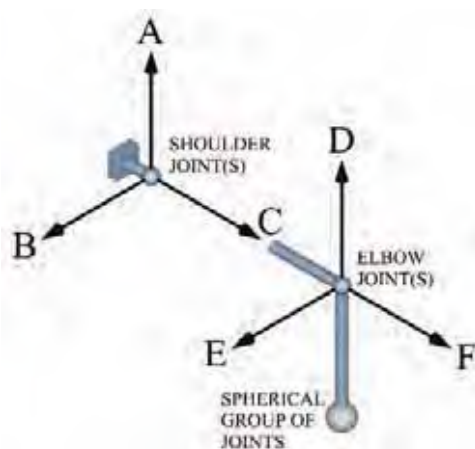
Figure 1. Possible Joint Directions for the KS Family of Layouts
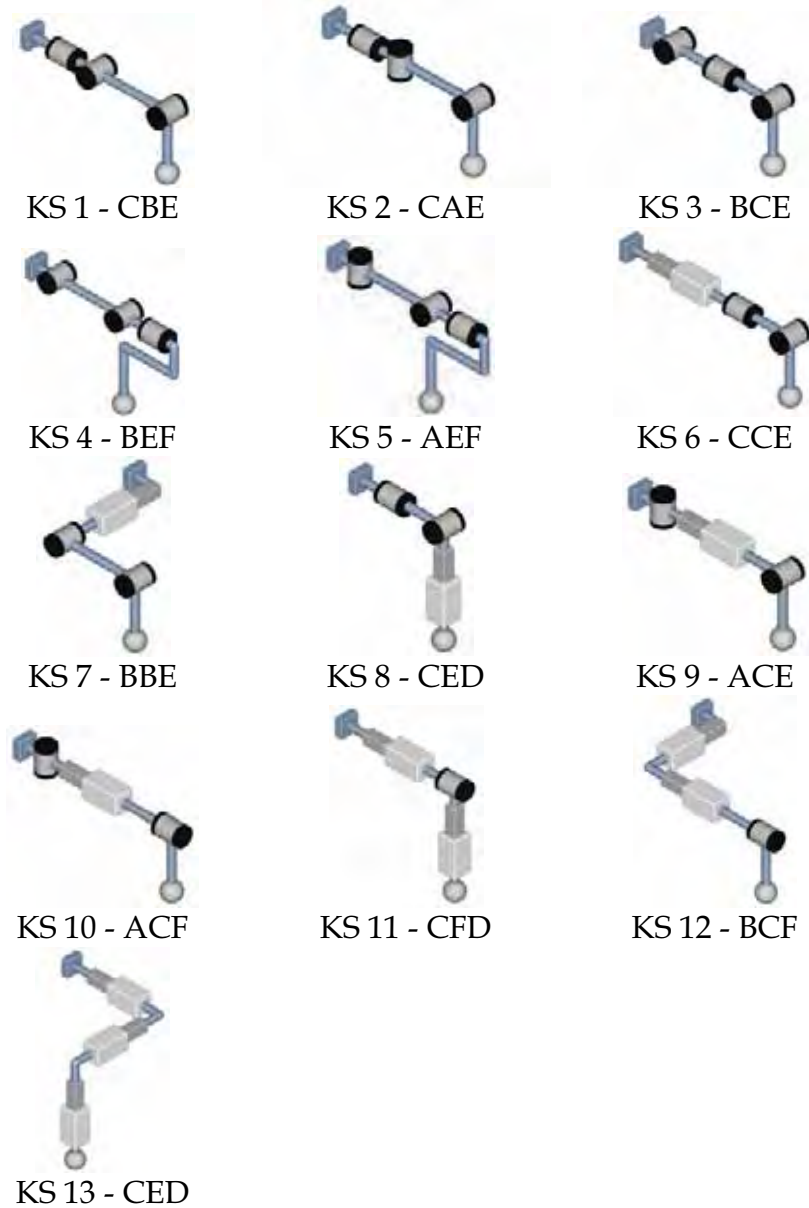
Figure 2. KS Family of Joint Layouts

Podhorodeski (1992) extended the work of Podhorodeski and Pittens (1992, 1994) to include prismatic joints in the layouts. Podhorodeski (1992) concluded that there are 17 layouts belonging to the KS family: five layouts comprised of three revolute joints; nine layouts comprised of two revolute joints and one prismatic joint; two layouts comprised of one revolute joint and two prismatic joints; and one layout comprised of three prismatic joints. However, four of the layouts comprised of two revolute joints and one prismatic joint (layouts he denotes AAE, AAF, ABF, and BAE) are not kinematically simple, by the definition set out in this chapter, due to an unnecessary offset existing between the second and third joints.

Yang et al. (2001) used the concepts developed by Podhorodeski and Pittens (1992, 1994) to attempt to generate all unique KS layouts comprised of two revolute joints and one prismatic joint. The authors identified eight layouts. Of these eight layouts, five layouts (the layouts they denote CAE, CAF, CBF, CFE, and CCE) are not kinematically simple, as defined in this chapter, in that they incorporate unnecessary offsets and one layout (the layout they denote CBE) is not capable of spatial motion.

The purpose of this chapter is to clarify which joint layouts comprised of a combination of revolute and/or prismatic joints belong to the KS family. The chapter first identifies all layouts belonging to the KS family. Zero-displacement diagrams and Denavit and Hartenberg (D&H) parameters (1955) used to model the layouts are presented. The complete forward and inverse displacement solutions for the KS family of layouts are shown. The application of the KS family of joint layouts and the application of the presented forward and inverse displacement solutions to both serial and parallel manipulators is discussed.

2. The Kinematically-Simple Family of Joint Layouts

The possible layouts can be divided into four groups: layouts with three revolute joints; layouts with two revolute joints and one prismatic joint; layouts with one revolute joint and two prismatic joints; and layouts with three prismatic joints.

2.1 Layouts with Three Revolute Joints

Using arguments of kinematic equivalency and motion capability, Podhorodeski and Pittens (1992, 1994) identified five unique KS layouts representative of all layouts comprised of three revolute joints. Referring to Figure 1, the joint directions for these layouts can be represented by the axes directions CBE, CAE, BCE, BEF, and AEF, and are illustrated as KS 1 to 5 in Figure 2, respectively.

Fundamentally degenerate layouts occur when either the three axes of the main arm intersect to form a spherical group (see Figure 3a) or when the axis of the final revolute joint intersects the spherical group at the wrist (see Figure 3b), i.e., the axis of the third joint is in the D direction of Figure 1. Note that for any KS layout, if the third joint is a revolute joint, the axis of the joint cannot intersect the spherical group at the wrist or the layout will be incapable of fully spatial motion.

(a) Layout CBF   (b) Layout CBD
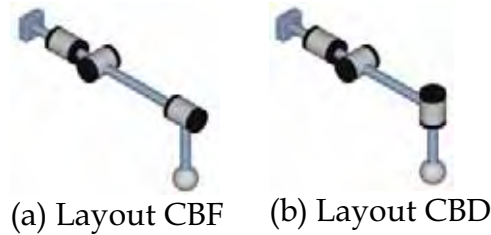
Figure 3. Examples of the Two Types of Degenerate Revolute-Revolute-Revolute Layouts

2.2 Layouts with Two Revolute Joints and One Prismatic Joint

Layouts consisting of two revolute joints and one prismatic joint can take on three forms: prismatic-revolute-revolute; revolute-revolute-prismatic; and revolute-prismatic-revolute.

2.2.1 Prismatic-Revolute-Revolute Layouts

For a prismatic-revolute-revolute layout to belong to the KS family, either the two revolute joints will be perpendicular to one another or the two revolute joints will be parallel to one another. If the two revolute joints are perpendicular to one another, then the two axes must intersect to form a pointer, otherwise an unnecessary offset would exist between the two joints and the layout would not be kinematically simple. The prismatic-pointer layout can be represented by the axes directions CCE and is illustrated as KS 6 in Figure 2.
For the case where the two revolute joints are parallel to one another, in order to achieve full spatial motion, the axes of the revolute joints must also be parallel to the axis of the prismatic joint. If the axes of the revolute joints were perpendicular to the axis of the prismatic joint, the main-arm's ability to move the centre of the spherical group would be restricted to motion in a plane, i.e., fundamentally degenerate. In addition, a necessary link length must exist between the two revolute joints. The axes for this layout can be represented with the directions BBE and the layout is illustrated as KS 7 in Figure 2.

2.2.2 Revolute-Revolute-Prismatic Layouts

For a revolute-revolute-prismatic layout to belong to the KS family, either the two revolute joints will be perpendicular to one another or the two revolute joints will be parallel to one another. If the two revolute joints are perpendicular to one another, then the two axes must intersect to form a pointer, otherwise an unnecessary offset would exist between the two joints and the layout would not be kinematically simple. The pointer-prismatic layout can be represented by the axes directions CED and is illustrated as KS 8 in Figure 2.

For the case where the two revolute joints are parallel to one another, the axes of the revolute joints must also be parallel to the axis of the prismatic joint. In addition, a necessary link length must exist between the two revolute joints. The axes for this layout can be represented with the directions ADD. Note that for this configuration, the layout is fundamentally degenerate, unless an additional link length is added between joints two and three, since without the additional link length, the axis of the second revolute joint would always pass through the centre of the spherical joint group (see Figure 4a). Figure 4b illustrates the non-degenerate KS layout with an additional link length between the second revolute joint and the prismatic joint. However, the layout of Figure 4b is kinematically equivalent to KS 7 and therefore is not counted as a unique KS layout.

(a) Layout ADD   (b) Layout ADD with Offset
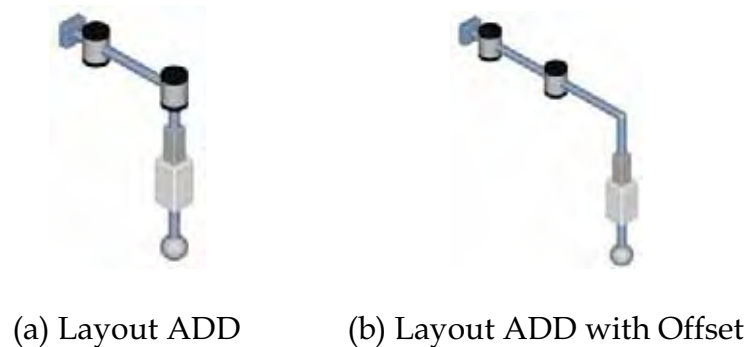
Figure 4. Revolute-Revolute-Prismatic Layouts: a) Degenerate; b) Non-Degenerate

2.2.3 Revolute-Prismatic-Revolute Layouts

For a revolute-prismatic-revolute layout, in order to achieve spatial motion and belong to the KS class, the axes of the two revolute joints must be orthogonal to one another. The resulting KS layouts of axes can be represented by the axes directions ACE and ACF and are illustrated as KS 9 and KS 10 in Figure 2, respectively.

2.3 Layouts with One Revolute Joint and Two Prismatic Joints

Layouts consisting of one revolute joint and two prismatic joints can take on three forms: prismatic-revolute-prismatic; prismatic-prismatic-revolute; and revolute-prismatic-prismatic.

2.3.1 Prismatic-Revolute-Prismatic Layouts

For a prismatic-revolute-prismatic layout, the two prismatic joints must be perpendicular to each other. In order to achieve spatial motion and be kine-

matically simple, the axis of the revolute joint must be parallel to the axis of one of the prismatic joints. The feasible layout of joint directions can be represented by the axes directions CFD and is illustrated as KS 11 in Figure 2.

2.3.2 Prismatic-Prismatic-Revolute Layouts

For a prismatic-prismatic-revolute layout, the two prismatic joints must be perpendicular to each other. In order to achieve spatial motion and be kinematically simple, the axis of the revolute joint must be parallel to one of the prismatic joints. The feasible layout of joint directions can be represented by the axes directions BCF and is illustrated as KS 12 in Figure 2.

2.3.3 Revolute-Prismatic-Prismatic Layouts

For a revolute-prismatic-prismatic layout, the two prismatic joints must be perpendicular to each other. In order to achieve spatial motion and be kinematically simple, the axis of the revolute joint must be parallel to the axis of one of the prismatic joints. The feasible layout of joint directions can be represented by the axes directions CCD. Note that this layout is kinematically equivalent to the prismatic-revolute-prismatic KS 11. Therefore, the revolute-prismatic-prismatic layout is not kinematically unique. For a further discussion on collinear revolute-prismatic axes please see Section 2.5.

2.4 Layouts with Three Prismatic Joints

To achieve spatial motion with three prismatic joints and belong to the KS class, the joint directions must be mutually orthogonal. A representative layout of joint directions is CED. This layout is illustrated as KS 13 in Figure 2.

2.5 Additional Kinematically-Simple Layouts

The layouts above represent the 13 layouts with unique kinematics belonging to the KS family. However, additional layouts that have unique joint structures can provide motion that is kinematically equivalent to one of the KS layouts. For branches where the axes of a prismatic and revolute joint are collinear, there are two possible layouts to achieve the same motion. Four layouts, KS 6, 7, 11, and 12, have a prismatic joint followed by a collinear revolute joint. The order of these joints could be reversed, i.e., the revolute joint could come first followed by the prismatic joint. The order of the joints has no bearing on the kinematics of the layout, but would be very relevant in the physical design of a manipulator. Note that the d_j and θ_j elements of the corresponding rows in the D&H tables (see Section 3.2) would need to be interchanged along with

an appropriate change in subscripts. The presented forward and inverse displacement solutions in Sections 4.1 and 4.2 would remain unchanged except for a change in the relevant subscripts.

In addition to the above four layouts, as discussed in Section 2.2.2, the layout shown in Figure 4b is kinematically equivalent to KS 7. Therefore, there are five additional kinematically-simple layouts that can be considered part of the KS family.

3. Zero-Displacement Diagrams and D&H Parameters

3.1 Zero-Displacement Diagrams

The zero-displacement diagrams ($\theta_i = 0$, for all revolute joints i) for the KS family of layouts for Craig's (1989) convention of frame assignment are presented in Figures 5 to 7. Note that the KS layouts in Figure 2 are not necessarily shown in zero-displacement. The rotations necessary to put each of the KS Layouts from zero-displacement configuration into the configuration illustrated in Figure 2 are outlined in Table 1.

KS	Rotations	KS	Rotations
1	$\theta_2 = \dfrac{\pi}{2}$	2	$\theta_2 = \dfrac{\pi}{2}$
3	$\theta_3 = \dfrac{\pi}{2}$	4	$\theta_2 = \dfrac{\pi}{2}$ & $\theta_3 = \dfrac{\pi}{2}$
5	$\theta_2 = -\dfrac{\pi}{2}$ & $\theta_3 = \dfrac{\pi}{2}$	6	$\theta_3 = -\dfrac{\pi}{2}$
7	None	8	$\theta_2 = -\dfrac{\pi}{2}$
9	$\theta_3 = \dfrac{\pi}{2}$	10	None
11	None	12	$\theta_3 = \dfrac{\pi}{2}$
13	None		

Table 1. Rotations Necessary to put Each of the KS Layouts from Zero-Displacement Configuration (Figures 5 to 7) into the Configuration Illustrated in Figure 2

3.2 D&H Parameters

Table 2 shows the D&H parameters for the kinematically-simple family of joint layouts. The D&H parameters follow Craig's frame assignment convention (Craig, 1989) and correspond to the link transformations:

$$
{}^{j-1}_{j}\mathbf{T} = Rot_{\hat{\mathbf{x}}_{j-1}}\left(\alpha_{j-1}\right) Trans_{\hat{\mathbf{x}}_{j-1}}\left(a_{j-1}\right) Trans_{\hat{\mathbf{z}}_{j}}\left(d_{j}\right) Rot_{\hat{\mathbf{z}}_{j}}\left(\theta_{j}\right)
$$

(1)

$$
= \begin{bmatrix}
\cos(\theta_j) & -\sin(\theta_j) & 0 & a_{j-1} \\
\sin(\theta_j)\cos(\alpha_{j-1}) & \cos(\theta_j)\cos(\alpha_{j-1}) & -\sin(\alpha_{j-1}) & -\sin(\alpha_{j-1})d_j \\
\sin(\theta_j)\sin(\alpha_{j-1}) & \cos(\theta_j)\sin(\alpha_{j-1}) & \cos(\alpha_{j-1}) & \cos(\alpha_{j-1})d_j \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

where ${}^{j-1}_{j}\mathbf{T}$ is a homogeneous transformation describing the location and orientation of link-frame F_j with respect to link-frame F_{j-1}, $Rot_{\hat{\mathbf{x}}_{j-1}}\left(\alpha_{j-1}\right)$ denotes a rotation about the $\hat{\mathbf{x}}_{j-1}$ axis by α_{j-1}, $Trans_{\hat{\mathbf{x}}_{j-1}}\left(a_{j-1}\right)$ denotes a translation along the $\hat{\mathbf{x}}_{j-1}$ axis by a_{j-1}, $Trans_{\hat{\mathbf{z}}_{j}}\left(d_{j}\right)$ denotes a translation along the $\hat{\mathbf{z}}_{j}$ axis by d_{j}, and $Rot_{\hat{\mathbf{z}}_{j}}\left(\theta_{j}\right)$ denotes a rotation about the $\hat{\mathbf{z}}_{j}$ axis by θ_{j}.

The homogeneous transformation of equation (1) is of the form:

$$
{}^{j-1}_{j}\mathbf{T} = \begin{bmatrix} {}^{j-1}_{j}\mathbf{R} & {}^{j-1}\mathbf{p}_{o_{j-1}\rightarrow o_j} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}
$$

(2)

where ${}^{j-1}_{j}\mathbf{R}$ is a 3x3 orthogonal rotation matrix describing the orientation of frame F_j with respect to frame F_{j-1} and ${}^{j-1}\mathbf{p}_{o_{j-1}\rightarrow o_j}$ is a vector from the origin of frame F_{j-1} to the origin of frame F_j.
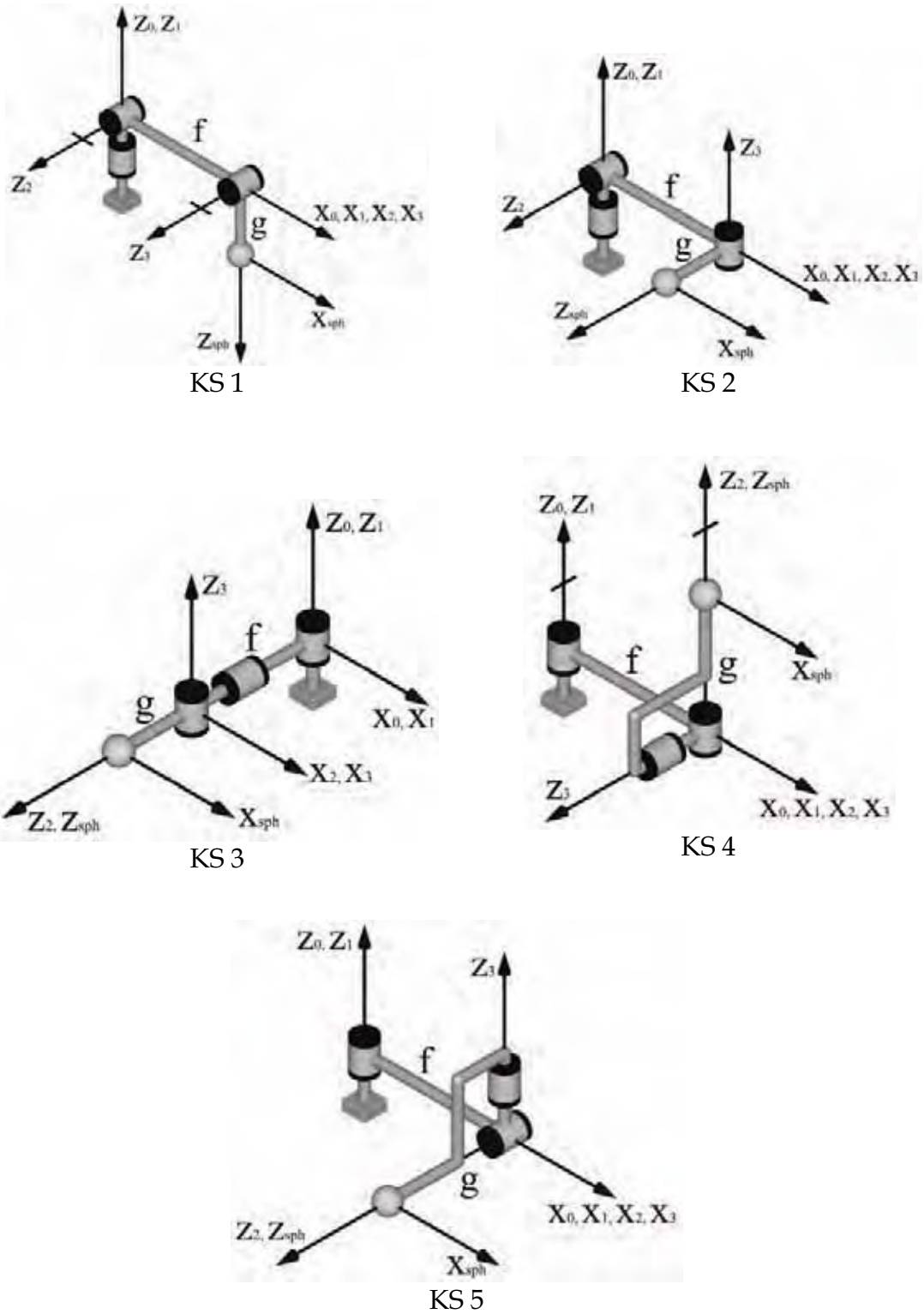
Figure 5. Zero-Displacement Diagrams for Layouts with Three Revolute Joints (KS 1 to 5)
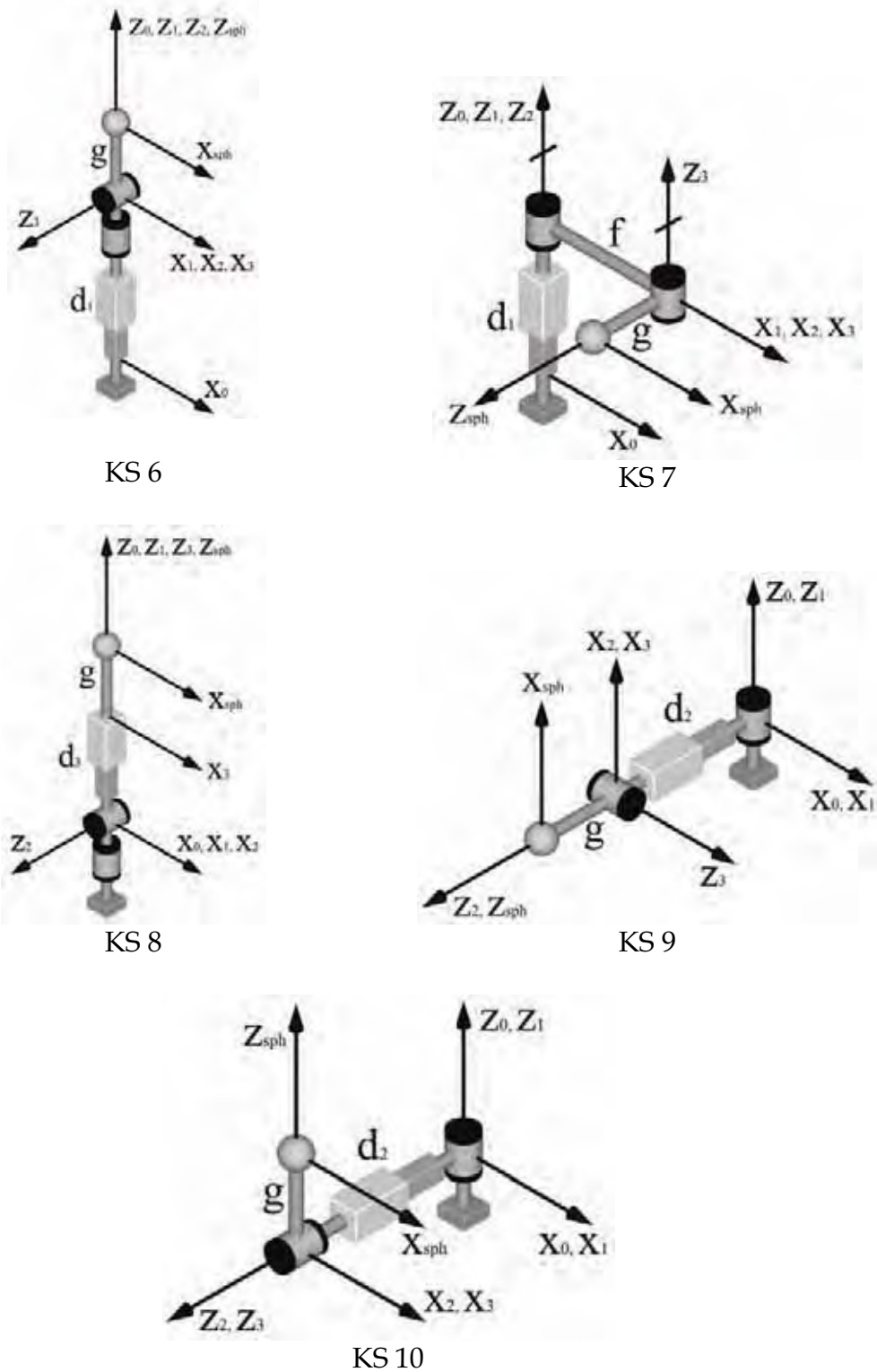
Figure 6. Zero-Displacement Diagrams for Layouts with Two Revolute Joints and One Prismatic Joint (KS 6 to 10)

KS 11   KS 12
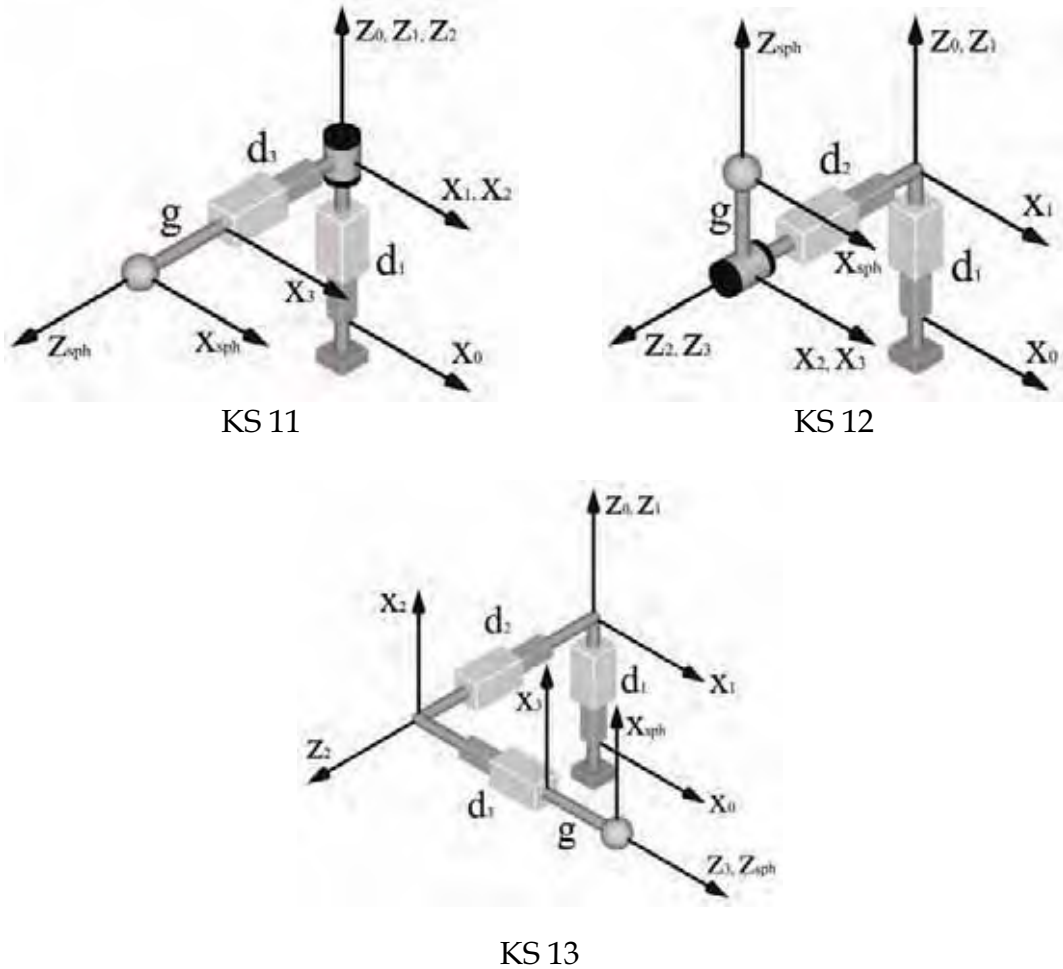
KS 13

Figure 7. Zero-Displacement Diagrams for Layouts with One Revolute Joint and Two Primatic Joints (KS 11 and 12) or Three Prismatic Joints (KS 13)

4. Forward and Inverse Displacement Solutions

4.1 Forward Displacement Solutions for the KS Family of Layouts

The position and orientation of the spherical wrist frame F_{sph} with respect to the base frame F_0 is found from:

$$
{}^0_{sph}\mathbf{T} = {}^0_1\mathbf{T}\ {}^1_2\mathbf{T}\ {}^2_3\mathbf{T}\ {}^3_{sph}\mathbf{T} = \begin{bmatrix} {}^0_{sph}\mathbf{R} & {}^0\mathbf{p}_{o_0 \to o_{sph}} \\ 0\quad 0\quad 0 & 1 \end{bmatrix} \tag{3}
$$

where the homogeneous transformation ${}^{j-1}_{j}\mathbf{T}$ is defined in equation (1). The transformation ${}^0_{sph}\mathbf{T}$ is the solution to the forward displacement problem: ${}^0_{sph}\mathbf{R}$

is the change in orientation due to the first three joints and ${}^{0}\mathbf{p}_{o_0 \to o_{sph}}$ is the location of the spherical wrist centre. The homogeneous transformations ${}^{0}_{sph}\mathbf{T}$ for the KS family of layouts can be found in Tables 3 and 4. Note that in Tables 3 and 4, c_i and s_i denote $\cos(\theta_i)$ and $\sin(\theta_i)$, respectively.

KS	F_{j-1}	α_{j-1}	a_{j-1}	d_j	θ_j	F_j	KS	F_{j-1}	α_{j-1}	a_{j-1}	d_j	θ_j	F_j
1	F_0	0	0	0	θ_1	F_1	2	F_0	0	0	0	θ_1	F_1
	F_1	$\pi/2$	0	0	θ_2	F_2		F_1	$\pi/2$	0	0	θ_2	F_2
	F_2	0	f	0	θ_3	F_3		F_2	$-\pi/2$	f	0	θ_3	F_3
	F_3	$\pi/2$	0	g	0	F_{sph}		F_3	$\pi/2$	0	g	0	F_{sph}
3	F_0	0	0	0	θ_1	F_1	4	F_0	0	0	0	θ_1	F_1
	F_1	$\pi/2$	0	f	θ_2	F_2		F_1	0	f	0	θ_2	F_2
	F_2	$-\pi/2$	0	0	θ_3	F_3		F_2	$\pi/2$	0	0	θ_3	F_3
	F_3	$\pi/2$	0	g	0	F_{sph}		F_3	$-\pi/2$	0	g	0	F_{sph}
5	F_0	0	0	0	θ_1	F_1	6	F_0	0	0	d_1	0	F_1
	F_1	$\pi/2$	f	0	θ_2	F_2		F_1	0	0	0	θ_2	F_2
	F_2	$-\pi/2$	0	0	θ_3	F_3		F_2	$\pi/2$	0	0	θ_3	F_3
	F_3	$\pi/2$	0	g	0	F_{sph}		F_3	$-\pi/2$	0	g	0	F_{sph}
7	F_0	0	0	d_1	0	F_1	8	F_0	0	0	0	θ_1	F_1
	F_1	0	0	0	θ_2	F_2		F_1	$\pi/2$	0	0	θ_2	F_2
	F_2	0	f	0	θ_3	F_3		F_2	$-\pi/2$	0	d_3	0	F_3
	F_3	$\pi/2$	0	g	0	F_{sph}		F_3	0	0	g	0	F_{sph}
9	F_0	0	0	0	θ_1	F_1	10	F_0	0	0	0	θ_1	F_1
	F_1	$\pi/2$	0	d_2	$\pi/2$	F_2		F_1	$\pi/2$	0	d_2	0	F_2
	F_2	$\pi/2$	0	0	θ_3	F_3		F_2	0	0	0	θ_3	F_3
	F_3	$-\pi/2$	0	g	0	F_{sph}		F_3	$-\pi/2$	0	g	0	F_{sph}
11	F_0	0	0	d_1	0	F_1	12	F_0	0	0	d_1	0	F_1
	F_1	0	0	0	θ_2	F_2		F_1	$\pi/2$	0	d_2	0	F_2
	F_2	$\pi/2$	0	d_3	0	F_3		F_2	0	0	0	θ_3	F_3
	F_3	0	0	g	0	F_{sph}		F_3	$-\pi/2$	0	g	0	F_{sph}
13	F_0	0	0	d_1	0	F_1							
	F_1	$\pi/2$	0	d_2	$\pi/2$	F_2							
	F_2	$\pi/2$	0	d_3	0	F_3							
	F_3	0	0	g	0	F_{sph}							

Table 2. D&H Parameters for the KS Layouts

KS	$^0_{sph}\mathbf{T}$
1	$$\begin{bmatrix} c_1 c_{23} & s_1 & c_1 s_{23} & c_1(c_2 f + s_{23} g) \\ s_1 c_{23} & -c_1 & s_1 s_{23} & s_1(c_2 f + s_{23} g) \\ s_{23} & 0 & -c_{23} & s_2 f - c_{23} g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
2	$$\begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 & c_1 c_2 s_3 + s_1 c_3 & c_1 c_2 f + (c_1 c_2 s_3 + s_1 c_3)g \\ s_1 c_2 c_3 + c_1 s_3 & -s_1 s_2 & s_1 c_2 s_3 - c_1 c_3 & s_1 c_2 f + (s_1 c_2 s_3 - c_1 c_3)g \\ s_2 c_3 & c_2 & s_2 s_3 & s_2 f + s_2 s_3 g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
3	$$\begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 & c_1 c_2 s_3 + s_1 c_3 & s_1 f + (c_1 c_2 s_3 + s_1 c_3)g \\ s_1 c_2 c_3 + c_1 s_3 & -s_1 s_2 & s_1 c_2 s_3 - c_1 c_3 & -c_1 f + (s_1 c_2 s_3 - c_1 c_3)g \\ s_2 c_3 & c_2 & s_2 s_3 & s_2 s_3 g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
4	$$\begin{bmatrix} c_{12} c_3 & -s_{12} & -c_{12} s_3 & c_1 f - c_{12} s_3 g \\ s_{12} c_3 & c_{12} & -s_{12} s_3 & s_1 f - s_{12} s_3 g \\ s_3 & 0 & c_3 & c_3 g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
5	$$\begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 & c_1 c_2 s_3 + s_1 c_3 & c_1 f + (c_1 c_2 s_3 + s_1 c_3)g \\ s_1 c_2 c_3 + c_1 s_3 & -s_1 s_2 & s_1 c_2 s_3 - c_1 c_3 & s_1 f + (s_1 c_2 s_3 - c_1 c_3)g \\ s_2 c_3 & c_2 & s_2 s_3 & s_2 s_3 g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Table 3. Forward Displacement Solutions for KS 1 to 5

4.2 Inverse Displacement Solutions for the KS Family of Layouts

For the inverse displacement solution, the location of the spherical wrist centre with respect to the base, $^0\mathbf{p}_{o_0 \to o_{sph}}$, is known:

$$^0\mathbf{p}_{o_0 \to o_{sph}} = \begin{Bmatrix} p_x \\ p_y \\ p_z \end{Bmatrix} \tag{4}$$

Paul (1981) presented a methodology to solve the inverse displacement problem of 6-joint manipulators with a spherical wrist. To demonstrate the application of this methodology to the inverse displacement problem for the KS family, KS 1 will be used as an example.

KS	$_{sph}^{0}\mathbf{T}$	KS	$_{sph}^{0}\mathbf{T}$
6	$\begin{bmatrix} c_2c_3 & -s_2 & -c_2s_3 & -c_2s_3g \\ s_2c_3 & c_2 & -s_2s_3 & -s_2s_3g \\ s_3 & 0 & c_3 & c_3g+d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	7	$\begin{bmatrix} c_{23} & 0 & s_{23} & c_2f+s_{23}g \\ s_{23} & 0 & -c_{23} & s_2f-c_{23}g \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
8	$\begin{bmatrix} c_1c_2 & -s_1 & -c_1s_2 & -c_1s_2(d_3+g) \\ s_1c_2 & c_1 & -s_1s_2 & -s_1s_2(d_3+g) \\ s_2 & 0 & c_2 & c_2(d_3+g) \\ 0 & 0 & 0 & 1 \end{bmatrix}$	9	$\begin{bmatrix} s_1s_3 & -c_1 & s_1c_3 & s_1(c_3g+d_2) \\ -c_1s_3 & -s_1 & -c_1c_3 & -c_1(c_3g+d_2) \\ c_3 & 0 & -s_3 & -s_3g \\ 0 & 0 & 0 & 1 \end{bmatrix}$
10	$\begin{bmatrix} c_1c_3 & -s_1 & -c_1s_3 & -c_1s_3g+s_1d_2 \\ s_1c_3 & c_1 & -s_1s_3 & -s_1s_3g-c_1d_2 \\ s_3 & 0 & c_3 & c_3g \\ 0 & 0 & 0 & 1 \end{bmatrix}$	11	$\begin{bmatrix} c_2 & 0 & s_2 & s_2(g+d_3) \\ s_2 & 0 & -c_2 & -c_2(g+d_3) \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
12	$\begin{bmatrix} c_3 & 0 & -s_3 & -s_3g \\ 0 & 1 & 0 & -d_2 \\ s_3 & 0 & c_3 & c_3g+d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	13	$\begin{bmatrix} 0 & 0 & 1 & g+d_3 \\ 0 & -1 & 0 & -d_2 \\ 1 & 0 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table 4. Forward Displacement Solutions for KS 6 to 13

From the forward displacement solution presented in Table 3 for KS 1, the following relation exists:

$$\begin{Bmatrix} c_1(c_2f+s_{23}g) \\ s_1(c_2f+s_{23}g) \\ s_2f-c_{23}g \\ 1 \end{Bmatrix} = \begin{Bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{Bmatrix} \tag{5}$$

where the left-hand-side of equation (5) is the last column of $_{sph}^{0}\mathbf{T}$. Pre-multiplying both sides of equation (5) by $_{0}^{1}\mathbf{T} = \left(_{1}^{0}\mathbf{T}\right)^{-1}$ isolates θ_1 to the right-hand-side of the expression:

$$\begin{Bmatrix} c_2f+s_{23}g \\ 0 \\ s_2f-c_{23}g \\ 1 \end{Bmatrix} = \begin{Bmatrix} c_1p_x+s_1p_y \\ -s_1p_x+c_1p_y \\ p_z \\ 1 \end{Bmatrix} \tag{6}$$

From the second row of equation (6), a solution for θ_1 can be found as:

$$\theta_1 = \text{atan2}(p_y, p_x) \text{ or } \theta_1 = \text{atan2}(-p_y, -p_x) \tag{7}$$

where atan2 denotes a quadrant corrected arctangent function (Paul, 1981). Squaring and adding the first three rows of equation (6) allows an expression for s_3 to be found thus yielding a solution for θ_3 of:

$$\theta_3 = \text{atan2}\left(s_3, \pm\sqrt{1-s_3^2}\right), \text{ where } s_3 = \frac{p_x^2 + p_y^2 + p_z^2 - f^2 - g^2}{2fg} \tag{8}$$

From rows one and three of equation (6), after substituting $c_{23} = c_2 c_3 - s_2 s_3$ and $s_{23} = s_2 c_3 + c_2 s_3$, expressions for s_2 and c_2 can be found thus yielding a solution for θ_2 of:

$$\theta_2 = \text{atan2}(s_2, c_2) \tag{9}$$

where

$$s_2 = \frac{c_3 g\left(c_1 p_x + s_1 p_y\right) + \left(f + s_3 g\right) p_z}{f^2 + g^2 + 2s_3 fg}$$

$$c_2 = \frac{\left(f + s_3 g\right)\left(c_1 p_x + s_1 p_y\right) - c_3 g p_z}{f^2 + g^2 + 2s_3 fg}$$

A similar procedure can be followed for the other KS layouts. Inverse displacement solutions for all 13 of the KS layouts are summarized in Tables 5 and 6.

KS	Inverse Displacement Solutions
1	$\theta_1 = \mathrm{atan2}(p_y, p_x)$ or $\mathrm{atan2}(-p_y, -p_x)$ $\theta_3 = \mathrm{atan2}\left(s_3, \pm\sqrt{1-s_3^2}\right)$, where $s_3 = \dfrac{p_x^2 + p_y^2 + p_z^2 - f^2 - g^2}{2fg}$ $\theta_2 = \mathrm{atan2}(s_2, c_2)$, where $s_2 = \dfrac{c_3 g(c_1 p_x + s_1 p_y) + (f + s_3 g)p_z}{f^2 + g^2 + 2s_3 fg}$ & $c_2 = \dfrac{(f + s_3 g)(c_1 p_x + s_1 p_y) - c_3 g p_z}{f^2 + g^2 + 2s_3 fg}$
2	$\theta_3 = \mathrm{atan2}\left(s_3, \pm\sqrt{1-s_3^2}\right)$, where $s_3 = \dfrac{p_x^2 + p_y^2 + p_z^2 - f^2 - g^2}{2fg}$ $\theta_2 = \mathrm{atan2}\left(s_2, \pm\sqrt{1-s_2^2}\right)$, where $s_2 = \dfrac{p_z}{f + s_3 g}$ $\theta_1 = \mathrm{atan2}(s_1, c_1)$, where $s_1 = \dfrac{(f + s_3 g)c_2 p_y + c_3 g p_x}{p_x^2 + p_y^2}$ & $c_1 = \dfrac{(f + s_3 g)c_2 p_x - c_3 g p_y}{p_x^2 + p_y^2}$
3	$\theta_3 = \mathrm{atan2}\left(\pm\sqrt{1-c_3^2}, c_3\right)$, where $c_3 = \dfrac{p_x^2 + p_y^2 + p_z^2 - f^2 - g^2}{2fg}$ $\theta_2 = \mathrm{atan2}\left(s_2, \pm\sqrt{1-s_2^2}\right)$, where $s_2 = \dfrac{p_z}{s_3 g}$ $\theta_1 = \mathrm{atan2}(s_1, c_1)$, where $s_1 = \dfrac{(f + c_3 g)p_x + c_2 s_3 g p_y}{p_x^2 + p_y^2}$ & $c_1 = \dfrac{c_2 s_3 g p_x - (f + c_3 g)p_y}{p_x^2 + p_y^2}$
4	$\theta_3 = \mathrm{atan2}\left(\pm\sqrt{1-c_3^2}, c_3\right)$, where $c_3 = p_z/g$ $\theta_2 = \mathrm{atan2}\left(\pm\sqrt{1-c_2^2}, c_2\right)$, where $c_2 = \dfrac{f^2 + s_3^2 g^2 - p_x^2 - p_y^2}{2s_3 fg}$ $\theta_1 = \mathrm{atan2}(s_1, c_1)$, where $s_1 = \dfrac{s_2 s_3 g p_x + (f - c_2 s_3 g)p_y}{p_x^2 + p_y^2}$ & $c_1 = \dfrac{(f - c_2 s_3 g)p_x - s_2 s_3 g p_y}{p_x^2 + p_y^2}$
5	$\theta_1 = \mathrm{atan2}(p_y, p_x) \pm \mathrm{atan2}\left(\sqrt{p_x^2 + p_y^2 - (f + c_2 s_3 g)^2}, \, f + c_2 s_3 g\right)$, where $c_2 s_3 = \dfrac{p_x^2 + p_y^2 + p_z^2 - f^2 - g^2}{2fg}$ $\theta_3 = \mathrm{atan2}\left(\pm\sqrt{1-c_3^2}, c_3\right)$, where $c_3 = (s_1 p_x - c_1 p_y)/g$ $\theta_2 = \mathrm{atan2}(s_2, c_2)$, where $s_2 = \dfrac{p_z}{s_3 g}$ & $c_2 = \dfrac{c_1 p_x + s_1 p_y - f}{s_3 g}$

Table 5. Inverse Displacement Solutions for KS 1 to 5

KS	Inverse Displacement Solutions
6	$d_1 = p_z \pm \sqrt{-p_x^2 - p_y^2 + g^2}$ $\theta_3 = \text{atan2}\left(\pm\sqrt{1-c_3^2}, c_3\right)$, where $c_3 = (p_z - d_1)/g$ $\theta_2 = \text{atan2}(s_2, c_2)$, where $s_2 = \dfrac{-p_y}{s_3 g}$ & $c_2 = \dfrac{-p_x}{s_3 g}$
7	$d_1 = p_z$ $\theta_3 = \text{atan2}\left(s_3, \pm\sqrt{1-s_3^2}\right)$, where $s_3 = \dfrac{p_x^2 + p_y^2 - f^2 - g^2}{2fg}$ $\theta_2 = \text{atan2}(s_2, c_2)$, where $s_2 = \dfrac{c_3 g p_x + (f + s_3 g)p_y}{(f + s_3 g)^2 + c_3^2 g^2}$ & $c_2 = \dfrac{(f + s_3 g)p_x - c_3 g p_y}{(f + s_3 g)^2 + c_3^2 g^2}$
8	$\theta_1 = \text{atan2}(p_y, p_x)$ or $\text{atan2}(-p_y, -p_x)$ $d_3 = \pm\sqrt{p_x^2 + p_y^2 + p_z^2} - g$ $\theta_2 = \text{atan2}(s_2, c_2)$, where $s_2 = \dfrac{-c_1 p_x - s_1 p_y}{d_3 + g}$ & $c_2 = \dfrac{p_z}{d_3 + g}$
9	$\theta_1 = \text{atan2}(p_x, -p_y)$ or $\text{atan2}(-p_x, p_y)$ $d_2 = s_1 p_x - c_1 p_y \pm \sqrt{g^2 - p_z^2}$ $\theta_3 = \text{atan2}(s_3, c_3)$, where $s_3 = -p_z/g$ & $c_3 = (s_1 p_x - c_1 p_y - d_2)/g$
10	$\theta_3 = \text{atan2}\left(\pm\sqrt{1-c_3^2}, c_3\right)$, where $c_3 = p_z/g$ $d_2 = \pm\sqrt{p_x^2 + p_y^2 - s_3^2 g^2}$ $\theta_1 = \text{atan2}(s_1, c_1)$, where $s_1 = \dfrac{-s_3 g p_y + d_2 p_x}{p_x^2 + p_y^2}$ & $c_1 = \dfrac{-s_3 g p_x - d_2 p_y}{p_x^2 + p_y^2}$
11	$d_1 = p_z$ $d_3 = \pm\sqrt{p_x^2 + p_y^2} - g$ $\theta_2 = \text{atan2}(s_2, c_2)$, where $s_2 = \dfrac{p_x}{d_3 + g}$ & $c_2 = \dfrac{-p_y}{d_3 + g}$
12	$d_2 = -p_y$ $d_1 = p_z \pm \sqrt{-p_x^2 + g^2}$ $\theta_3 = \text{atan2}(s_3, c_3)$, where $s_3 = -p_x/g$ & $c_3 = (p_z - d_1)/g$
13	$d_3 = p_x - g$ $d_2 = -p_y$ $d_1 = p_z$

Table 6. Inverse Displacement Solutions for KS 6 to 13

Referring to Tables 5 and 6, for KS 1 to 6, 8, 9, and 10, up to four possible solutions exist to the inverse displacement problem. For KS 7, 11, and 12, up to two possible solutions exist for the inverse displacement problem. For KS 13 there is only one solution to the inverse displacement problem.

For the inverse displacement solutions presented, undefined configurations occur when the spherical wrist centre of the arm intersects either the first or second joint axes, provided the axes are for a revolute joint. In such a configuration, the inverse solution becomes undefined, i.e., an infinity of possible solutions exist. Looking at KS 3 of Figure 5 as an example, if $s_3 = 0$ as illustrated, the spherical wrist centre intersects the second joint axis and the solution for θ_2 becomes arbitrary. Similarly, if $p_x = p_y = 0$, the spherical wrist centre intersects the first joint axis and the solution for θ_1 becomes arbitrary.

Table 7 reports all of the undefined configurations for the KS family of layouts. If an undefined configuration was encountered, a value would be assigned to the arbitrary joint displacement.

KS	Undefined Configurations	KS	Undefined Configurations
1	$p_x = p_y = 0 \Rightarrow \theta_1$ is arbitrary $f^2 + g^2 + 2s_3 fg = 0 \Rightarrow \theta_2$ is arbitrary	2	$p_x = p_y = 0 \Rightarrow \theta_1$ is arbitrary $f + s_3 g = 0 \Rightarrow \theta_2$ is arbitrary
3	$p_x = p_y = 0 \Rightarrow \theta_1$ is arbitrary $s_3 = 0 \Rightarrow \theta_2$ is arbitrary	4	$p_x = p_y = 0 \Rightarrow \theta_1$ is arbitrary $s_3 = 0 \Rightarrow \theta_2$ is arbitrary
5	$p_x = p_y = 0 \Rightarrow \theta_1$ is arbitrary $s_3 = 0 \Rightarrow \theta_2$ is arbitrary	6	$s_3 = 0 \Rightarrow \theta_2$ is arbitrary
7	$\left(f + s_3 g\right)^2 + c_3^2 g^2 = 0 \Rightarrow \theta_2$ is arbitrary	8	$p_x = p_y = 0 \Rightarrow \theta_1$ is arbitrary $d_3 + g = 0 \Rightarrow \theta_2$ is arbitrary
9	$p_x = p_y = 0 \Rightarrow \theta_1$ is arbitrary	10	$p_x = p_y = 0 \Rightarrow \theta_1$ is arbitrary
11	$d_3 + g = 0 \Rightarrow \theta_2$ is arbitrary	12	None
13	None		

Table 7. Undefined Configurations for the Inverse Displacement Solutions of the KS Layouts

5. Discussion

5.1 Application of the KS Layouts

The KS family of layouts can be used as main-arms for serial manipulators or as branches of parallel manipulators. For example, KS 1 is a common main-arm layout for numerous industrial serial manipulators. KS 4 is the branch configuration used in the RSI Research 6-DOF Master Controller parallel joy-stick (Podhorodeski, 1991). KS 8 is a very common layout used in many parallel manipulators including the Stewart-Gough platform (Stewart, 1965-66). KS 13 is the layout used in Cartesian manipulators.

The choice of which KS layout to use for a manipulator would depend on factors such as the shape of the desired workspace, the ease of manufacture of the manipulator, the task required, etc. For example, layout KS 1 provides a large spherical workspace. Having the second and third joints parallel in KS 1 allows for the motors of the main-arm to be mounted close to the base and a simple drive-train can used to move the third joint.

5.2 Reconfigurable Manipulators

KS layouts are also very useful for reconfigurable manipulators. Podhorodeski and Nokleby (2000) presented a Reconfigurable Main-Arm (RMA) manipulator capable of configuring into all five KS layouts comprised of revolute only joints (KS 1 to 5). Depending on the task required, one of the five possible layouts can be selected.

Yang, et al. (2001) showed how KS branches are useful for modular reconfigurable parallel manipulators.

5.3 Application of the Presented Displacement Solutions

5.3.1 Serial Manipulators

If a KS layout is to be used as a main-arm of a serial manipulator, the spherical wrist needs to be actuated. Figure 8 shows the zero-displacement configuration and Table 8 the D&H parameters for the common roll-pitch-roll spherical-wrist layout. The wrist shown in Figure 8 can be attached to any of the KS layouts.

Figure 8. Zero-Displacement Diagram for the Roll-Pitch-Roll Spherical Wrist

F_{j-1}	α_{j-1}	a_{j-1}	d_j	θ_j	F_j
F_{sph}	0	0	0	θ_4	F_4
F_4	$-\pi/2$	0	0	θ_5	F_5
F_5	$\pi/2$	0	0	θ_6	F_6

Table 8. D&H Parameters for the Roll-Pitch-Roll Spherical Wrist

For the KS family of layouts with a spherical wrist, the forward displacement solution is:

$$
{}^0_{ee}\mathbf{T} = \left({}^0_1\mathbf{T}\ {}^1_2\mathbf{T}\ {}^2_3\mathbf{T}\ {}^3_{sph}\mathbf{T} \right)\left({}^{sph}_4\mathbf{T}\ {}^4_5\mathbf{T}\ {}^5_6\mathbf{T} \right){}^6_{ee}\mathbf{T} = {}^0_{sph}\mathbf{T}\ {}^{sph}_6\mathbf{T}\ {}^6_{ee}\mathbf{T}
$$

(10)

where ${}^6_{ee}\mathbf{T}$ is the homogeneous transformation describing the end-effector frame F_{ee} with respect to frame F_6 and would be dependent on the type of tool attached, ${}^0_{sph}\mathbf{T}$ is defined in equation (3), and ${}^{sph}_6\mathbf{T}$ is:

$$
{}^{sph}_6\mathbf{T} = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 & 0 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 & 0 \\ -s_5c_6 & s_5s_6 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

(11)

For a 6-joint serial manipulator, Pieper (1968) demonstrated that for a manipulator with three axes intersecting, a closed-form solution to the inverse displacement problem can be found. As demonstrated by Paul (1981), for a 6-joint manipulator with a spherical wrist, the solutions for the main-arm and wrist displacements can be solved separately. Therefore, the presented inverse

displacement solutions for the KS family of layouts (see Section 4.2) can be used to solve for the main-arm joint displacements for serial manipulators that use KS layouts as their main-arm and have a spherical wrist.

For the inverse displacement solution of the main-arm joints, the location ($^0\mathbf{p}_{o_0 \to o_6}$) and orientation ($^0_6\mathbf{R}$) of frame F_6 with respect to the base frame in terms of the known value $^0_{ee}\mathbf{T}$ can be found from:

$$^0_6\mathbf{T} = {}^0_{ee}\mathbf{T}\left({}^6_{ee}\mathbf{T}\right)^{-1} = {}^0_{ee}\mathbf{T}\,{}^{ee}_6\mathbf{T} = \begin{bmatrix} {}^0_6\mathbf{R} & {}^0\mathbf{p}_{o_0 \to o_6} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{12}$$

where $^{ee}_6\mathbf{T}$ is constant and known. Since the manipulator has a spherical wrist:

$$^0\mathbf{p}_{o_0 \to o_{sph}} = {}^0\mathbf{p}_{o_0 \to o_6} = {}^0\mathbf{p}_{o_0 \to o_5} = {}^0\mathbf{p}_{o_0 \to o_4} = \begin{Bmatrix} p_x \\ p_y \\ p_z \end{Bmatrix} \tag{13}$$

where p_x, p_y, and p_z are found from equation (12). The inverse displacement solutions for the KS family of layouts discussed in Section 4.2 can now be used to solve for the main-arm joint displacements.

For the inverse displacement solution of the spherical wrist joints, in terms of the known value $^0_{ee}\mathbf{T}$, the orientation of F_6 with respect to the base frame, $^0_6\mathbf{R}$, was defined in equation (12). Note that:

$$^3_6\mathbf{R} = {}^0_3\mathbf{R}^T\,{}^0_6\mathbf{R} = {}^3_0\mathbf{R}\,{}^0_6\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{14}$$

Since the main arm joint displacements were solved above, the elements of matrix $^3_0\mathbf{R}$ are known values and thus the right-hand-side of equation (14) is known, i.e., r_{ij}, $i = 1$ to 3 and $j = 1$ to 3, are known values.

Substituting the elements of the rotation matrix $^3_6\mathbf{R} = {}^3_{sph}\mathbf{R}\,{}^{sph}_6\mathbf{R}$ into equation (14) yields:

$$^3_6\mathbf{R} = {}^3_{sph}\mathbf{R}\,{}^{sph}_6\mathbf{R} = {}^3_{sph}\mathbf{R} \begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & c_4 s_5 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 + c_4 c_6 & s_4 s_5 \\ -s_5 c_6 & s_5 s_6 & c_5 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{15}$$

where $_{sph}^{3}\mathbf{R}$ is dependent on the D&H parameter α_3 for the manipulator, i.e.:

$$
\begin{aligned}
&_{sph}^{3}\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{if } \alpha_3 = 0 \\
&_{sph}^{3}\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \text{if } \alpha_3 = \frac{\pi}{2} \\
&_{sph}^{3}\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}, \text{if } \alpha_3 = -\frac{\pi}{2}
\end{aligned}
\tag{16}
$$

Equation (15) can be used to derive expressions for the wrist joint displacements θ_4, θ_5, and θ_6. For example, if $\alpha_3 = \pi/2$, equation (15) becomes:

$$
\begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & c_4 s_5 \\ s_5 c_6 & -s_5 s_6 & -c_5 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 + c_4 c_6 & s_4 s_5 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}
\tag{17}
$$

Using element (2, 3) of equation (17) allows θ_5 to be solved as:

$$
\theta_5 = \text{atan2}\left(\pm \sqrt{1 - c_5^2}, c_5\right), \text{ where } c_5 = -r_{23}
\tag{18}
$$

Using elements (1, 3) and (3, 3) of equation (17) allows θ_4 to be solved as:

$$
\theta_4 = \text{atan2}(s_4, c_4), \text{ where } s_4 = \frac{r_{33}}{s_5} \ \& \ c_4 = \frac{r_{13}}{s_5}
\tag{19}
$$

Using elements (3, 1) and (3, 2) of equation (17) allows θ_6 to be solved as:

$$
\theta_6 = \text{atan2}(s_6, c_6)
\tag{20}
$$

where

$$
s_6 = \frac{-c_4 r_{31} + s_4 c_5 r_{32}}{-s_4^2 c_5^2 - c_4^2}
$$

$$
c_6 = \frac{-s_4 c_5 r_{31} - c_4 r_{32}}{-s_4^2 c_5^2 - c_4^2}
$$

Note that if $s_5 = 0$, joint axes 4 and 6 are collinear and the solutions for θ_4 and θ_6 are not unique. In this case, θ_4 can be chosen arbitrarily and θ_6 can be solved for.

Similar solutions can be found for the cases where α_3 equals 0 and $-\pi/2$.

5.3.2 Parallel Manipulators

Two frames common to the branches of the parallel manipulator are established, one frame attached to the base (F_{base}) and the other attached to the platform (F_{plat}). The homogeneous transformations from the base frame F_{base} to the base frame of each of the m branches F_{0_i} are denoted $^{base}_{0_i}\mathbf{T}$, i = 1 to m. The homogeneous transformations from the platform frame F_{plat} to the m passive spherical group frames F_{sph_i} are denoted $^{plat}_{sph_i}\mathbf{T}$, i = 1 to m. Note that for a given parallel manipulator all $^{base}_{0_i}\mathbf{T}$ and $^{plat}_{sph_i}\mathbf{T}$ would be known and would be constant.

For the inverse displacement solution, for the ith branch, the location and orientation of the spherical wrist frame, F_{sph_i}, with respect to the base frame of the branch, F_{0_i}, in terms of the known value $^{base}_{plat}\mathbf{T}$ can be found from:

$$
^{0_i}_{sph_i}\mathbf{T} = \left(^{base}_{0_i}\mathbf{T} \right)^{-1}{}^{base}_{plat}\mathbf{T}\,^{plat}_{sph_i}\mathbf{T} = {}^{0_i}_{base}\mathbf{T}\,^{base}_{plat}\mathbf{T}\,^{plat}_{sph_i}\mathbf{T} = \begin{bmatrix} {}^{0_i}_{sph_i}\mathbf{R} & {}^{0_i}\mathbf{p}_{o_{0_i} \to o_{sph_i}} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}
\tag{21}
$$

where $^{0_i}_{sph_i}\mathbf{R}$ is the orientation of F_{sph_i} with respect to F_{0_i} and $^{0_i}\mathbf{p}_{o_{0_i} \to o_{sph_i}}$ is the position vector from the origin of F_{0_i} to the origin of F_{sph_i} with respect to F_{0_i}. The position vector is defined as:

$$
^{0_i}\mathbf{p}_{o_{0_i} \to o_{sph_i}} = \begin{Bmatrix} p_{x_i} \\ p_{y_i} \\ p_{z_i} \end{Bmatrix}
\tag{22}
$$

where p_{x_i}, p_{y_i}, and p_{z_i} are known values. The inverse displacement solutions for the KS family of layouts shown in Section 4.2 can then be used to solve for the joint displacements for branches i =1 to m.

Unlike the forward displacement problem of serial manipulators, the forward displacement problem of parallel manipulators is challenging. Raghavan

(1993) showed that for the general 6-6[1] Stewart-Gough platform up to 40 solutions could exist to the forward displacement problem. Note that the notation *i-j* denotes the number of connection points of the branches at the base and platform, respectively.

Innocenti and Parenti-Castelli (1990) showed that for a class of parallel manipulators that have the branches connected at three distinct points on the end-effector platform (e.g., 6-3 and 3-3 layouts), the forward displacement problem can be reduced to a 16th order polynomial of one variable leading to a maximum of 16 possible solutions to the forward displacement problem.

Numerous researchers (e.g., Inoue, et al. (1986); Waldron, et al. (1989); Cheok, et al. (1993); Merlet (1993); Notash and Podhorodeski (1994 and 1995); and Baron and Angeles (2000)) have shown that utilizing redundant sensing in parallel manipulators is necessary to achieve a unique solution to the forward displacement problem. For the purposes of the solutions presented here, it is assumed that the manipulator is redundantly sensed and that the positions of the passive spherical groups ($\mathbf{p}_i$, $i = 1$ to m) would be known.

Noting that:

$$
{}^{base}_{plat}\mathbf{T} = \begin{bmatrix} {}^{base}_{plat}\mathbf{R} & {}^{base}\mathbf{p}_{o_{base} \to o_{plat}} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}
$$

(23)

For 6-3 and 3-3 layouts, the origin of F_{plat} can be found as:

$$
{}^{base}\mathbf{p}_{o_{base} \to o_{plat}} = \frac{{}^{base}(\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3)}{3}
$$

(24)

where $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ are the positions of the passive spherical groups. The positions of the passive spherical groups can be found using the solutions presented in Tables 3 to 4.

The orientation of the platform frame can be found as:

$$
{}^{base}_{plat}\mathbf{R} = {}^{base}\begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} \end{bmatrix}
$$

(25)

where

---

[1] Note that the notation *i-j* denotes the number of connection points of the branches at the base and platform, respectively.

$$^{base}\mathbf{n}= \,^{base}\!\left(\frac{\mathbf{p}_3 - \mathbf{p}_2}{\|\mathbf{p}_3 - \mathbf{p}_2\|} \right)$$

$$^{base}\mathbf{a} = \frac{^{base}\mathbf{n} \times \,^{base}\mathbf{c}}{\left\| ^{base}\mathbf{n} \times \,^{base}\mathbf{c} \right\|}$$

$$^{base}\mathbf{o} = \,^{base}\mathbf{a} \times \,^{base}\mathbf{n}$$

with

$$^{base}\mathbf{c}= \,^{base}\!\left(\frac{\mathbf{p}_1 - \mathbf{p}_2}{\|\mathbf{p}_1 - \mathbf{p}_2\|} \right)$$

For 6-6 and 3-6 layouts, the origin of F_{plat} can be found as:

$$^{base}\mathbf{p}_{o_{base} \to o_{plat}} = \frac{^{base}(\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3 + \mathbf{p}_4 + \mathbf{p}_5 + \mathbf{p}_6)}{6} \tag{26}$$

where $\mathbf{p}_1$ to $\mathbf{p}_6$ are the positions of the passive spherical groups. Note that it is assumed that the passive spherical groups are symmetrically distributed about the platform. The positions of the passive spherical groups can be found using the solutions presented in Tables 3 and 4.

The orientation of the platform frame can be found as:

$$^{base}_{plat}\mathbf{R} = \,^{base}\begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} \end{bmatrix} \tag{27}$$

where

$$^{base}\mathbf{n}= \,^{base}\!\left(\frac{\mathbf{p}_c - \mathbf{p}_b}{\|\mathbf{p}_c - \mathbf{p}_b\|} \right)$$

$$^{base}\mathbf{a} = \frac{^{base}\mathbf{n} \times \,^{base}\mathbf{c}}{\left\| ^{base}\mathbf{n} \times \,^{base}\mathbf{c} \right\|}$$

$$^{base}\mathbf{o} = \,^{base}\mathbf{a} \times \,^{base}\mathbf{n}$$

with

$$^{base}\mathbf{c} = \frac{\mathbf{p}_a - \mathbf{p}_b}{\|\mathbf{p}_a - \mathbf{p}_b\|}$$

$$\mathbf{p}_a = (\mathbf{p}_1 + \mathbf{p}_2)/2$$

$$\mathbf{p}_b = (\mathbf{p}_3 + \mathbf{p}_4)/2$$

$$\mathbf{p}_c = (\mathbf{p}_5 + \mathbf{p}_6)/2$$

6. Conclusions

The complete set of layouts belonging to a kinematically simple (KS) family of spatial joint layouts were presented. The considered KS layouts were defined as ones in which the manipulator (or branch of a parallel manipulator) incorporates a spherical group of joints at the wrist with a main-arm comprised of successfully parallel or perpendicular joints with no unnecessary offsets or lengths between joints. It was shown that there are
13 layouts having unique kinematics belonging to the KS family: five layouts comprised of three revolute joints; five layouts comprised of two revolute joints and one prismatic joint; two layouts comprised of one revolute joint and two prismatic joints; and one layout comprised of three prismatic joints. In addition, it was shown that there are a further five kinematically-simple layouts having unique joint structures, but kinematics identical to one of the 13 KS layouts.
Zero-displacement diagrams, D&H parameters, and the complete forward and inverse displacement solutions for the KS family of layouts were presented. It was shown that for the inverse displacement problem up to four possible solutions exist for KS 1 to 6, 8, 9, and 10, up to two possible solutions exist for KS 7, 11, and 12, and only one solution exists for KS 13. The application of the KS family of joint layouts and the application of the presented forward and inverse displacement solutions to both serial and parallel manipulators was discussed.

Acknowledgements

7. References

Baron, L. & Angeles, J. (2000). The Direct Kinematics of Parallel Manipulators Under Joint-Sensor Redundancy. *IEEE Transactions on Robotics and Automation*, Vol. 16, No. 1, pp. 12-19.

Cheok, K. C.; Overholt, J. L. & Beck, R. R. (1993). Exact Methods for Determining the Kinematics of a Stewart Platform Using Additional Displacement Sensors. *Journal of Robotic Systems*, Vol. 10, No. 5, pp. 689-707.

Craig, J. J. (1989). *Introduction To Robotics: Mechanics and Control - Second Edition*, Addison-Wesley Publishing Company, Don Mills, Ontario, Canada.

Denavit, J. & Hartenberg, R. S. (1955). A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Transactions of the ASME, Journal of Applied Mechanics*, June, pp. 215-221.

Innocenti, C. & Parenti-Castelli, V. (1990). Direct Position Analysis of the Stewart Platform Mechanism. *Mechanism and Machine Theory*, Vol. 25, No. 6, pp. 611-621.

Inoue, H.; Tsusaka, Y. & and Fukuizumi, T. (1986). Parallel Manipulator, In: Robotics Research: The Third International Symposium, Faugeras, O. D. & Giralt, G., (Ed.), pp. 321-327, MIT Press, Cambridge, Massachusetts, USA.

Merlet, J.-P. (1993). Closed-Form Resolution of the Direct Kinematics of Parallel Manipulators Using Extra Sensors Data, *Proceedings of the 1993 IEEE International Conference on Robotics and Automation - Volume 1*, May 2-6, 1993, Atlanta, Georgia, USA, pp. 200-204.

Notash, L. & Podhorodeski, R. P. (1994). Complete Forward Displacement Solutions for a Class of Three-Branch Parallel Manipulators. *Journal of Robotic Systems*, Vol. 11, No. 6, pp. 471-485.

Notash, L. & Podhorodeski, R. P. (1995). On the Forward Displacement Problem of Three-Branch Parallel Manipulators. *Mechanism and Machine Theory*, Vol. 30, No. 3, pp. 391-404.

Paul, R. P. (1981). *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, Cambridge, Massachusetts, USA.

Pieper, D. L. (1968). *The Kinematics of Manipulators Under Computer Control*. Ph.D. Dissertation, Stanford University, Stanford, California, USA.

Podhorodeski, R. P. (1991). A Screw Theory Based Forward Displacement Solution for Hybrid Manipulators, *Proceedings of the 2nd National Applied Mechanics and Robotics Conference - Volume I*, November 3-6, 1991, Cincinnati, Ohio, USA, pp. IIIC.2--1 - IIIC.2-7.

Podhorodeski, R. P. (1992). Three Branch Hybrid-Chain Manipulators: Structure, Displacement, Uncertainty and Redundancy Related Concerns, *Proceedings of the 3rd Workshop on Advances in Robot Kinematics*, September 7-9, 1992, Ferrara, Italy, pp. 150-156.

Podhorodeski, R. P. & Nokleby, S. B. (2000). Reconfigurable Main-Arm for As-
       sembly of All Revolute-Only Kinematically Simple Branches. *Journal of
       Robotic Systems*, Vol. 17, No. 7, pp. 365-373.
Podhorodeski, R. P. & Pittens, K. H. (1992). A Class of Hybrid-Chain Manipu-
       lators Based on Kinematically Simple Branches, *Proceedings of the 1992
       ASME Design Technical Conferences - 22nd Biennial Mechanisms Conference*,
       September 13-16, 1992, Phoenix, Arizona, USA, pp. 59-64.
Podhorodeski, R. P. & Pittens, K. H. (1994). A Class of Parallel Manipulators
       Based on Kinematically Simple Branches. *Transactions of the ASME,
       Journal of Mechanical Design*, Vol. 116, No. 3, pp. 908-914.
Raghavan, M. (1993). The Stewart Platform of General Geometry Has 40 Con-
       figurations. *Transactions of the ASME, Journal of Mechanical Design*, Vol.
       115, No. 2, pp. 277-282.
Stewart, D. (1965-66). *A Platform With Six Degrees of Freedom*. Proceedings of
       the Institution of Mechanical Engineers, Vol. 180, Part 1, No. 15, pp. 371-
       386.
Waldron, K. J.; Raghavan, M. & Roth, B. (1989). Kinematics of a Hybrid Series-
       Parallel Manipulation System. *Transactions of the ASME, Journal of Dy-
       namic Systems, Measurement, and Control*, Vol. 111, No. 2, pp. 211-221.
Yang, G.; Chen, I.-M.; Lim, W. K. & Yeo, S. H. (2001). Kinematic Design of
       Modular Reconfigurable In-Parallel Robots. *Autonomous Robots*, Vol. 10,
       Issue 1, pp. 83-89.

On the Analysis and Kinematic Design of a Novel 2-DOF Translational Parallel Robot

Jinsong Wang, Xin-Jun Liu and Chao Wu

1. Introduction

The conceptual design of parallel robots can be dated back to the time when Gough established the basic principles of a device with a closed-loop kinematic structure (Gough 1956), that can generate specified position and orientation of a moving platform so as to test tire wear and tear. Based on this principle, Stewart designed a platform for use as an aircraft simulator in 1965 (Stewart 1965). In 1978, Hunt (1978) made a systematic study of robots with parallel kinematics, in which the planar 3-RPS (R-revolute joint, P-prismatic joint, and S-spherical joint) parallel robot is a typical one. Since then, parallel robots have been studied extensively by numerous researchers.

The most studied parallel robots are with 6 DOFs. These parallel robots possess the advantages of high stiffness, low inertia, and large payload capacity. However, they suffer the problems of relatively small useful workspace and design difficulties (Merlet 2000). Furthermore, their direct kinematics possess a very difficult problem; however the same problem of parallel robots with 2 and 3 DOFs can be described in a closed form (Liu 2001). As is well known, there are three kinds of singularities in parallel robots (Gosselin and Angeles 1990). Moreover, not all singularities of a 6-DOF parallel robot can be found easily. But for a parallel robot with 2 and 3 DOFs, the singularities can always be identified readily. For such reasons, parallel robots with less than 6 DOFs, especially 2 and 3 DOFs, have increasingly attracted more and more researchers' attention with respect to industrial applications (Tsai & Stamper 1996; Ceccarelli 1997; Tonshoff et al 1999; Siciliano 1999; Liu et al. 2001; Liu et al. 2005; Liu & Kim 2003). In these designs, parallel robots with three translational DOFs have been playing important roles in the industrial applications (Tsai & Stamper 1996; Clavel 1988; Hervé 1992; Kim & Tsai 2002; Zhao & Huang 2000; Carricato & Parenti-Castelli 2001; Kong & Gosselin 2002; Liu et al. 2003), especially, the DELTA robot (Clavel 1988), which is evident from the fact that the design of the DELTA robot is covered by a family of 36 patents (Bonev 2001). Tsai's robot (Tsai & Stamper 1996), in which each of the three legs consists of a parallelogram, is the first design to solve the problem of UU chain. A 3-

translational-DOF parallel robot, Star, was designed by Hervé based on group theory (Hervé 1992). Such parallel robots have wide applications in the industrial world, e.g., pick-and-place application, parallel kinematic machines, and medical devices.

The most famous planar 2-DOF parallel robots (Asada & Kanade 1983; McCloy 1990; Gao et al. 1998) are the well-known five-bar mechanism with prismatic actuators or revolute actuators. In the case of the robot with revolute actuators, the mechanism consists of five revolute pairs and the two joints fixed to the base are actuated. In the case of the robot with prismatic actuators, the mechanism consists of three revolute pairs and two prismatic joints, in which the prismatic joints are usually actuated. The output of the robot is the translational motion of a point on the end-effector, i.e., the orientation of the end-effector is also changed correspondingly. Accordingly, some versions of the 2-DOF translational parallel robot (TPR) have been disclosed. One of them has been applied in precise pick & place operations at high speed in IWF at Technical University of Braunschweig. In 2001, another 2-DOF TPR has been proposed for the conceptual design of a 5-axis machine tool (Liu 2001). The structure, kinematics and dynamics of the TPR were discussed in details (Liu et al., 2002; Liu et al., 2005). Recently, a 2-DOF TPR with revolute actuators was introduced (see Table 1 in (Liu & Wang, 2003); Huang et al., 2004). The TPR presented in (Liu 2001; Liu et al., 2005) has been used in the design of a planer machine tool with a gantry structure instead of a traditional one with serial chains to improve its stiffness and inertia characteristics. However, all of these TPRs consist of at least of one parallelogram. Here, a novel 2-DOF TPR with only revolute and prismatic joints will be proposed. The robot can position an objective with constant orientation with high speed.

As it is one of the most important and challenging issues in the parallel robot, optimal kinematic design has drawn more and more researchers' attention (Gosselin & Angeles, 1989; Chablat & Wenger, 2003; Stock & Miller, 2004; Ottaviano & Ceccarelli, 2002; Cervantes-Sánchez et al., 2001). The objective of optimal kinematic design is determining the dimension or link length of a robot with respect to desired performance(s). Due to the parameter infinity and the instability of performance in a whole workspace, optimal kinematic design is one of the most challenging problems in the field of parallel robot. The commonly used methods are first to develop an objective function and then to reach the result using the numerical method with an algorithm. These methodologies have the disadvantages in common, i.e., the objective function is difficult to be established; the numerical procedure may lead to a solution that is quite far away from the optimal solution; the process is iterative and time consuming; and, most fatally, only one optimal solution can be provided. To overcome the disadvantages, in this chapter, a new optimal design methodology will be proposed for the parallel robot. Using a normalization method, the dimensional characteristic parameters of the robot will be normalized. The nor-

malization can guarantee that a dimensional robot and its corresponding normalized robot are similar not only in size but also in performance. The dimensional robot is defined as *similarity robot* (SR), and the normalized robot is referred to as *basic similarity robot* (BSR). A design space which embodies all kinds of BSRs will be established. The space can be used not only in analysis but also in the optimal design of the parallel robot. Within the design space, the performance atlas that illustrates the relationship between a performance index and the BSRs can be plotted. The optimal kinematic design can be implemented with respect to the performance atlases. Design examples will be finally given in the chapter. Compared with the traditional design methods, the proposed optimal design methodology has some advantages as follows: (a) one performance index corresponds to one atlas; (b) for such a reason in (a), the fact that some performance indices are antagonistic is no longer a problem in the design; (c) the optimal design process can consider multi-objective functions or multi-indices, and also guarantees the optimal result; and finally, (d) the design method provides a set of possible solutions, and ideally, all the design solutions.

2. Description of the 2-DOF TPR and its Topological Architectures

2.1 Architecture description

The novel 2-DOF translational parallel robot proposed here is shown in Fig. 1(a). A schematic of the robot is shown in Fig. 1(b). The end-effector of the robot is connected to the base by two kinematic legs 1 and 2. Leg 1 consists of three revolute joints and leg 2 two revolute joints and one cylinder joint, or three revolute joints and one prismatic joint. In each leg, the revolute joints are parallel to each other. The axes of the revolute joints in leg 1 are normal to those of the joints in leg 2. The two joints attached to the end-effector are put in the adjacent sides of a square. The kinematic chain of the robot is denoted as RRR-RRC (C-cylinder joint) or RRR-RRRP.

2.2 Capability

Here, a Plücker coordinate like $\$_j = (\overline{x}, \overline{y}, \overline{z}; \hat{x}, \hat{y}, \hat{z})$ is used to describe the capability of an object j. In $\$_j$, $Tr_j = (\overline{x}, \overline{y}, \overline{z})$ and $Ro_j = (\hat{x}, \hat{y}, \hat{z})$ express the translation and rotation of the object, respectively. If an element in $\$$ is equal to 0, there is no such a translation or rotation. If it is equal to 1, there is the capability. For example, $\overline{x} = 0$ means that the object has no the translation along the x-axis; $\hat{y} = 1$ indicates that the object can rotate about the y-axis.

Observing only the leg 1, the Plücker coordinate of the end-effector in the leg can be written as $\$_1$=(1, 1, 0; 0, 0, 1). Letting the leg 1 alone, the Plücker coordinate of the end-effector with the leg 2 can be expressed as $\$_2$=(1, 1, 1; 1, 0, 0). Then, the intersection of the two Plücker coordinates $\$_1$ and $\$_2$ is $, i.e.,

$$\$=\$_1 \cap \$_1=(1, 1, 0; 0, 0, 1) \cap (1, 1, 1; 1, 0, 0)= (1, 1, 0; 0, 0, 0) \tag{1}$$

which describes the capability of the robot, i.e., the translations of the end-effector along the x and y axes. That means the end-effector has two purely translational degrees of freedom with respect to the base.

(a)

(b)

Figure 1. The 2-DOF translational parallel robot: (a) the CAD model; (b) the schematic

2.3 Topological architectures

Observing the robot shown in Fig. 1, it is not difficult to reach such a conclusion that if the axes of the joints in the leg 1 are normal to those of the joints in the leg 2 the robot will have two translational DOFs. Based on this idea, some topological architectures are shown in Fig. 2. It is noteworthy that the leg 2 shown in Fig. 1 and Fig. 2 can be also the kinematic chain RRR(Pa) shown in Fig. 3, where Pa means planar parallelogram.

3. Kinematics Analysis

Although the robot has some topologies, this chapter consideres only the architecture shown in Fig. 1. In this section, the inverse and forward kinematics of the robot will be given.

3.1 Inverse kinematics

As illustrated in Fig. 1(b), a reference frame $\Re : O - xy$ is fixed to the base at the joint point A_1 and a moving reference frame $\Re' : O' - x'y'$ is attached to the end-effector, where O' is the reference point on the end-effector. Vectors $\boldsymbol{p}_{i\Re}$ ($i = 1, 2$) will be defined as the position vectors of points P_i in frames $\Re$, and vectors $\boldsymbol{b}_{i\Re}$ ($i = 1, 2$) as the position vectors of points B_i in frame $\Re$. The geometric parameters of the robot are $A_1B_1 = R_1(r_1)$, $B_1P_1 = R_2(r_2)$, $PP_i = R_3(r_3)$, $A_2B_2 = L_1(l_1)$, $B_2P_2 = L_2(l_2)$, and the distance between the point A_1 and the guideway is $L_3(l_3)$, where R_n and L_n (n=1,2,3) are dimensional parameters, and r_n and l_n non-dimensional parameters. The position of point O' in the fixed frame $\Re$ is denoted as vector

$$c_\Re = (x, \quad y)^{\mathrm{T}} \tag{2}$$

(a)

(b)   (c)

Figure 2. Some topological architectures: (a) RRR-RRRP chain (arrangement is different from that shown in Fig. 1); (b) RPR-RRRP chain; (c) PRR-RRRP chain.

The vectors of $\boldsymbol{b}_{1\Re}$ in the fixed frame $\Re$ can be written as

$$\boldsymbol{b}_{1\Re} = \left(R_1 \cos\theta \quad R_1 \sin\theta\right)^{\mathrm{T}} \tag{3}$$

where θ is the actuated input for the leg 1. Vector $\boldsymbol{p}_{1\Re}$ in the fixed frame $\Re$ can be written as

$$\boldsymbol{p}_{1\Re} = \left(-R_3 \quad 0\right)^{\mathrm{T}} + \boldsymbol{c}_{\Re} = \left(x - R_3 \quad y\right)^{\mathrm{T}} \tag{4}$$

Figure 3. One topological architecture of the leg 2

The inverse kinematics problem of the leg 1 can be solved by writing following constraint equation

$$\left\| \boldsymbol{p}_{1\Re} - \boldsymbol{b}_{1\Re} \right\| = R_2 \tag{5}$$

that is

$$\left(x-R_3-R_1\cos\theta\right)^2+\left(y-R_1\sin\theta\right)^2=R_2^2 \tag{6}$$

Then, there is

$$\theta=2\tan^{-1}(m) \tag{7}$$

where

$$m=\frac{-b+\sigma\sqrt{b^2-4ac}}{2a} \tag{8}$$

$\sigma=1$ or -1

$a=\left(x-R_3\right)^2+y^2+R_1^2-R_2^2+2\left(x-R_3\right)R_1$

$b=-4yR_1$

$c=\left(x-R_3\right)^2+y^2+R_1^2-R_2^2-2\left(x-R_3\right)R_1$

For the leg 2, it is obvious that

$$s=x \tag{9}$$

in which s is the input of the leg 2. From Eqs. (8) and (9), we can see that there are two solutions for the inverse kinematics of the robot. Hence, for a given robot and for prescribed values of the position of the end-effector, the required actuated inputs can be directly computed from Eqs. (7) and (9). To obtain the configuration as shown in Fig.1, parameter σ in Eq. (8) should be 1. This configuration is called the "+" working mode. When $\sigma=-1$, the corresponding configuration is referred to as the "—" working mode.

3.2 Forward kinematics

The forward kinematic problem is to obtain the output with respect to a set of given inputs. From Eqs. (6) and (9),one obtains

$$y=e+\sigma\sqrt{f} \tag{11}$$

and

$$x=s \tag{12}$$

where, $e=R_1\sin\theta$ and $f=R_2^2-\left(s-R_3-R_1\cos\theta\right)^2$. Therefore, there are also two forward kinematic solutions for the robot. The parameter $\sigma=-1$ corre-

sponds to the configuration shown in Fig. 1, which is denoted as the *down-configuration*. When $\sigma = 1$, the configuration is referred to as the *up-configuration*. These two kinds of configurations correspond to two kinds of assembly modes of the robot.

Figure 4 illustrates two kinds of working modes of the robot. The two kinds of assembly modes are shown in Fig. 5. In this chapter, the robot with both the "+" working mode and *down-configuration* will be considered only.

Figure 4. Two kinds of working modes: (a) "+" working mode; (b) "−" working mode

Figure 5. Two kinds of assembly modes: (a) *down-configuration*; (b) *up-configuration*

4. Singularity Analysis

4.1 Jocabian matrix

Equations (6) and (9) can be differentiated with respect to time to obtain the velocity equations. This leads to

$$\dot{s} = \dot{x} \tag{13}$$

$$R_1\left[y\cos\theta - (x - R_3)\sin\theta\right]\dot{\theta} = (x - R_3 - R_1\cos\theta)\dot{x} + (y - R_1\sin\theta)\dot{y} \tag{14}$$

which can be written in an equation of the form

$$A\dot{q} = B\dot{p} \tag{15}$$

where $\dot{q} = \begin{pmatrix} \dot{s} & \dot{\theta} \end{pmatrix}^{\mathrm{T}}$ and $\dot{p} = \begin{pmatrix} \dot{x} & \dot{y} \end{pmatrix}^{\mathrm{T}}$ are the joint and Cartesian space velocity vectors, respectively, and A and B are , respectively, the 2×2 matrices and can be expressed as

$$A = \begin{bmatrix} 1 & 0 \\ 0 & R_1 y\cos\theta - R_1(x - R_3)\sin\theta \end{bmatrix}, \text{ and } B = \begin{bmatrix} 1 & 0 \\ x - R_3 - R_1\cos\theta & y - R_1\sin\theta \end{bmatrix} \tag{16}$$

If matrix A is nonsingular, the Jacobian matrix of the robot can be obtained as

$$J = A^{-1}B = \begin{bmatrix} 1 & 0 \\ \dfrac{x - R_3 - R_1\cos\theta}{R_1 y\cos\theta - R_1(x - R_3)\sin\theta} & \dfrac{y - R_1\sin\theta}{R_1 y\cos\theta - R_1(x - R_3)\sin\theta} \end{bmatrix} \tag{17}$$

from which one can see that there is no any parameter of L_n (n=1,2,3) in this matrix.

4.2 Singularity

In the parallel robot, singularities occur whenever A, B or both, become singular. As a singularity leads to an instantaneous change of the robot's DOF, the analysis of parallel robots has drawn considerable attention. For the parallel robot studied here, since there is no any parameter of the leg 2 involved in the Jacobian matrix (see Eqs. (16) and (17)), the singularity is actually only that of the leg 1.

The stationary singularity occurs when A becomes singular but B remains invertible. $|A| = 0$ leads to $R_1 y\cos\theta - R_1(x - R_3)\sin\theta = 0$, i.e. $\tan\theta = y/(x - R_3)$.

Physically, this corresponds to the configuration when leg 1 $A_1 B_1 P_1$ is completely extended or folded. This singularity is also referred to as the serial singularity. For example, for the robot with the parameters $R_1 = 1.2\text{mm}$ and $R_2 = 0.8\text{mm}$, two configurations of this kind of singularity are shown in Fig. 6. The loci of point P for this kind of singularity can be expressed as

$$C_{fir\_o}: (x - R_3)^2 + y^2 = (R_1 + R_2)^2 \tag{18}$$

and

$$C_{fir\_i}: (x - R_3)^2 + y^2 = (R_1 - R_2)^2 \tag{19}$$

For the above example, if $R_3 = 0.5\text{mm}$ the loci of point P are shown in Fig. 7. Note that, $R_1 = 0$ leads to $\det(A) = 0$ as well. Therefore, $R_1 = 0$ also results in this kind of singularity.

The uncertainty singularity, occurring only in closed kinematics chains, arises when B becomes singular but A remains invertible. $|B| = 0$ results in $y = R_1 \sin\theta$. Physically, this corresponds to the configuration when link $B_1 P_1$ is parallel to the x-axis. Two such configurations are shown in Fig. 8. In such a singularity, the loci of point P can be written as

$$C_{\text{sec}\_r}: (x - R_3 - R_2)^2 + y^2 = R_1^2 \tag{20}$$

and

$$C_{\text{sec}\_l}: (x - R_3 + R_2)^2 + y^2 = R_1^2 \tag{21}$$

Figure 6. Two kinds configurations of the stationary singularity: (a) $A_1 B_1 P_1$ is completely extended; (b) $A_1 B_1 P_1$ is completely folded

Figure 7. Singular loci of point P when the robot is in the stationary singularity

Figure 8. Two kinds configurations of the uncertainty singularity: (a) point P_1 is in the right of point B_1; (b) point P_1 is in the left of point B_1

It is noteworthy that the singular loci of a robot when R_1 is greater than R_2 is different from those when R_2 is greater than R_1. The two cases are shown in Fig. 9. From Figs. 7 and 9, we can see that the uncertainty singular loci are always inside the region bounded by the stationary singular loci; and there are usually tangent points between the two kinds of loci.

The analysis on the kinematics of the robot shows that there are two solutions for both the inverse and forward kinematics. Any one of the singularities will result in the change of solution number of the kinematics. For example, the stationary singularity leads to the loss of solution number of the inverse kinematics. While in the uncertainty singular configuration, the solution number of the forward kinematics can be less or more than two. Then the stationary singularity can be called the inverse kinematic singularity, and the uncertainty singularity the forward kinematic singularity.

Figure 9. Singular loci of point P when the robot is in the stationary singularity: (a) $R_1 \geq R_2$; (b) $R_1 < R_2$

5. Workspace Analysis

One of the most important issues in the design process of a robot is its workspace. For parallel robots, this issue may be more critical since parallel robots will sometimes have rather a limited workspace.

5.1 Theoretical workspace

Theoretical workspace of the studied robot is defined as the region that the output point can reach if θ changes from 0 to 2π and s between $-\infty$ and ∞ without the consideration of interference between links and the singularity.

From Eq. (6), one can see that if θ is specified, the workspace of the leg 1 is a circle centered at the point $(R_1 \cos\theta + R_3, \quad R_1 \sin\theta)$ with a radius of R_2. The circle is denoted as C_{11}. If θ_i changes from 0 to 2π, the center point is located at a circle centered at point $(R_3 \quad 0)$ with a radius of R_1. The circle is denoted as C_{12}. Then, the workspace of the leg is the enveloping region of the circle C_{11} when its center rolls at the circle C_{12}. Actually, the enveloping region is an annulus bounded by two circles C_{fir_o} and C_{fir_i} given in Eqs. (18) and (19), respectively. Especially, when $R_1 = R_2$ the workspace is the region bounded by the circle C_{fir_o}.

Thinking about the architecture of the studied parallel robot, we can see that the workspace of leg 1 is limited with respect to the parameters R_1 and R_2.

But, the workspace of leg 2 has the advantage along *x*-axis. That means the workspace can be infinite if the input *s* is not limited. Practically, this case cannot occur. However, to enlarge the workspace of the robot, we are sure to find a solution that the workspace of leg 1 can be embodied by that of leg 2. Actually, enlarging the workspace is our pursuing objective. In this sense, the workspace of the robot should be that of the leg 1. The workspace of the leg 1 is then our research objective.

For example, the theoretical workspace of leg 1 of the robot with parameters $R_1 = 1.2\text{mm}$, $R_2 = 0.8\text{mm}$ and $R_3 = 0.5\text{mm}$ is shown as the shaded region in Fig. 10. The theoretical workspace and any other type of workspace of the robot can be that which embodies the corresponding workspace of the leg 1 by assigning appropriate values to the parameters L_n (*n*=1,2,3), which will be described in details in the section 7.2. Therefore, in this chapter, the workspace of the leg 1 is regarded as the workspace of the parallel robot. The theoretical workspace is actually bounded by the stationary singularity loci C_{fir_i} and C_{fir_o}. Its area can be calculated by

$$S_{tw} = \pi\left[(R_1 + R_2)^2 - (R_1 - R_2)^2\right] = 4\pi R_1 R_2 \tag{21}$$

From Fig. 9, we can see that within the theoretical workspace there is stationary singularity.

Figure 10. Theoretical workspace of the robot

5.2 Usable workspace

As there exist singular loci inside the theoretical workspace, if a robot wants to move from one point to another it maybe should passes a singular configuration. That means it maybe changes from one working mode to another. In practice, changing working mode during the working process is definitely impossible. Therefore, we should find out a working space without singularity.

The *usable workspace* is defined as the maximum continuous workspace that contains no singular loci inside but bounded by singular loci outside. According to this definition, not every point within the usable workspace can be available for a practical robot. The robot will be out of control at the points on the boundaries and their neighborhoods. But within this region, the robot with a specified working mode can move freely.

In Section 4.2, two kinds of singular loci have been presented for the robot as shown in Fig. 9. The stationary singularity is actually the boundary of a theoretical workspace. Then, a robot with every working mode can have such singular loci. However, as the uncertainty singularity occurs inside the workspace, not every working mode has all such singularities. Normally, for most parallel robots studied here, there are four tangent points between the two kinds of singular loci. The points can be used to identify which singular loci a specified working mode can have. For example, all singular loci of the robot $R_1 = 1.2\text{mm}$, $R_2 = 0.8\text{mm}$ and $R_3 = 0.5\text{mm}$ are shown in Fig. 9. Fig. 11 shows some singular configurations and singular loci of the robot. As shown in Fig. 11, there are four tangent points m, v, q and k between the four singular loci C_{fir_i}, C_{fir_o}, C_{sec_l} and C_{sec_r}. At these four points, both of the stationary and uncertainty singularities occur. The four points divide the singular curves C_{sec_l} and C_{sec_r} into four parts. At the arcs $m1q$ and $v3k$, the robot is in singular only when it is with the "+" mode. At the arcs $m2q$ and $v4k$, the working mode "−" is in singular.

Figure 11. The uncertainty singular loci of a robot with different working modes

What we are concerned about here is the robot with the "+" working mode. Fig. 12 shows all singular loci of such kinds of robots.

The theoretical workspace is divided into two parts by the singular loci shown in Fig. 12, which can be used to identify the usable workspaces of the robots with the "+" working mode and, at the same time, the *down-configuration*. In order to reduce the occupating space, the lower region shown in Fig. 12 is referred to as the *usable workspace* of the parallel robot. They are shown as the shaded region in Fig. 13. Actually, the *usable workspace* is the half of the theoretical workspace. The area can be calculated by

$$S_{uw} = \frac{\pi}{2}\left[(R_1 + R_2)^2 - (R_1 - R_2)^2\right] = 2\pi R_1 R_2 \tag{22}$$

Figure 12. Singular loci of the robot with the "+" working mode: (a) $R_1 \geq R_2$; (b) $R_1 < R_2$

Figure 13. Usable workspace of the robot with both the "+" working mode and *down-configuration*: (a) $R_1 \geq R_2$; (b) $R_1 < R_2$

5.3 Workspace atlas

To apply a specified robot in practice, we usually should determine the link lengths with respect to a desired application. This is actually the so-called optimal kinematic design (parameter synthesis) of the robot. In such a process, one of the most classical tools that has been using is the chart.

Chart is a kind of tool to show the relationship between concerned parameters. As it is well known, the performance of a parallel robot depends not only on the pose of the end-effector but also on the link lengths (dimensions). Disregarding the pose, each of the links can be the length between zero and infinite. And there are always several links in a parallel robot. Then the combination of the links with different lengths will be infinite. They undoubtedly have different performance characteristics. In order to summarize the characteristics of a performance, we must show the relationship between it and geometrical parameters of the parallel robot. To this end, a finite space that must contain all kinds of robots (with different link lengths) should be first developed. Next is to plot the chart considering a desired performance. In this paper, the space is referred to as the design space. The chart that can show the relationship between performances and link lengths is referred to as atlas.

5.3.1 Development of a design space

The Jacobian matrix is the matrix that maps the relationship between the velocity of the end-effector and the vector of actuated joint rates. This matrix is the most important parameter in the field. Almost all performances are depended on this parameter. Therefore, based on the Jacobian matrix, we can identify which geometrical parameter should be involved in the analysis and kinematic design.

For the parallel robot considered here, there are three parameters in the Jacobian matrix (see Eq. (17)), which are R_1, R_2 and R_3. Theoretically, any one of the parameters R_1, R_2 and R_3 can have any value between zero and infinite. This is the biggest difficulty to develop a design space that can embody all robots (with different link lengths) within a finite space. For this reason, we must eliminate the physical link size of the robots.

Let

$$D = (R_1 + R_2 + R_3)/3 \tag{23}$$

One can obtain 3 non-dimensional parameters r_i by means of

$$r_1 = R_1/D, \; r_2 = R_2/D, \; r_3 = R_3/D \tag{24}$$

This would then yield

$$r_1 + r_2 + r_3 = 3 \qquad (25)$$

From Eq.(25), the three non-dimensional parameters r_1, r_2 and r_3 have limits, i.e.,

$$0 < r_1,\ r_2,\ r_3 < 3 \qquad (26)$$

Based on Eqs. (25) and (26), one can establish a design space as shown in Fig. 14(a), in which the triangle ABC is actually the design space of the parallel robot. In Fig. 14(a), the triangle ABC is restricted by r_1, r_2 and r_3. Therefore it can be figured in another form as shown in Fig. 14(b), which is referred to as the planar-closed configuration of the design space. In this design space, each point corresponds a kind of robot with specified value of r_1, r_2 and r_3.

For convenience, two orthogonal coordinates r and t are utilized to express r_1, r_2 and r_3. Thus, by using

$$\begin{cases} r = 2r_1/\sqrt{3} + r_3/\sqrt{3} \\ t = r_3 \end{cases} \qquad (27)$$

coordinates r_1, r_2 and r_3 can be transformed into r and t. Eq. (27) is useful for constructing a performance atlas.

From the analysis of singularity and workspace, we can see that the singular loci and workspace shape of a robot when $r_1 > r_2$ are different from those of the robot when $r_1 < r_2$. For the convenience of analysis, the line $r_1 = r_2$ is used to divide the design space into two regions as shown in Fig. 14(b).

(a)   (b)

Figure 14. Design space of the 2-DOF translational parallel robot

5.3.2 Workspace characteristics

Using the normalization technique in Eqs. (23) and (24), the dimensional parameters R_1, R_2 and R_3 were changed to non-dimensional ones r_1, r_2 and r_3. The kinematic, singularity and workspace analysis results can be obtained by replacing R_n (n=1,2,3) with r_n (n=1,2,3) in Eqs. (2)-(22). Then, using Eq. (21), we can calculate the theoretical workspace area of each robot in the design space shown in Fig. 14(b). As a result, the atlas of the workspace can be plotted as shown in Fig. 15. To plot the atlas, one should first calculate the theoretical workspace area of each non-dimensional robot with r_1, r_2 and r_3, which is included in the design space. Using the Eq. (27), one can then obtain the relationship between the area and the two orthogonal coordinates r and t (see Fig. 14(b)). This relationship is practically used to plot the atlas in the planar system with r and t. The subsequent atlases are also plotted using the same method. Fig. 15 shows not only the relationship between the workspace area and the two orthogonal coordinates but that between the area and the three non-dimensional parameters as well. What we are really most concerned about is the later relationship. For this reason, r and t are not appeared in the figure. From Fig. 15, one can see that

- The theoretical workspace area is inverse proportional to parameter r_3;
- The area atlas is symmetric with respect to $r_1 = r_2$, which means that the area of a kind of robot with $r_1 = u$, $r_2 = w$ ($u,w < 3$) and $r_3 = 3-u-w$ is identical to that of a robot with $r_1 = w$, $r_2 = u$ ($u,w < 3$) and $r_3 = 3-u-w$;
- The area reaches its maximum value when $r_1 = r_2 = 1.5$ and $r_3 = 0$. The maximum value is 9π.

Since the *usable workspace* area is the half of the theoretical workspace area, the atlas of *usable workspace* is identical with that of Fig. 15 in distribution but is different in area value. From Figs. 10 and 15, we can see that the theoretical workspaces of robots $r_1 = u$ and $r_2 = w$, and $r_1 = w$ and $r_2 = u$ are identical with each other not only in area but also in shape. It is noteworthy that, although, the *usable workspace* area atlas is also symmetric about the line $r_1 = r_2$, the *usable workspace* shape of the robot with $r_1 = u$ and $r_2 = w$ is no longer same as that of the robot with $r_1 = w$ and $r_2 = u$. This result is not difficult to be reached from Fig. 13.

Figure 15. Atlas of the theoretical workspace of the parallel robot

5.3.3 Similarity robots

From Fig. 15, one can know the workspace performance of a non-dimensional parallel robot. Our objective is usually the dimensional robot. If the workspace performance of a robot with parameters r_n (n=1,2,3) is clear, one should know the corresponding performance of the robot with parameters R_n (n=1,2,3). Otherwise, the normalization of geometric parameters and the developed design space will be nonsense. Comparing Eqs. (21) and (22), it is not difficult to reach the following relationship

$$S_{tw} = D^2 \, S'_{tw} \text{ and } S_{uw} = D^2 \, S'_{uw} \tag{28}$$

where S'_{tw} and S'_{uw} are the theoretical and usable workspace areas, respectively, of a non-dimensional robot. Eq. (28) indicates that the workspace of a dimensional robot is D^2 times that of a non-dimensional robot. That means, from Fig. 15, one can also know the workspace performance of a dimensional robot.

Therefore, the robot with normalized parameters rn (n=1,2,3) has a generalized significance. The workspace performance of such a robot indicates not only the performance of itself but also those of the robots with parameters Drn, i.e. Rn. Here, the robots with parameters Drn are defined as similarity robots; and the robot with parameters rn is referred to as the basic similarity robot. The analysis in the subsequent sections will show that the similarity robots are similar in terms of not only the workspace performance but also other performances, such as conditioning index and stiffness. For these reasons, the normalization of the geometric parameters can be reasonably applied to the optimal design of the robot. And it also simplifies the optimal design process.

6. Atlases of Good-Condition Indices

From Section 5, one can know characteristics of the workspace, especially the usable workspace of a robot with given r_n or R_n (n=1,2,3). Usually, in the design process and globally evaluation of a performance, a kind of workspace is inevitable. Unfortunately, due to the singularity, neither the theoretical workspace nor the *usable workspace* can be used for these purposes. Therefore, we should define a workspace where each configuration of the robot can be far away from the singularity. As it is well known, the condition number of Jacobian matrix is an index to measure the distance of a configuration to the singularity. The local conditioning index, which is the reciprocal of the condition number, will then be used to define some good-condition indices in this section.

6.1 Local conditioning index

Mathematically, the condition number of a matrix is used in numerical analysis to estimate the error generated in the solution of a linear system of equations by the error in the data (Strang, 1976). The condition number of the Jacobian matrix can be written as

$$\kappa = \left\| J \right\| \left\| J^{-1} \right\| \tag{29}$$

where $\left\| \bullet \right\|$ denotes the Euclidean norm of the matrix, which is defined as

$$\left\| J \right\| = \sqrt{tr\left(J^T W J\right)}; \quad W = \frac{1}{n} I \tag{30}$$

in which n is the dimension of the Jacobian matrix and I the $n \times n$ identity matrix. Moreover, one has

$$1 \leq \kappa \leq \infty \tag{31}$$

and hence, the reciprocal of the condition number, i.e., $1/\kappa$, is always defined as the local conditioning index (LCI) to evaluate the control accuracy, dexterity and isotropy of a robot. This number must be kept as large as possible. If the number can be unity, the matrix is an isotropic one, and the robot is in an isotropic configuration.

6.2 Good-condition workspace

Let's first check how the LCI is at every point in the workspace of the similarity robot with parameters $R_1 = 1.2\text{mm}$, $R_2 = 0.8\text{mm}$ and $R_3 = 0.5\text{mm}$. Its *us-*

able workspace is shown in Fig. 13(a). Fig. 16 shows the distribution of the LCI in the workspace.

Figure 16. Distribution of the LCI in the usable workspace

From Fig. 16 one can see that, in the *usable workspace*, there exist some points where the LCI will be zero or very small. At these points the control accuracy of the robot will be very poor. These points will not be used in practice. They should be excluded in the design process. The left workspace, which will be used in practice, can be referred to as good-condition workspace (GCW) that is bounded by a specified LCI value, i.e., $1/\kappa$. Then, the set of points where the LCI is greater than or equal to (GE) a specified LCI is defined as the GCW.

Using the numerical method, by letting the minimum LCI be 0.3, the GCW area of each basic similarity robot in the design space shown in Fig. 14(b) can be calculated.

The corresponding atlas can be then plotted as shown in Fig. 17, from which one can see that

- The GCW area is inverse proportional to parameter r_3;
- The area atlas is no longer symmetric with respect to the line $r_1 = r_2$. In another sense, this indicates that a large theoretical or usable workspace of a robot doesn't mean that it has a large GCW;
- The maximum value of the GCW area is still that of the robot $r_1 = r_2 = 1.5$ and $r_3 = 0$.

Since there is no singularity within the whole GCW, it can be used as a reference in the definition of a global index, e.g. global conditioning index.

Figure 17. Atlas of the good-condition workspace when LCI $\geq$ 0.3

6.3 Global conditioning index

Jacobian matrix is pose-dependent (see Eq. (17)). Then, the LCI is depended on the pose as well. This indicates that the LCI at one point may be different from that at another point. Therefore, the LCI is a local index. In order to evaluate the global behaviour of a robot on a workspace, a global index can be defined as (Gosselin & Angeles, 1989)

$$\eta_J = \int_W 1/\kappa_J \, dW \Big/ \int_W \, dW \qquad (32)$$

which is the global conditioning index (GCI). In Eq. (32), W is the workspace. In particular, a large value of the index ensures that a robot can be precisely controlled.

For the robot studied here, the workspace W in Eq. (32) can be the GCW when LCI $\geq$ 0.3. The relationship between the GCI and the three normalized parameters r_n (n=1,2,3) can be studied in the design space. The corresponding atlas is shown in Fig. 18, from which one can see that the robots near $r_1 = 1.2$ have large GCI. Some of these robots have very large GCW, some very small.

6.4 Global stiffness index

Disregarding the physical characteristic, kinematically, there will be deformation on the end-effector if an external force acts on it.

Figure 18. Atlas of the global conditioning index

This deformation is dependent on the robot's stiffness and on the external force. The robot stiffness affects the dynamics and position accuracy of the device, for which stiffness is an important performance index. The static stiffness (or rigidity) of the robot can be a primary consideration in the design of a parallel robot for certain applications.

Equation (8) can be rewritten as

$$\dot{q} = J\dot{p} \tag{33}$$

On the other hand, by virtue of what is called the duality of kinematics and statics (Waldron & Hunt, 1988), the forces and moments applied at the end-effector under static conditions are related to the forces or moments required at the actuators to maintain the equilibrium by the transpose of the Jacobian matrix J. We can write

$$\tau = J^{\mathrm{T}} f \tag{34}$$

where f is the vector of actuator forces or torques, and τ is the generalized vector of Cartesian forces and torques at the end-effector.
In the joint coordinate space, a diagonal stiffness matrix K_p is defined to express the relationship between the actuator forces or torques f and the joint displacement vector Δq according to

$$f = K_p \Delta q \tag{35}$$

With

$$K_p = \begin{bmatrix} k_{p1} & \\ & k_{p2} \end{bmatrix} \tag{36}$$

in which k_{pi} is a scalar representing the stiffness of each of the actuators.

In the operational coordinate space, we define a stiffness matrix K which relates the external force vector τ to the output displacement vector D of the end-effector according to

$$\tau = KD \tag{37}$$

The Eq. (33) also describes the relationship between the joint displacement vector Δq and the output displacement vector D, i.e.,

$$\Delta q = JD \tag{38}$$

From Eqs. (34), (35) and (38), we get

$$\tau = J^T K_p J D \tag{39}$$

Thus, the stiffness matrix K is expressed as

$$K = J^T K_p J \tag{40}$$

Then, we have

$$D = K^{-1} \tau \tag{41}$$

From Eq. (41), one can write

$$D^T D = \tau^T \left(K^{-1} \right)^T K^{-1} \tau \tag{42}$$

Let the external force vector τ be unit, i.e.,

$$\|\tau\|^2 = \tau^T \tau = 1 \tag{43}$$

Under the condition (43), one can derive the extremum of the norm of vector D. In order to obtain the conditional extremum, using the Lagrange multiplier λ_D, one can construct the Lagrange equation as following

$$L_D = \boldsymbol{\tau}^{\mathrm{T}}\left(\boldsymbol{K}^{-1}\right)^{\mathrm{T}}\boldsymbol{K}^{-1}\boldsymbol{\tau} - \lambda_D\ (\boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{\tau} - 1) \tag{44}$$

The necessary condition to the conditional extremum is

$$\frac{\partial L_D}{\partial \lambda_D} = 0: \ \boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{\tau} - 1 = 0, \text{ and } \frac{\partial L_D}{\partial \boldsymbol{\tau}} = 0: \ \left(\boldsymbol{K}^{-1}\right)^{\mathrm{T}}\boldsymbol{K}^{-1}\boldsymbol{\tau} - \lambda_D\ \boldsymbol{\tau} = 0 \tag{45}$$

from which one can see that the Lagrange multiplier λ_D is actually an eigenvalue of the matrix $\left(\boldsymbol{K}^{-1}\right)^{\mathrm{T}}\boldsymbol{K}^{-1}$. Then, the norm of vector D can be written as

$$\|\boldsymbol{D}\|^2 = \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D} = \boldsymbol{\tau}^{\mathrm{T}}\left(\boldsymbol{K}^{-1}\right)^{\mathrm{T}}\boldsymbol{K}^{-1}\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{T}}\ \lambda_D\ \boldsymbol{\tau} = \lambda_D \tag{46}$$

Therefore, the extremum of $\|\boldsymbol{D}\|^2$ is the extremum of the eigenvalues of the matrix $\left(\boldsymbol{K}^{-1}\right)^{\mathrm{T}}\boldsymbol{K}^{-1}$. Then, if $k_{p1} = k_{p2} = 1$ and $\|\boldsymbol{\tau}\|^2 = 1$, the maximum and minimum deformations on the end-effector can be described as

$$\|\boldsymbol{D}_{\max}\| = \sqrt{\max\left(|\lambda_{Di}|\right)} \text{ and } \|\boldsymbol{D}_{\min}\| = \sqrt{\min\left(|\lambda_{Di}|\right)} \tag{47}$$

where λ_{Di} ($i = 1, 2$) are the eigenvalues of the matrix $\left(\boldsymbol{K}^{-1}\right)^{\mathrm{T}}\boldsymbol{K}^{-1}$. $\|\boldsymbol{D}_{\max}\|$ and $\|\boldsymbol{D}_{\min}\|$ are actually the maximum and minimum deformations on the end-effector when both the external force vector and the matrix $\boldsymbol{K}_p$ are unity. The maximum and minimum deformations form a deformation ellipsoid, whose axes lie in the directions of the eigenvectors of the matrix $\left(\boldsymbol{K}^{-1}\right)^{\mathrm{T}}\boldsymbol{K}^{-1}$. Its magnitudes are the maximum and minimum deformations given by Eq. (47). The maximum deformation $\|\boldsymbol{D}_{\max}\|$, which can be used to evaluate the stiffness of the robot, is defined as the local stiffness index (LSI). The smaller the deformation is, the better the stiffness is.

Similarly, based on Eq. (47), the global stiffness index (GSI) that can evaluate the stiffness of a robot within the workspace is defined as

$$\eta_{D\max} = \frac{\int_W \|\boldsymbol{D}_{\max}\| dW}{\int_W dW} \tag{48}$$

where, for the robot studied here, W is the GCW when $LCI \geq 0.3$. Usually, $\eta_{D\max}$ can be used as the criterion to design the robot with respect to its stiffness. Normally, we expect that the index value should be as small as possible. Figure 19 shows the atlas of $\eta_{D\max}$, from which one can see that the larger the parameter r_3, the smaller the deformation. That means the stiffness is proportional to the parameter r_3.

Figure 19. Atlas of the global stiffness index

7. Optimal Design based on the Atlas

In this section, a method for the optimal kinematic design of the parallel robot will be proposed based on the results of last sections.

7.1 Optimum region with respect to desired performances

Relationships between performance indices and the link lengths of the 2-DOF translational parallel robot have been studied. The results have been illustrated by their atlases, from which one knows visually which kind of robot can be with a better performance and which cannot. This is very important for us to find out a global optimum robot for a specified application. In this section, the optimum region will be shown first with respect to possible performances.

7.1.1 Workspace and GCI

In almost all designs, the workspace and GCI are usually considered. From the atlas of the GCW (see the Fig. 17), we can see that the workspace of a robot when r_1 is near 1.5 and r_3 is shorter can be larger. From the atlas of GCI (Fig. 18), we know that robots near $r_1 = 1.2$ have better GCI. If the GCW area, denoted as S'_{GCW}, is supposed to be greater than 6 ($S'_{GCW} > 6$) and the GCI greater than 0.54, the optimum region in the design space can be obtained shown as the shaded region in Fig. 20(a). The region is denoted as $\Omega_{GCW-GCI} = \left[(r_1,\ r_2,\ r_3) \mid S'_{GCW} > 6 \text{ and } \eta_J > 0.54\right]$ with performance restriction. One can also obtain an optimum region with better workspace and GCI, for example, the region $\Omega'_{GCW-GCI}$ where $S'_{GCW} > 7$ and $\eta_J > 0.57$ as shown in Fig. 20(b). In order to get a better result, one can decrease the optimum region with stricter restriction. Such a region contains some *basic similarity robots*, which are all possible optimal results.

Figure 20. Two optimum region examples with respect to both GCI and GCW performance restrictions

After the optimum region is identified, there are two ways to achieve the optimal design result with non-dimensional parameters. One is to search a most optimal result within the region $\Omega_{GCW-GCI}$ or $\Omega'_{GCW-GCI}$ using one classical searching algorithm based on an established object function. The method will yield a unique solution. This is not the content of this paper. Another one is to select a robot within the obtained optimum region. For example, the *basic similarity robot* with $r_1 = 1.2$, $r_2 = 1.65$ and $r_3 = 0.15$ can be selected as the candidate if only workspace and GCI are involved in the design. Its GCW area and the

GCI value are 7.2879 and 0.5737, respectively. The robot with only r_n (n=1,2,3) parameters and its GCW are shown in Fig. 21.

Figure 21. The robot with parameters $r_1 = 1.2$, $r_2 = 1.65$ and $r_3 = 0.15$ in the $\Omega'_{GCW-GCI}$ region and its GCW when LCI ≥ 0.3

Actually, we don't recommend the former method for achieving an optimal result. The solution based on the objective function approach is a mathematical result, which is unique. Such a result is maybe not the optimal solution in practice. Practically, we usually desire a solution subjecting to our application conditions. From this view, it is unreasonable to provide a unique solution for the optimal design of a robot. Since we cannot predict any application condition previously, it is most ideally to provide all possible optimal solutions, which allows a designer to adjust the link lengths with respect to his own design condition. The advantage of the later method is just such an approach that allows the designer to adjust the design result fitly by trying to select another candidate in the optimum region.

7.1.2 Workspace, GCI, and GSI

In this paper, stiffness is evaluated by the maximum deformation of the end-effector when the external force and the stiffness of each of the actuators are unit. A robot with smaller $\eta_{D\max}$ value usually has better stiffness. Since accuracy is inherently related to the stiffness, actually, the stiffness index used here can also evaluate the accuracy of the robot. To achieve an optimum region with respect to all of the three indices, the GCW can be specified as $S'_{GCW} > 6$, GCI $\eta_J > 0.54$ and GSI $\eta_{D\max} < 7.0$. The optimal region will be $\Omega_{GCW-GCI-GSI} = \left[(r_1,\ r_2,\ r_3)\,|\,S'_{GCW} > 6,\ \eta_J > 0.54,\ \text{and}\ \eta_{D\max} < 7\right]$ shown in Fig. 22. For

example, the values of the GCW, GCI and GSI of the *basic similarity robot* with parameters $r_1 = 1.12$, $r_2 = 1.68$ and $r_3 = 0.2$ in the optimum region are $S'_{GCW} = 6.8648$, $\eta_J > 0.5753$ and $\eta_{D\max} = 6.5482$. Fig. 23 shows the robot and its GCW when LCI is GE 0.3.

Figure 22. One optimum region example with respect to the GCI, GCW and GSI performance restrictions

Figure 23. The robot with parameters $r_1 = 1.12$, $r_2 = 1.68$ and $r_3 = 0.2$ in the $\Omega_{GCW-GCI-GSI}$ region and its GCW when LCI≥ 0.3

7.2 Dimension determination based on the obtained optimum example

The final objective of optimum design is determining the link lengths of a robot, i.e. the *similarity robot*. In the last section, some optimum regions have been presented as examples. These regions consist of *basic similarity robots* with non-dimensional parameters. The selected optimal *basic similarity robots* are comparative results, not final results. Their workspaces may be too small to be used in practice. In this section, the dimension of an optimal robot will be determined with respect to a desired workspace.

As an example of presenting how to determine the *similarity robot* with respect to the optimal *basic similarity robot* obtained in section 7.1, we consider the robot with parameters $r_1 = 1.12$, $r_2 = 1.68$ and $r_3 = 0.2$ selected in section 7.1.2. The robot is from the optimum region $\Omega_{GCW-GCI-GSI}$, where the workspace, GCI and stiffness are all involved in the design objective. To improve the GCI and GSI performances of the robot, letting LCI be GE 0.5, the values of the GCW, GCI and GSI of the robot with parameters $r_1 = 1.12$, $r_2 = 1.68$ and $r_3 = 0.2$ are $S'_{GCW} = 4.0735$, $\eta_J > 0.6977$ and $\eta_{D\max} = 2.5373$. Fig. 24 shows the revised GCW. Comparing Figs. 23 and 24, it is obvious that the improvement of performances GCI and GSI is based on the sacrifice of the workspace area.

Figure 24. GCW of the robot with parameters $r_1 = 1.12$, $r_2 = 1.68$ and $r_3 = 0.2$ when $LCI \ge 0.5$

The process to find the dimensions with respect to a desired practical workspace can be summarized as following:

Step 1: Investigating the distribution of LCI and LSI on the GCW of the *basic similarity robot*. For the aforementioned example, the distribution is

shown in Fig. 25 (a) and (b), respectively, from which one can see the distributing characteristics of the two performances. The investigation can help us determining whether it is necessary to adjust the GCW. For example, if the stiffness at the worst region of the GCW cannot satisfy the specification on stiffness, one can increase the specified LCI value to reduce the GCW. In contrary, if the stiffness is permissible, one can decrease the specified LCI value to increase the GCW.

(a)

(b)

Figure 25. Distribution of LCI and LSI in the GCW of the *basic similarity robot* when LCI≥0.5: (a) LCI; (b) LSI

Step 2: Determining the factor D, which was used to normalize the parameters of a dimensional robot to those that are non-dimensional. The GCW area when LCI≥0.5 of the selected *basic similarity robot* is

$S'_{GCW} = 4.0735$. If the desired workspace area S_{GCW} of a dimensional robot is given with respect to the design specification, the factor D can be obtained as $D = \sqrt{S_{GCW}/S'_{GCW}}$, which is identical with the relationship in Eq. (28). For example, if the desired workspace shape is similar to the GCW shown in Fig. 24 and its area $S_{GCW} = 500\text{mm}$, there is $D = \sqrt{S_{GCW}/S'_{GCW}} = \sqrt{500/4.0735} \approx 11.08\text{mm}$.

Step 3: Achieving the corresponding *similarity robot* by means of dimensional factor D. As given in Eq. (24), the relationship between a dimensional parameter and a non-dimensional one is $R_n = Dr_n$ (n=1,2,3). Then, if D is determined, R_n can be obtained. For the above example, there are $R_1 = 12.41\text{mm}$, $R_2 = 18.61\text{mm}$ and $R_3 = 2.22\text{mm}$. In this step, one can also check the performances of the *similarity robot*. For example, Fig. 26 (a) shows the distribution of LCI on the desired workspace, from which one can see that the distribution is the same as that shown in Fig. 25 (a) of the *basic similarity robot*. The GCI is still equal to 0.6977. Fig. 26 (b) illustrates the distribution of LSI on the workspace. Comparing Fig. 26 (b) with Fig. 25 (b), one can see that the distributions of LSI are the same. The GSI value is still equal to 2.5373. Then, the factor D does not change the GCI, GSI, and the distributions of LCI and LSI on the workspaces. For such a reason, we can say that, if a *basic similarity robot* is optimal, any one of its *similarity robots* is optimal.

Step 4: Determining the parameters L_n (n=1,2,3) that are relative to the leg 2. Since the parameters are not enclosed in the Jacobian matrix, they are not the optimized objects. They can be determined with respect to the desired workspace. Strictly speaking, the workspace analyzed in the former sections is that of the leg 1. As mentioned in section 5.1, to maximize the workspace of the 2-DOF parallel translational robot and, at the same time, to reduce the cost, the parameters L_n (n=1,2,3) should be designed as those with which the workspace of leg 2 can just embody the workspace of the leg 1. To this end, the parameters should be subject to the following equations

$$Y_{max} = L_1 + L_2 - L_3 + R_3 \tag{49}$$

$$Y_{min} = L_1 - L_2 - L_3 + R_3 \tag{50}$$

in which Y_{max} and Y_{min} are *y*-coordinates of the topmost and lowest points of the desired workspace. For the desired GCW shown in Fig. 26, there are $Y_{max} = \text{-3.32mm}$ and $Y_{min} = \text{-29.92mm}$. Substituting them

in Eqs. (49) and (50), we have $L_2 = 13.30\text{mm}$. To reduce the manufacturing cost, let $L_1 = L_2$, which leads to $L_3 = 32.14\text{mm}$

Step 5: Calculating the input limit for each actuator. The range of each input parameter can be calculated from the inverse kinematics. For the obtained *similarity robot*, there are $\theta \in [-83.3040°, \ 81.7649°]$ and $s \in [-6.10\text{mm}, \ 25.49\text{mm}]$.

Then, the parameters of the optimal robot with respect to the desired workspace $S_{GCW} = 500\text{mm}$ are $R_1 = 12.41\text{mm}$, $R_2 = 18.61\text{mm}$, $R_3 = 2.22\text{mm}$, $L_1 = L_2 = 13.30\text{mm}$, $L_3 = 32.14\text{mm}$, $\theta \in [-83.3040°, \ 81.7649°]$ and $s \in [-6.10\text{mm}, \ 25.49\text{mm}]$. It is noteworthy that this result is only one of all possible solutions. If the designer picks up another *basic similarity robot* from the optimum region, the final result will be different. This is actually one of the advantages of this optimal design method. The designer can adjust the final result to fit his design condition. It is also worth notice that, actually, the desired workspace shape cannot be that shown in Fig. 26. It is usually in a regular shape, for example, a circle, a square or a rectangle. In this case, a corresponding similar workspace should be first identified in the GCW of the *basic similarity robot* in **Step 2**. This workspace, which is the subset of the GCW, is normally just embodied by the GCW. The identified workspace area will be used to determine the factor D with respect the desired workspace area in **Step 2**.
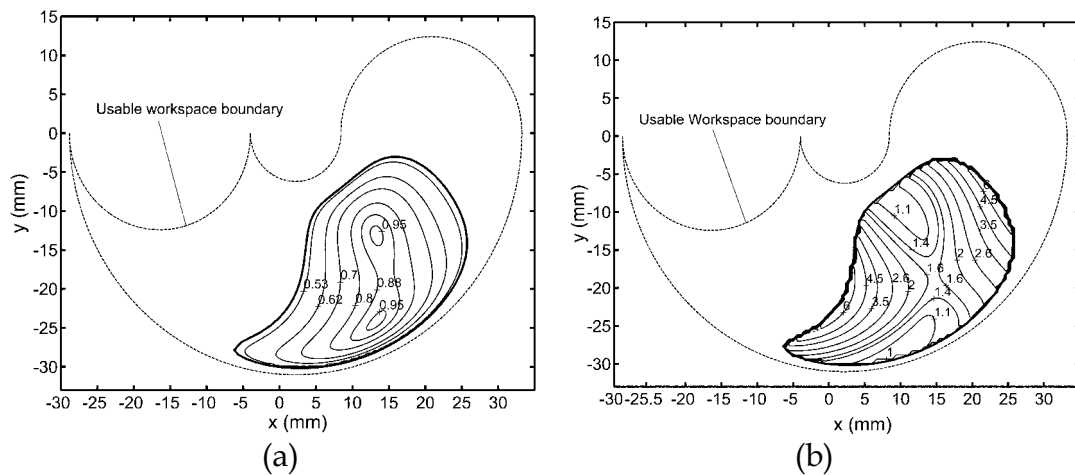
Figure 26. Distribution of LCI and LSI in the desired workspace of the obtained *similarity robot*: (a) LCI; (b) LSI

8. Conclusion and Future Works

In this chapter, a novel 2-DoF translational robot is proposed. One advantage of the robot is that it can position a rigid body in a 2D plane while maintaining a constant orientation. The proposed robot can be used in light industry where high speed is needed. The inverse and forward kinematics problems, workspace, conditioning indices, and singularity are presented here. In particular, the optimal kinematic design of the robot is investigated and a design method is proposed.

The key issue of this design method is the construction of a geometric design space based on the geometric parameters involved, which can embody all *basic similarity robots*. Then, atlases of desired indices can be plotted. These atlases can be used to identify an optimal region, from which an ideal candidate can be selected. The real-dimensional parameters of a *similarity robot* can be found by considering the desired workspace and the *good-condition workspace* of the selected *basic similarity robot*. Compared with other design methods, the proposed methodology has some advantages: (a) one performance criterion corresponds to one atlas, which can show visually and globally the relationship between the index and design parameters; (b) for the same reason in (a), the fact that some performance criteria are antagonistic is no longer a problem in the design; (c) the optimal design process can consider multi-objective functions or multi-criteria, and also guarantees the optimality of the result; and finally, (d) the method provides not just one solution but all possible solutions.

The future work will focus on the development of the computer-aided design of the robot based on the proposed design methodology, the development of the robot prototype, and the experience research of the prototype.

Acknowledgement

9. References

Asada, H. and Kanade, T. (1983). Desgin of direct-drive mechanical arms, *ASME Journal of Vibration, Acoustics, Stress, and Reliability in Design,* Vol.105, pp.312-316.

Bonev, I. (2001). The Delta parallel robot-the story of success, http://www.parallelmic. org/Reviews/Review002p.html.

Carricato, M. and Parenti-Castelli, V. (2001). A family of 3-DOF translational parallel manipulators, *Proceedings of the 2001 ASME Design Engineering*

Technical Conferences, Pittsburgh, Pennsylvania, paper DETC2001/DAC-21035.

Cervantes-Sánchez, J.J., Hernández-Rodríguez, J.C. and Angeles, J. (2001) On the kinematic design of the 5R planar, symmetric manipulator, *Mechanism and Machine Theory*, Vol.36, pp.1301-1313

Chablat, D. & Wenger, P. (2003). Architecture optimization of a 3-DoF parallel mechanism for machining applications: the Orthoglide, IEEE Transactions on Robotics and Automation, Vol. 19, pp.403–410

Clavel, R. (1988). DELTA: a fast robot with parallel geometry, *Proceedings of 18th Int. Symp. on Industrial Robot*, pp. 91-100.

Gao, F., Liu, X.-J. and Gruver, W.A. (1998). Performance evaluation of two degrees of freedom planar parallel robots, *Mechanisms and Machine Theory*, Vol.33, pp.661-668.

Gosselin, C. & Angeles, J. (1989). The optimum kinematic design of a spherical three-degree-of-freedom parallel manipulator, *J. Mech. Transm. Autom. Des.*, Vol.111, pp.202-207

Gosselin, C.M. & Angeles, J. (1990). Singularity analysis of closed loop kinematic chains, *IEEE Trans. on Robotics and Automation,* Vol.6, pp.281-290.

Gough, V. E. (1956). Contribution to discussion of papers on research in automobile stability, control and tyre performance, *Proceedings of Auto Div Inst Mech Eng*, pp.392-395.

Hervé, J. M. (1992). Group mathematics and parallel link mechanisms, *Proceedings of IMACS/SICE Int. Symp. On Robotics, Mechatronics, and Manufacturing Systems*, pp.459-464.

Hunt, K. H. (1978). *Kinematic geometry of mechanisms*, Clarendon Press, Oxford.

Kim, H.S. & Tsai, L.-W. (2002). Design optimization of a Cartesian parallel manipulator, *Proceedings of ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Montreal, Canada, paper DETC2002/MECH-34301

Kong, X. & Gosselin, C.M. (2002). Kinematics and singularity analysis of a novel type of 3-CRR 3-DOF translational parallel manipulator, *International Journal of Robotics Research*, Vol.21, pp.791-798.

Liu, X.-J. (2001). *Mechanical and kinematics design of parallel robotic mechanisms with less than six degrees of freedom*, Post-Doctoral Research Report, Tsinghua University, Beijing.

Liu, X.-J. & Kim, J. (2005). A new spatial three-DoF parallel manipulator with high rotational capability, *IEEE/ASME Transactions on Mechatronics*, Vol.10, No.5, pp.502-512.

Liu, X.-J. & Wang, J. (2003). Some new parallel mechanisms containing the planar four-bar parallelogram, *International Journal of Robotics Research*, Vol.22, No.9, pp.717-732

Liu, X.-J., Jeong, J., & Kim, J. (2003). A three translational DoFs parallel cube-manipulator, *Robotica*, Vol.21, No.6, pp.645-653.

Liu, X.-J., Kim, J. and Wang, J. (2002). Two novel parallel mechanisms with less than six DOFs and the applications, *Proceedings of Workshop on Fundamental Issues and Future Research Directions for Parallel Mechanisms and Manipulators*, pp. 172-177, Quebec City, QC, Canada, October, 2002.

Liu, X.-J., Wang, J., Gao F., & Wang, L.-P. (2001). On the analysis of a new spatial three degrees of freedom parallel manipulator, *IEEE Transactions on Robotics and Automation*, Vol.17, pp.959-968.

Liu, X.-J., Wang, Q.-M., & Wang, J. (2005). Kinematics, dynamics and dimensional synthesis of a novel 2-DOF translational manipulator, *Journal of Intelligent & Robotic Systems*, Vol.41, No.4, pp.205-224.

McCloy, D. (1990). Some comparisons of serial-driven and parallel driven mechanisms, *Robotica*, Vol.8, pp.355-362.

Merlet, J.-P. (2000). *Parallel robots*, Kluwer Academic Publishers, London.

Ottaviano, E. & Ceccarelli, M. (2002). Optimal design of CaPaMan (Cassino Parallel Manipulator) with a specified orientation workspace, *Robotica*, Vol.20, pp.159–166

Siciliano, B. (1999). The Tricept robot: inverse kinematics, manipulability analysis and closed- loop direct kinematics algorithm, *Robotica,* Vol.17, pp.437-445.

Stewart, D. (1965). A platform with six degrees of freedom, *Proc Inst Mech Eng*, Vol.180, pp.371-386,

Stock, M. & Miller, K. (2004). Optimal kinematic design of spatial parallel manipulators: application to linear Delta robot, *Journal of Mechanical Design*, Vol.125, pp.292-301

Strang, G. (1976). *Linear algebra and its application*, Academic Press, New York

Tonshoff, H.K., Grendel, H. and Kaak, R. (1999). Structure and characteristics of the hybrid manipulator Georg V, In: *Parallel Kinematic Machines*, C.R. Boer, L. Molinari- Tosatti and K.S. Smith (editors), pp.365-376, Springer-Verlag London Limited.

Tsai, L. W. & Stamper, R. (1996). A parallel manipulator with only translational degrees of freedom, *Proceedings of ASME 1996 Design Engineering Technical Conference*, Irvine, CA, paper 96-DETC-MECH -1152.

Waldron, K.J. & Hunt, K.H. (1988). Series-parallel dualities in actively coordinated mechanisms, *Robotics Research*, Vol.4, pp.175-181.

Zhao, T. S. & Huang, Z. (2000). A novel three-DOF translational platform mechanism and its kinematics, *Proceedings of ASME 2000 International Design Engineering Technical Conferences*, Baltimore, Maryland, paper DETC2000/MECH-14101.

Industrial and Mobile Robot Collision–Free Motion Planning Using Fuzzy Logic Algorithms

Tzafestas S.G. and Zavlangas P.

1. Introduction

Motion planning is a primary task in robot operation, where the objective is to determine collision-free paths for a robot that works in an environment that contains some moving obstacles (Latombe, 1991; Fugimura, 1991; Tzafestas, 1999). A moving obstacle may be a rigid object, or an object with joints such as an industrial manipulator. In a persistently changing and partially unpredictable environment, robot motion planning must be on line. The planner receives continuous flow of information about occurring events and generates new commands while previous planned motions are being executed. Off – line robot motion planning is a one – shot computation prior to the execution of any motion, and requires all pertinent data to be available in advance. With an automatic motion planner and appropriate sensing devices, robots can adapt quickly to unexpected changes in the environment and be tolerant to modeling errors of the workspace. A basic feature of intelligent robotic systems is the ability to perform autonomously a multitude of tasks without complete a priori information, while adapting to continuous changes in the working environment.

Clearly, both robotic manipulators and mobile robots (as well their combination, i.e. mobile manipulators (Seraji, 1998; Tzafestas & Tzafestas, 2001)) need proper motion planning algorithms. For the robotic manipulators, motion planning is a critical aspect due to the fact that the end effector paths have always some form of task constraints. For example, in arc welding the torch may have to follow a complex 3-dimensional path during the welding process. Specifying manually such paths can be tedious and time consuming. For the mobile robots (indoor and outdoor robots) motion planning and autonomous navigation is also a critical issue, as evidenced by applications such as office cleaning, cargo delivery, autonomous wheel chairs for the disabled,etc.

Our purpose in this chapter is to present a solution of the motion planner design problem using fuzzy logic and fuzzy reasoning. Firstly, the case of industrial robotic manipulators is considered, and then the class of mobile robots is treated. The methodology adopted is primarily based on some recent results

derived by the authors (Moustris & Tzafestas, 2005; Zavlangas & Tzafestas, 2000; 2001; 2002). To help the reader appreciate the importance of the techniques presented in the chapter, a short review is first included concerning the general robot motion planning problem along with the basic concepts and some recent results. A good promising practical approach is to use fuzzy logic along the path of behavior–based system design which employs Brooks' subsumption architecture (Brooks, 1986; Izumi & Watanabe, 2000; Topalov & Tzafestas, 2001; Watanabe et al., 1996; Watanabe et al., 2005).

Section 2 provides the overview of robot motion planning for industrial and mobile robots. Section 3 presents the authors' technique for industrial manipulators' fuzzy path planning and navigation. Section 4 extends this technique to mobile robots and discusses the integration of global and local path planning and navigation. Global path planning uses topological maps for representing the robot's environment at the global level, in conjunction with the potential field method. Section 5 presents a representative set of experimental results for the SCARA Adept 1 robotic manipulator and the Robuter III mobile robot (which can be equipped with a robotic arm). Results for both local and global path planning / navigation are included for the Robuter III robot. Finally, a general discussion on the results of the present technique is provided, along with some indications for future directions of research.

2. Robot Motion Planning : An Overview

2.1 Review of Basic Motion Planning Concepts

Robot motion planning techniques have received a great deal of attention over the last twenty years. It can roughly be divided into two categories : *global and local*. Most of the research in global techniques has been focused *on off-line planning* in static environments. A plan is then computed as a geometric path. An important concept developed by this research is the *Configuration space* or *C-space* of a robot (Latombe, 1991; Lozano-Perez, 1983).

The global techniques, such as *road map* (Latombe, 1991), cell *decomposition* (Latombe, 1991) and *potential fields* methods (Khatib, 1986), generally assume that a complete model of the robot's environment is available.

The *roadmap approach* to path planning consists of capturing the connectivity of the robot's free space in a network of one-dimensional curves (called the roadmap), lying in the free space. Once the roadmap has been constructed, it is used as a set of standardized paths. Path planning is thus reduced to connecting the initial and goal configuration to points in the roadmap and searching it for a path between these points (Latombe, 1991).

Cell decomposition methods are perhaps the motion planning methods that have been most extensively studied so far (Latombe, 1991). They consist of decomposing the robot's free space into simple regions, called *cells*, such that a path between any two configurations in a cell can be easily generated. A nondirected graph representing the adjacency relation between the cells is then constructed and searched. This graph is called the *connectivity graph*. Its nodes are the cells extracted from the free space and two nodes are connected by a link if only the corresponding cells are adjacent. The outcome of the search is a sequence of cells called a *channel*. A continuous free path can be computed from this sequence (Latombe, 1991). A straightforward approach to motion planning is to discretize the *configuration space* into a fine regular grid of configurations and to search this grid for a free space. This approach requires powerful heuristics to guide the search. Several types of heuristics have been proposed. The most successful ones take the form of functions that are interpreted as *potential fields* (Latombe, 1991). The robot is represented as a point in configuration space, moving under the influence of an artificial potential produced by the goal configuration and the C-obstacles. Typically, the goal configuration generates an *"attractive potential"* which pulls the robot towards the goal, and the C-obstacles produce a *"repulsive potential"* which pushes the robot away from them. The generated gradient of the total potential is treated as an artificial force applied to the robot. At every configuration, the direction of this force is considered to be the most promising direction of motion.

The advantage of global approaches lies in the fact that a complete trajectory from the starting point to the target point can be computed *off-line*. However, global approaches are not appropriate for fast obstacle avoidance. Their strength is *global path planning*. Additionally, these methods were proven problematic when the global world model is inaccurate, or simply not available, as it is typically the case in the most populated environments. Some researchers have shown how to update global world models based on sensory inputs, using probabilistic representations. A second disadvantage of global methods is their low speed due to the inherent complexity of robot motion planning. This is particularly the case if the underlying world model changes with time, because of the resulting requirement for repeated adjustments of the global plan. In such cases, planning using a global model is usually too expensive to be done repeatedly.

Local approaches, on the other hand, use only a small fraction of the world model to generate robot control. This comes at the obvious disadvantage that they cannot produce optimal solutions. Local approaches are easily trapped at local minima. However, the key advantage of local techniques over global ones lies in their low computational complexity, which is particularly important when the world model is updated frequently based on sensor information. For example, potential field methods determine the next step by assuming that obstacles assert negative forces on the robot, and that the target location asserts a

positive force. These methods are extremely fast, and they typically consider only a small subset of obstacles close to the robot. However, such methods have often failed to find trajectories between closely spaced obstacles; they also can produce oscillatory behaviour in narrow spaces.

2.2 Motion Planning of Mobile Robots

To be useful in the real world, mobile robots need to move safely in unstructured environments and achieve their given goals despite unexpected changes in their surroundings. The environments of real robots are rarely predictable or perfectly known so it does not make sense to make precise plans before moving. The robot navigation problem can be decomposed into the following two problems (Ratering & Gini, 1995) :

- *Getting to the goal.* This is a global problem because short paths to the goal generally cannot be found using only local information. The topology of the space is important in finding good routes to the goal.
- *Avoiding obstacles.* This can often be solved using only local information, but for an unpredictable environment it cannot be solved in advance because the robot needs to sense the obstacles before it can be expected to avoid them.

Over the years, robot collision avoidance has been a component of high-level controls in hierarchical robot systems. Collision avoidance has been treated as a planning problem, and research in this area was focused on the development of collision-free path planning algorithms. These algorithms aim at providing the low-level control with a path that will enable the robot to accomplish its assigned task free from any risk of collision. However, this places limits on the robot's real-time capabilities for precise, fast, and highly interactive operations in a cluttered and evolving environment. Collision avoidance at the low-level control is not intended to replace high-level functions or to solve planning problems. The purpose is to make better use of low-level control capabilities in performing real-time operations. A number of different architectures for autonomous robot navigation have been proposed in the last twenty years (Latombe, 1991; Fugimura, 1991; Tzafestas, 1999). These include hierarchical architectures that partition the robot's functionalities into high-level (model and plan) and low-level (sense and execute) layers; behaviour – based architectures that achieve complex behaviour by combining several simple behaviour-producing units; and hybrid architectures that combine a layered organization with a behaviour-based decomposition of the execution layer (see e.g., (Izumi & Watanabe, 2000; Watanabe et al., 1996; Topalov & Tzafestas, 2001; Watanabe et al., 2005; Lozano-Perez, 1983; Khatib, 1986; Ratering & Gini, 1995; Erdmann & Lozano-Perez, 1987; Griswold & Elan, 1990; Gil de Lamadrid & Gini, 1990;

Fibry, 1987; Gat, 1991; Sugeno & Nishida, 1985; Yen & Pflunger, 1992)). While the use of hybrid architectures is gaining increasing consensus in the field, a number of technological gaps still remain.

As mentioned in Section 2.1, the classical approaches can be mainly divided into two categories : *global path planning* and *local navigation* (Latombe, 1991). The global approaches preassume that a complete representation of the configuration space has been computed before looking for a path. They are complete in the sense that if a path exists it will be found. Unfortunately, computing the complete configuration space is very time consuming, worst, the complexity of this task grows exponentially as the number of degrees of freedom increases. Consequently, today most of the robot path planners are used off-line. The planner is equipped with a model of the environment and produces a path which is passed to the robot controller for execution. In general, the time necessary to achieve this, is not short enough to allow the robot to move in dynamic environments. The local approaches need only partial knowledge of the robot's workspace. The decisions to move the robot are taken using local criteria and heuristics to choose the most promising direction. Consequently, the local methods are much faster. Unfortunately, they are not complete, it may happen that a solution exists and cannot be found. The local approaches consider planning as an optimization problem, where finding a path to the goal configuration corresponds to the optimization of some given function. As an optimization technique, the local approaches are subject to get trapped in some local minima, where a path to the goal has not been found and from which it is impossible or, at least, very difficult to escape.

From the above, it is very clear, that both global and local techniques have many advantages, as well as important disadvantages. The output of a global path planner is a continuous path along which the robot will not collide with obstacles. However, any model of the real world will be incomplete and inaccurate, thus collisions may still occur if the robot moves blindly along such a path. One conventional application is for the robot to track the global path. More recently, work has been done on increasing the level of competence, by including real-time collision avoidance capabilities. Such local or reactive behaviours operate in real time but cannot solve the global problem of moving to an arbitrary goal. It is very clear that to built a complete system, the above approaches must be combined. A path planner must provide the robot with a global path to the goal. A local controller then moves the robot along the global path while handling small changes in the environment and unexpected or moving obstacles.

Some researchers have solved the navigation problem by solving these two sub-problems one after the other. A path is first found from the robot's initial position to the goal and then the robot approximates this path as it avoids obstacles. This method is restrictive in that the robot is required to stay fairly close to or perhaps on a given path. This would not work well if the path goes

through a passageway which turns out to be blocked by an unforeseen obstacle. Solutions that are only local or reactive (Brooks, 1986) can lead the robot into local minima traps. Solutions that assume a priori knowledge of the position of the obstacles (e.g. (Fugimura, 1991; Erdmann & Lozano-Perez, 1987)), or select a path using only information on stationary obstacles, and determine the speed of the robot while following the path (e.g. (Griswold & Elan, 1990)), or solutions that require the robot to stay within some distance from its assigned path, while avoiding unknown moving obstacles (e.g., (Gil de Lamadrid & Gini, 1990)), are not always sufficiently flexible to deal with situations in which an obstacle blocks a path to the goal.

In the general case, knowledge of the environment is partial and approximate; sensing is noisy; the dynamics of the environment can only be partially predicted; and robot's hardware execution is not completely reliable. Though, the robot needs to make decisions and execute actions at the time-scale of the environment. Classical planning approaches have been criticized for not being able to adequately cope with this situation, and a number of *reactive approaches* to robot control have been proposed (e.g. (Fibry, 1987; Gat, 1991)), including the use of fuzzy control techniques (e.g., (Martinez et al., 1994; Seraji & Howard, 2002; Sugeno & Nishida, 1985; Yen & Pflunger, 1992)). Reactivity provides immediate response to unpredicted environmental situations by giving up the idea of reasoning about future consequences of actions. Reasoning about future consequences (sometimes called *"strategic planning"*), however, is still needed in order to intelligently solve complex tasks.

Some recent developments in mobile robot navigation using fuzzy logic algorithms include (Parhi, 2005) and (El Hajjaji, 2004). In (Parhi, 2005) the fuzzy controller enables the robot to avoid obstacles that are not mobile robots. The fuzzy rules steer the robot according to whether there are obstacles or targets around it and how far they are from it. Fuzzy logic is suitable for this problem because this information is usually not precisely known. In (El Hajjaji, 2004) the case of 4-wheel automotive vehicles is considered which are modeled by a Takagi-Sugeno type of model. The fuzzy controller is then designed to improve the stability of the vehicle. A comprehensive study of the kinematics of nonholonomic mobile manipulators composed by an n_a-joint robotic arm and a nonholonomic mobile platform having two independently driven wheels is provided in (Bayle et al., 2003). Finally, a new approach to the navigation of mobile robots for dynamic obstacle avoidance is proposed in (Belkhous et al., 2005). This approach merges the static and dynamic modes of path planning to provide an algorithm giving fast optimal solutions for static environments, and produces a new path whenever an unanticipated situation occurs.

3. Fuzzy Path Planning and Navigation of Industrial Manipulators

3.1 Fuzzy Obstacle Avoidance

Here, we will outline a technique developed by the authors (Zavlangas & Tzafestas, 2000), which has been primarily influenced by Khatib's (1986) artificial potential field method and the subsumption architecture developed by Brooks (1986). The local navigation approach is chosen because our main goal is to develop an on-line planner for fast collision – free trajectory generation. The proposed local navigator was implemented and applied to several practical scenarios. Our experimental results, some of which will be presented in Section 5, are very satisfactory. The technique is based on separate fuzzy logic-based obstacle avoidance units, each controlling one individual link l_j, $j = 1, \ldots, n$ Each unit has two principal inputs :

1. the distance between the link and the nearest obstacle d_j, and
2. the difference between the current link configuration and the target configuration,

$$\theta_j - \theta_{j,target} = \Delta\theta_j.$$

The output variable of each unit is the motor command τ_j. All these variables can be positive or negative, i.e., they inform both about the magnitude and the sign of displacement relative to the link – left or right. The motor command which can be interpreted as an actuation for the link motor is fed to the manipulator at each iteration (Figure 1). For the calculation of the distance, the only obstacles considered are those which fall into a bounded area surrounding each link and move along with it. In this implementation, each such area is chosen to be a cylindrical volume around each link. The area is as long as the link and reaches up to a predefined horizon. This area can be seen as a simplified model for the space scanned by ranging sensors (for example, ultra-sonic sensors) attached to the sides of a link (Pedrycz, 1995). Of course, other shapes to describe the 3-dimensional scan areas are conceivable. It is, for example, advisable to deal with the blind zones near the joints when the magnitude of the angles is large so as to assure that small moving obstacles are not missed by the algorithm (Pedrycz, 1995).
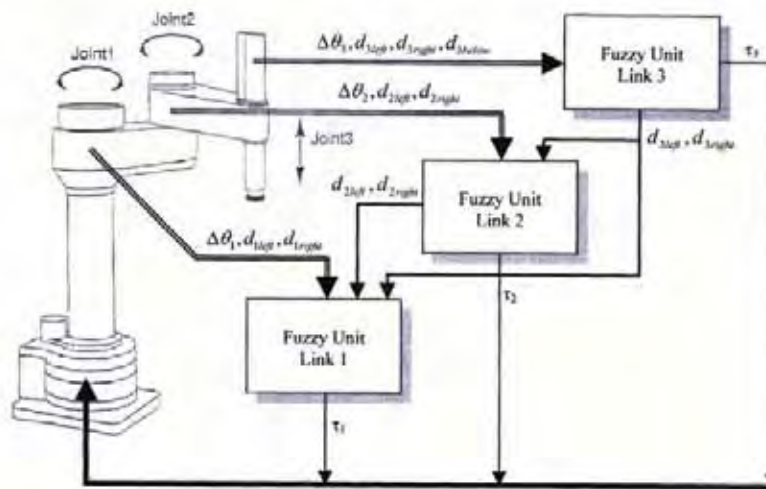
Figure 1. The Adept 1 industrial robotic manipulator connected to the corresponding fuzzy units.

Each fuzzy unit receives via an input the difference between target and actual configuration, and, via a second input, two values in a sequential way representing the distance between the corresponding link and the nearest obstacle on the left and on the right of this link. If no obstacle is detected inside the scan area, the fuzzy unit is informed of an obstacle in the far distance. Additionally, proximal units are informed about obstacles in the vicinity of more distal links. Besides an input from ultrasonic sensors, a camera can be used to acquire the environment. Either a stationary camera or a set of cameras which oversee the entire workspace can be utilised (Pedrycz, 1995; Jaitly & Fraser, 1996). The task of each fuzzy unit is to provide a control function which produces an appropriate motor command from the given inputs. In broad lines, the control function can be described as follows : on the one hand the function has to lead the corresponding link to its attracting end – position; on the other hand, it has to force the link to back up when approaching an obstacle which conveys a repelling influence. The fuzzy-rule-base (which represents the control function in each fuzzy unit) is built up by using common sense rules.

In our particular implementation, at each iteration the distances of the nearest obstacle on the left $\left(d_{jleft}\right)$ and on the right $\left(d_{jright}\right)$ of link l_j are fed sequentially into the fuzzy unit. This process could also be carried out in a parallel fashion where two equivalent fuzzy controllers compute the response for the left and the right obstacle separately. The resulting two motor commands are superimposed, hence, both obstacles influence the final motor command which is applied to the link. The use of this method guarantees that the repulsion caused by one obstacle on one side of the link does not result in a collision with a nearby obstacle on the opposite side. Only those obstacles are considered which are the nearest on the left and right.

In addition, fuzzy units of distal links communicate information about the distance to their nearest obstacles on the left and right to units of more proximal links. Once sent to fuzzy units of more proximal links, this information can be used by the decision process of those units to slow down or even reserve the motion of the more proximal links. Without this propagation of information the control strategy might fail in situations where one obstacle is "known" only to a fuzzy unit of a distal link, while proximal links continue their motion based on their local environment dictating an adverse motion for the rest of the arm. This is especially important, since the same change of angle occurring at a proximal link and at a distal link produces a different velocity at the manipulator's tip. Thus, the motion of a proximal link might not be sufficiently compensated by an adverse motion at a more distal link. Fuzzy units are only fed with the distance values of those obstacles which are inside the scan range. If no obstacle is detected inside a scan range, the fuzzy unit is informed of an obstacle which is *far left* or *far right,* respectively.

3.2 The Fuzzy Navigation Algorithm

The first input of each fuzzy unit is the difference between the actual angle and the target angle, $\theta_j - \theta_{j,target} = \Delta\theta_j \in \Theta_j$, $j = 1,\ldots,n$ (n is the number of links). The value $\Delta\theta_j$ is positive if the target is on the right, and negative if the target is on the left. The second input receives values describing the distance between link l_j and the nearest obstacles on the left and right in the "scanned" region, $d_j \in D_j$. An obstacle on the left produces a negative distance value, while an obstacle on the right produces a positive one. The single output is the motor command $\tau_j \in T_j$. A positive motor command moves the link to the left and a negative one to the right (Althoefer, 1996; Althoefer & Fraser, 1996).
Each universe of discourse D_j can be partitioned by fuzzy sets $\mu_1^{(j)},\ldots,\mu_{pj}^{(j)}$. Each of the sets $\mu_{\tilde{p}j}^{(j)}$, $\tilde{p}_j = 1,\ldots,p_j$, represents a mapping $\mu_{\tilde{p}j}^{(j)}(d_j): D_j \to [0,1]$ by which d_j is associated with a number in the interval $[0,1]$ indicating to what degree d_j is a member of the fuzzy set. Since d_j is a signed value, *"close_left"*, for example, may be considered as a particular fuzzy value of the variable distance and each d_j is assigned a number $\mu_{close\_left}(d_j) \in [0,1]$ which indicates the extent to which that d_j is considered to be close_left (Mamdani & Assilian, 1981). In an equivalent way, fuzzy sets $v_1^{(j)},\ldots,v_{qj}^{(j)}$ can be defined over the universe of discourse Θ_j. The Fuzzy Navigator is based on the Mamdani fuzzy model (Mamdani, 1974) and has an output set which is partitioned into fuzzy sets $\tau_{\tilde{r}j}$.
There is a variety of functions that can be employed to represent fuzzy sets (Mamdani & Assilian, 1981). In the proposed controller asymmetrical trape-

zoidal functions were employed to represent the fuzzy sets. The parameters, $ml^{(j)}$, $mr^{(j)}$, which are the x-coordinates of the left and right zero crossing, respectively, and $mcl^{(j)}$, $mcr^{(j)}$, which describe the x-coordinate (left and right) where the fuzzy set becomes 1, define the following trapezoidal functions :

$$\mu_{p_j}^{(j)}\left(d_j\right)=\begin{cases}\min\left(\dfrac{\left(d_j-ml_{p_j}^{(j)}\right)}{\left(mcl_{p_j}^{(j)}-ml_{p_j}^{(j)}\right)},0\right) & \text{if } ml_{p_j}^{(j)}<d_j\le mcl_{p_j}^{(j)},\\[2ex]1 & \text{if } mcl_{p_j}^{(j)}<d_j\le mcr_{p_j}^{(j)},\\[2ex]\min\left(\dfrac{\left(d_j-mr_{p_j}^{(j)}\right)}{mcr_{p_j}^{(j)}-mr_{p_j}^{(j)}},0\right) & \text{if } mcr_{p_j}^{(j)}<d_j\le mr_{p_j}^{(j)}.\end{cases}\tag{1}$$

As commonly done, the trapezoidal functions are continued as constant values of magnitude 1 at the left and right side of the interval (Equations (2) and (3)) :

$$\mu_{p_j}^{(j)}\left(d_j\right)=\begin{cases}1 & \text{if } ml_1^{(j)}<d_j\le mcl_1^{(j)},\\[2ex]1 & \text{if } mcl_1^{(j)}<d_j\le mcr_1^{(j)},\\[2ex]\min\left(\dfrac{\left(d_j-mr_1^{(j)}\right)}{\left(mcr_1^{(j)}-mr_1^{(j)}\right)},0\right) & \text{if } mcr_1^{(j)}<d_j\le mr_1^{(j)}.\end{cases}\tag{2}$$

and

$$\mu_{p_j}^{(j)}\left(d_j\right)=\begin{cases}\min\left(\dfrac{\left(d_j-ml_{p_j}^{(j)}\right)}{\left(mcl_{p_j}^{(j)}-ml_{p_j}^{(j)}\right)},0\right) & \text{if } ml_{p_j}^{(j)}<d_j\le mcl_{p_j}^{(j)},\\[2ex]1 & \text{if } mcl_{p_j}^{(j)}<d_j\le mcr_{p_j}^{(j)},\\[2ex]1 & \text{if } mcr_{p_j}^{(j)}<d_j\le mr_{p_j}^{(j)}.\end{cases}\tag{3}$$

Note, that for the two fuzzy sets of Equations (2) and (3) :

$$mcr_{p_j}^{(j)}=mr_{p_j}^{(j)},\ mcl_{p_j}^{(j)}=ml_{p_j}^{(j)}\tag{4}$$

The fuzzy sets for $\Delta\theta_j$ can be defined in the same way. The fuzzy sets $m_1^{(j)},\ldots,m_p^{(j)}$ and $n_1^{(j)},\ldots,n_q^{(j)}$ for link 2 as functions of the two inputs d_j and $\Delta\theta_j$ are shown in Figure 2, together with the output fuzzy set.
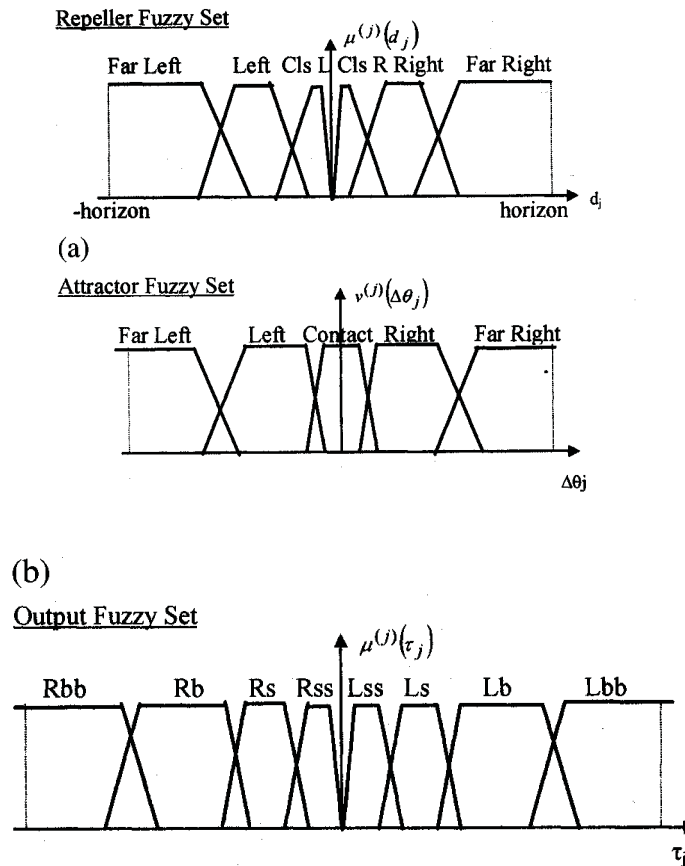
Figure 2. Repeller, attractor and output fuzzy sets for link 2.

The first diagram (a) is the fuzzy set of the distance between the link and the obstacle, and the second one (b) is the fuzzy set of the difference between the actual and target configuration. The output fuzzy set is shown in the third diagram (c).

Additionally to the two inputs d_j and $\Delta\theta_j$, each fuzzy unit (apart from the most distal one) uses the distance fuzzy sets of more distal links $\mu_{\tilde{p}_{j+1}}^{(j+1)},\ldots,\mu_{\tilde{p}_n}^{(n)}$ for decision making to assure that proximal links are slowed down, in case a more distal link is about to collide with an obstacle. Thus, each unit uses the following fuzzy sets : $\mu_{\tilde{p}_k}^{(k)}$, $k = j+1,\ldots,n$, and $v_{\tilde{q}_j}^{(j)}$. Each of the fuzzy sets $\mu_{\tilde{p}_k}^{(k)}$ and $v_{\tilde{q}_j}^{(j)}$ are associated with linguistic terms $A_{\tilde{p}_j}^{(j)}$ and $B_{\tilde{q}_j}^{(j)}$, respectively. Thus, for link l_j the linguistic control rules $R_1^{(j)},\ldots,R_{r_j}^{(j)}$, which constitute the rule base, can be defined as :

$$R_{\tilde{r}_j}^{(j)} : IF \ d_j \ is \ A_{\tilde{p}_j}^{(j)} \ AND \ldots AND \ d_n \ is \ A_{\tilde{p}_n}^{(n)} \ AND \ \Delta\theta_j \ is \ B_{\tilde{q}_j}^{(j)} \ THEN \ \tau_{\tilde{r}_j}$$

where $\tilde{r}_j = 1, \ldots, r_j$, r_j, is the number of rules for the fuzzy unit of link l_j, and $\tau_{\tilde{r}_j}$ is a numerical entry in the rule base used in the defuzzification process (Equation (5)). The most popular methods to calculate the fuzzy intersection (fuzzy – AND) are the *minimum* and *product* operators (Tzafestas & Venetsanopoulos, 1994; Zadeh, 1965; 1973). If the *minimum* operator is used, the minimum of the inputs is chosen. If the *product* operator is chosen, the inputs are multiplied with each other. While the result of the first approach contains only one piece of information, the second approach produces results which are influenced by all inputs (Brown, 1994).

Here, the fuzzy intersection is calculated by using the product operator "$*$" :

$$\sigma_{\tilde{r}_j} = \mu_{\tilde{p}_j,\tilde{r}_j}^{(j)}\left(d_j\right)\bigcap\cdots\bigcap\mu_{\tilde{p}_n,\tilde{r}_n}^{(n)}\left(d_n\right)\bigcap v_{\tilde{q}_j,\tilde{r}_j}^{(j)}\left(\Delta\theta_j\right) = \mu_{\tilde{p}_j,\tilde{r}_j}^{(j)}\left(d_j\right)*\cdots*\mu_{\tilde{p}_n,\tilde{r}_n}^{(n)}\left(d_n\right)*v_{\tilde{q}_j,\tilde{r}_j}^{(j)}\left(\Delta\theta_j\right)$$

The output of the unit is given by the centroid defuzzification over all rules (Kosko, 1992) :

$$\tau_j = \frac{\sum_{r_j=1}^{r_j}\tau_{r_j}*\mu_{p_j,r_j}^{(j)}\left(t_{r_j}\right)}{\sum_{r_j=1}^{r_j}\mu_{p_j,r_j}^{(j)}\left(t_{r_j}\right)} \tag{5}$$

The fuzzy rule base for link 2 is displayed in Table I.

	Far left	Left	Close left	Close right	Right	Far right
Far left	Ls	Rs	Rbb	Lb	Ls	Ls
Left	Lss	Rs	Rbb	Ls	Lss	Lss
Contact	nil	nil	nil	nil	nil	nil
Right	Rss	Rss	Rs	Lbb	Ls	Rss
Far right	Rs	Rs	Rb	Lbb	Ls	Rs
Lss, Rss : very small to the left / right					
Ls, Rs : small to the left / right					
Lb, Rb : big to the left / right					
Lbb, Rbb : very big to the left / right					

Table 1. FAM matrix (Fuzzy Associative Memory) for link 2

4. Fuzzy Path Planning and Navigation of Mobile Robots

4.1 General Description

Basically, the technique to be described for mobile fuzzy path planning and navigation is the same with that described in Section 3 for the case of industrial manipulators, i.e. it is based on Khatib's potential field method (Khatib, 1986) and on Brooks subsumption structure (Brooks, 1986).

Khatib computes an artificial potential field that has a strong repelling force in the vicinity of obstacles and an attracting force produced by the target location. The superposition of the two forces creates a potential field, which incorporates information about the environment. Following the steepest gradient from a start position, a path can be found that guides the robot to the target position avoiding obstacles. In our approach the amount of computation, that is required, is reduced by using only the nearest obstacles to determine the direction of motion. The main idea of Brooks is a collection of modules, which are interconnected on different layers with different hierarchies. These modules are for example *wall following, obstacle avoidance, goal reaching,* etc. Depending on sensory input, a module becomes active and generates a command for the robot. While Brooks' system resembles an expert system where for any input signal one specific reaction module or a specific combination of modules is active, our fuzzy approach is a parallel processing strategy where each input contributes to the final decision (see Section 3.2).

The technique is based on two fuzzy – based controllers, one for **steering control,** and the other for **velocity control**. The steering controller has three principal inputs :

1) the distance between the robot and the nearest obstacle d_j,
2) the angle between the robot and the nearest obstacle γ_j, and
3) the angle between the robot's direction and the straight line connecting the current position of the robot and the goal configuration $\theta_j = \alpha_j - \beta_j$, where β_j is the angular difference between the straight line connecting the robot's current position and the goal configuration, and α_j is the current direction of the robot (see Fig. 3).

The output variable of this unit is the required change of angle $\Delta\theta_j$, and can be considered as a command for the robot's steering actuators. The velocity controller has two principal inputs :

1) the distance between the robot and the nearest obstacle d_j,
2) the distance between the robot and the goal configuration d_{g_j}.

The output variable of this unit is an acceleration command Δv_j, and can be considered as a command for the robot's drive actuators. All these variables can be positive or negative, i.e. they do not only inform about the magnitude, but also about the sign of displacement relative to the robot – left or right. The motor commands are fed to the mobile platform at each iteration.

(a)   (b)
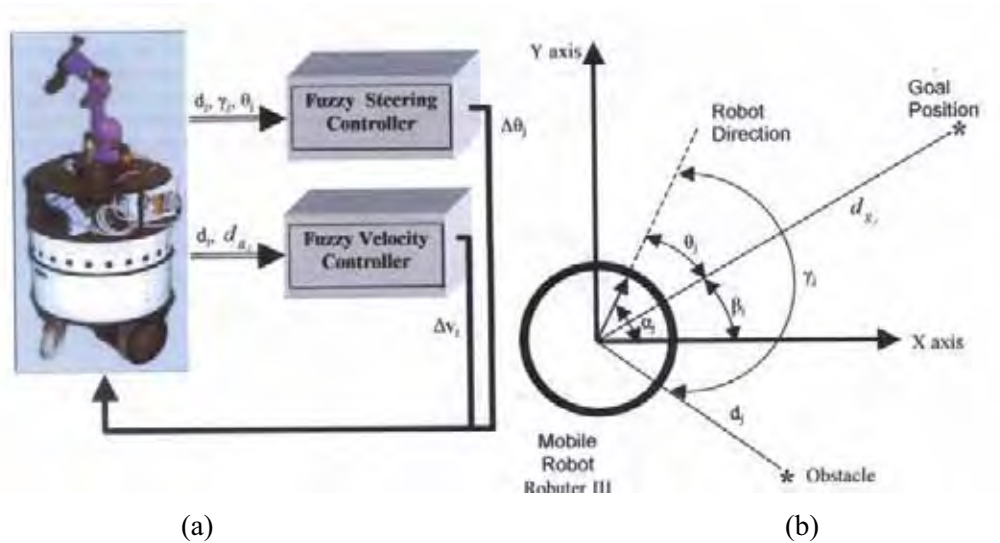
Figure3. (a) The Robosoft Robuter III mobile robot of IRAL / NTUA connected to the corresponding fuzzy-based obstacle avoidance unit.

This steering control unit receives via an input the angle between the robot's direction and the straight line connecting the current position of the robot and the goal configuration $\theta_j = \alpha_j - \beta_j$, and the distance and the angle to the nearest obstacle $\left(d_j, \gamma_j \right)$ (see (b)). If no obstacle is detected inside the scan area, the fuzzy unit is informed of an obstacle in the far distance. The output variable of this unit is the required change of angle $\Delta \theta_j$, and can be considered as a command for the robot's steering actuators. The velocity control unit has two inputs : the distance between the robot and the nearest obstacle d_j, and the distance between the robot and the goal configuration d_{g_j}. The output variable of this unit is an acceleration command Δv_j and can be considered as a command for the robot's driving actuators.

The obstacles considered are those that fall into a confined area surrounding the robot and moving along with it. Here, this area is chosen to be a cylindrical volume around the mobile platform. This area is regarded as a simplified model for the space scanned by ranging sensors (for example ultrasonic sensors) attached to the sides of the robot (Pedrycz, 1995). Besides an input from ultrasonic sensors, a camera can also be used to acquire the environment. Mo-

bile robots are usually equipped with a *pan / tilt platform* where a camera is mounted. This camera can be utilized as shown in (Pedrycz, 1995; Jaitly et al., 1996; Kamon & Rivlin, 1997). If no obstacle is detected inside the scan area, the fuzzy units are informed of an obstacle in the far distance. The task of the fuzzy units is to provide a control function, that produces appropriate motor commands from the given inputs. This control function on the one hand has to lead the mobile robot to its attracting goal–position, and on the other hand it has to force the robot to back up when approaching an obstacle which conveys a repelling influence. The fuzzy–rule – base (which represents the control function in the fuzzy unit) is constructed using common sense rules or by a neural network training algorithm (see e.g., (Tzafestas & Zavlangas, 1999)). An alternative fuzzy motion control scheme of mobile robots which employs the sliding – mode control principle can be found in (Rigatos & Tzafestas, 2000).

4.2 Detailed Description

As mentioned in Section 4.1 the proposed methodology is based on two fuzzy – based controllers, one for **steering control,** and the other for **velocity control**. The fuzzy controllers here are based on the functional reasoning control principles developed by Sugeno (see, for example, (Sugeno, 1985; Sugeno & Murakami, 1984)). For the steering controller, each input space is partitioned by fuzzy sets as shown in Fig. 4. Here, asymmetrical triangular and trapezoidal functions are utilized to describe each fuzzy set, which allow a fast computation, essential under real – time conditions (see Eqs. (6) – (8) and (11)). The fuzzy sets of the three inputs d_j, γ_j and θ_j are depicted in Figure 4. To calculate the fuzzy intersection, the product operator is employed (see Eq (10)). The final output of the unit is given by a weighted average over all rules (see Eq. (11)). Intuitively, the rules for obstacle – avoiding navigation can be written as sentences with three antecedents and one conclusion. This structure lends itself to a tabular representation such as the one shown in Table 2. This table represents the prior knowledge of the problem domain. The tools of fuzzy logic allow us to translate this intuitive knowledge into a control system. To translate Table 2 into fuzzy logic, the universe of discourse D_j which describes the distance $d_j \in D_j$ to the obstacle is partitioned by fuzzy sets $\mu_1^{(j)}, ..., \mu_{p_j}^{(j)}$, where p_j is the number of fuzzy sets. Each set $\mu_{\tilde{p}_j}^{(j)}$, $\tilde{p}_j = 1, ..., p_j$, represents a mapping $\mu_{\tilde{p}_j}^{(j)}(d_j): D_j \to [0,1]$ by which d_j is considered as a particular fuzzy value of the variable distance and each d_j is associated with, e.g., a number in the interval [0,1] indicating to what degree d_j is a member of the fuzzy set. Since d_j is a measure of distance, "very_close", it may be assigned a number : $\mu_{very\_close}(d_j) \in [0,1]$ which indicates the extent to which this particular d_j is considered to be "very_close" (Mamdani & Assilian, 1981).
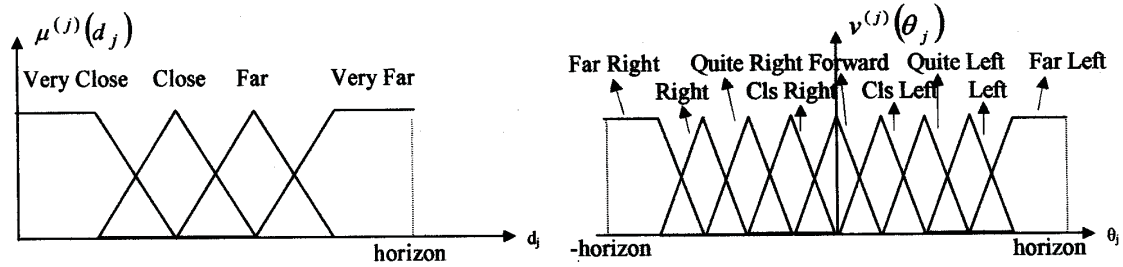
(a) Distance to Obstacle   (b) Angle to Goal Position
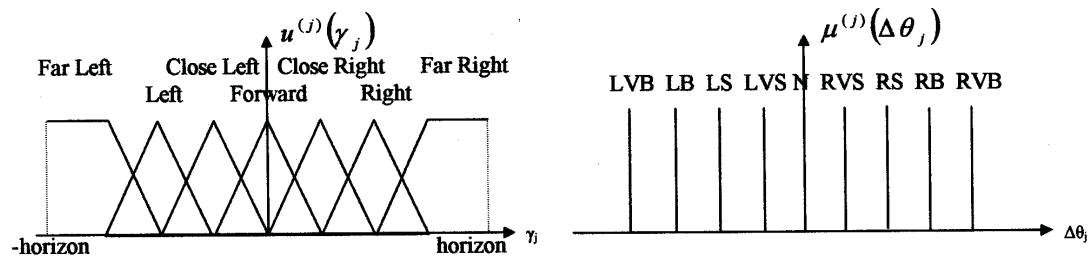
(c) Angle to Obstacle   (d) Steering Motor Command

Fig. 4. Fuzzy sets for the mobile robot : (a) distance to obstacle, (b) angle between robot and goal position, (c) angle between robot and obstacle, and (d) steering motor command. Note that the output is not partitioned into fuzzy sets, but consists of crisp values.

v_j / μ_j	very close	close	far	very far
far right	right big	right small	left very big	left very big
right	right big	right small	left big	left big
quite right	right big	right small	left small	left small
close right	right big	right small	left very small	left very small
forward	null	null	null	null
close left	right very small	right very small	right very small	right very small
quite left	right small	right small	right small	right small
left	right big	right big	right big	right big
far left	right very big	right very big	right very big	right very big
This rule-base is a translation of the common – sense knowledge of the problem domain into the language of fuzzy logic. Rows represent the fuzzy measures of the distance to an obstacle, while columns are fuzzy representations of the angle to the goal. Each element of the table can be interpreted as a particular motor actuation command.				

Table 2. A rule – base for the mobile robot when γ_j is "far_left".

Similarly, fuzzy sets $v_1^{(j)}, \dots v_{q_j}^{(j)}$ can be defined over the universe of discourse Θ_j which represents the angle between the robot's direction and the straight line connecting the current position of the robot and the goal configuration : $\theta_j \in \Theta_j$. Finally, fuzzy sets $u_1^{(j)}, \dots, u_{g_j}^{(j)}$ can be defined over the universe of discourse Γ_j that represents the angle to the nearest obstacle $\gamma_j \in \Gamma_j$. In contrast to the Mamdani's controller, Sugeno's controller (see (Sugeno, 1985; Sugeno & Murakami, 1984; Tzafestas & Zikidis, 2001)), of which ours is an example, has an output set which is not partitioned into fuzzy sets (see Fig. 2). Thus, the rule conclusions merely consist of scalars $\Delta\theta_{\tilde{r}_j}$, $\tilde{r}_j = 1, \dots, r_j$.

The fuzzy sets $\mu_{\tilde{p}_j}^{(j)}$, $\tilde{p}_j = 1, \dots, p_j$, are described by asymmetrical triangular and trapezoidal functions. Defining the parameters, $ml_{\tilde{p}_j}^{(j)}$ and $mr_{\tilde{p}_j}^{(j)}$ as the x-co-ordinates of the left and right zero crossing respectively, and $mcl_{\tilde{p}_j}^{(j)}$ and $mcr_{\tilde{p}_j}^{(j)}$ as the x-co-ordinates of the left and right side of the trapezoid's plateau, the trapezoidal functions can be written as :

$$
\mu_{\tilde{p}_j}^{(j)}\left(d_j\right) = \begin{cases} \max\left(\left(d_j - ml_{\tilde{p}_j}^{(j)}\right)/\left(mcl_{\tilde{p}_j}^{(j)} - ml_{\tilde{p}_j}^{(j)}\right), 0\right) & \text{if } d_j < mcl_{\tilde{p}_j}^{(j)} \\ 1 & \text{if } mcl_{\tilde{p}_j}^{(j)} \leq d_j \leq mcr_{\tilde{p}_j}^{(j)} \\ \max\left(\left(d_j - mr_{\tilde{p}_j}^{(j)}\right)/\left(mcr_{\tilde{p}_j}^{(j)} - mr_{\tilde{p}_j}^{(j)}\right), 0\right) & \text{if } d_j > mcr_{\tilde{p}_j}^{(j)} \end{cases} \tag{6}
$$

with $\tilde{p}_j = 1, \dots, p_j$. Triangular functions can be achieved by setting $mcl_{\tilde{p}_j}^{(j)} = mcr_{\tilde{p}_j}^{(j)}$.

At the left and right sides of the interval, the functions are continued as constant values of magnitude one, i.e. :

$$
\mu_1^{(j)}\left(d_j\right) = \begin{cases} 1 & \text{if } d_j \leq mcr_1^{(j)} \\ \max\left(\left(d_j - mr_1^{(j)}\right)/\left(mcr_1^{(j)} - mr_1^{(j)}\right), 0\right) & \text{if } d_j > mcr_1^{(j)} \end{cases} \tag{7}
$$

and

$$
\mu_{p_j}^{(j)}\left(d_j\right) = \begin{cases} \max\left(\left(d_j - ml_{p_j}^{(j)}\right)/\left(mcl_{p_j}^{(j)} - ml_{p_j}^{(j)}\right), 0\right) & \text{if } d_j \leq mcl_{p_j}^{(j)} \\ 1 & \text{if } d_j > mcl_{p_j}^{(j)} \end{cases} \tag{8}
$$

The fuzzy sets for θ_j and γ_j are defined analogously. Figure 4 shows the fuzzy sets $\mu_1^{(j)}, \dots, \mu_{p_j}^{(j)}$, $v_1^{(j)}, \dots, v_{q_j}^{(j)}$ and $u_1^{(j)}, \dots, u_{g_j}^{(j)}$.

Every fuzzy set, $\mu_{\tilde{p}_j}^{(j)}$, $v_{\tilde{q}_j}^{(j)}$ and $u_{\tilde{g}_j}^{(j)}$, is associated with linguistic terms $A_{\tilde{q}_j}^{(j)}$, $B_{\tilde{q}_j}^{(j)}$ and $C_{\tilde{g}_j}^{(j)}$ respectively. Thus, for the mobile robot, the linguistic control rules $R_1^{(j)},\ldots,R_{r_j}^{(j)}$ which constitute the rule base can be defined as :

$$R_{\tilde{r}_j}^{(j)} : \textbf{IF } d_j \text{ is } A_{\tilde{p}_j}^{(j)} \textbf{ AND } \theta_j \text{ is } B_{\tilde{q}_j}^{(j)} \textbf{ AND } \gamma_j \text{ is } C_{\tilde{g}_j}^{(j)} \textbf{ THEN } f\left(\tau_{\tilde{r}_j}\right) \ \left(\tilde{r}_j = 1,\ldots,r_j\right) \quad (9)$$

where the **AND** operations use the t-norm product operator :

$$\sigma_{\tilde{r}_j} = \mu_{\tilde{p}_j,\tilde{r}_j}^{(j)}\left(d_j\right) \cap v_{\tilde{q}_j,\tilde{r}_j}^{(j)}\left(\theta_j\right) \cap u_{\tilde{g}_j,\tilde{r}_j}^{(j)}\left(\gamma_j\right) = \mu_{\tilde{p}_j,\tilde{r}_j}^{(j)}\left(d_j\right) * v_{\tilde{q}_j,\tilde{r}_j}^{(j)}\left(\theta_j\right) * u_{\tilde{g}_j,\tilde{r}_j}^{(j)}\left(\gamma_j\right) \quad (10)$$

Finally, the output of the unit is given by a weighted average over all rules (see Fig. 2 and (Topalov & Tzafestas, 2001)) :

$$\Delta\theta_j = \frac{\displaystyle\sum_{\tilde{r}_j=1}^{r_j} \sigma_{\tilde{r}_j} \cdot \Delta\theta_{\tilde{r}_j}}{\displaystyle\sum_{\tilde{r}_j=1}^{r_j} \sigma_{\tilde{r}_j}} \quad (11)$$

Eq. (9) together with Eqs. (10) and (11) define how to translate the intuitive knowledge reflected in Table 2 into a fuzzy rule – base. The details of this translation can be modified by changing the number of fuzzy sets, the shape of the sets (by choosing the parameters, $ml_{\tilde{p}_j}^{(j)}$ and $mr_{\tilde{p}_j}^{(j)}$, $mcl_{\tilde{p}_j}^{(j)}$, $mcr_{\tilde{p}_j}^{(j)}$ as well as the value $\Delta\theta_{\tilde{r}_j}$ of each of the rules in Eq. (11). A technique for the automated selection of the above parameters is provided in (Tzafestas & Zikidis, 2001). As an example, the control rules for the particular mobile robot are shown in Table 2. In this application, the number of fuzzy sets which fuzzify the obstacle distance d_j, the angle to the goal θ_j and the angle to the nearest obstacle γ_j are chosen to be four, nine and seven, respectively. All the other parameters were refined by trial and error.
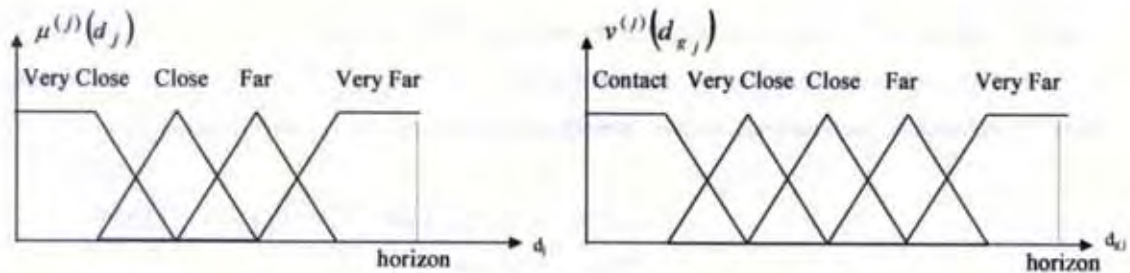
(a) Distance to Obstacle   (b) Angle to goal configuration
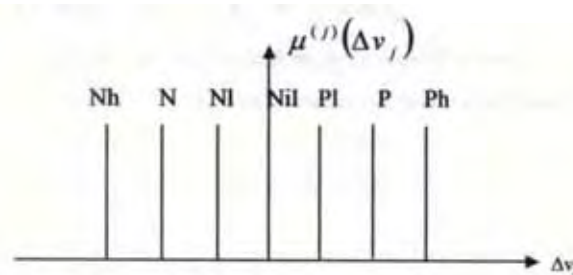
(c) Drive Motor Command

Figure 5. Fuzzy sets for the velocity control unit of the mobile robot : (a) the distance between the robot and the nearest obstacle, and (b) the distance between the robot and the goal configuration. Note that the output is an acceleration command and is not partitioned into fuzzy sets, but consists of crisp values.

For the velocity controller, each input space is partitioned by fuzzy sets as shown in Figure 5. Asymmetrical triangular and trapezoidal functions have also been used to describe each fuzzy set, which allow a fast computation, essential under real-time conditions (see Eqs. (12) – (13)).
To calculate the fuzzy intersection, the product operator is employed (see Eq. (16)). The final output of the unit is given by a weighted average over all rules (see Eq. (17)).
Intuitively, the rules for the velocity controller can be written as sentences with two antecedents and one conclusion. This structure is represented by Table 3 which provides the prior knowledge of the problem domain. The tools of fuzzy logic allow us to translate this intuitive knowledge into a control system.

$v\left(d_{g_j}\right)\Big/\mu\left(d_j\right)$	very close	close	far	very far
contact	negative high	negative high	negative high	negative high
very close	negative high	negative high	negative	negative low
close	negative high	negative	nil	nil
far	negative high	negative	positive low	positive
very far	negative high	negative	positive	positive high
Rows represent the fuzzy measures of the distance to an obstacle, while columns are fuzzy representations of the distance between the robot and the goal configuration. Each element of the table can be interpreted as a particular drive motor actuation command for the acceleration and deceleration of the mobile robot.				

Table 3. The rule – base for the velocity controller.

The fuzzy sets $\mu_{\tilde{p}_j}^{(j)}$, $\tilde{p}_j = 1, \ldots, p_j$, are described by asymmetrical triangular and trapezoidal functions. Defining the parameters, $ml_{\tilde{p}_j}^{(j)}$ and $mr_{\tilde{p}_j}^{(j)}$ as the x-co-ordinates of the left and right zero crossing respectively, and $mcl_{\tilde{p}_j}^{(j)}$ and $mcr_{\tilde{p}_j}^{(j)}$ as the x-co-ordinates of the left and right side of the trapezoid's plateau, the trapezoidal functions can be written as :

$$
\mu_{\tilde{p}_j}^{(j)}\left(d_j\right) = \begin{cases} \max\left(\left(d_j - ml_{\tilde{p}_j}^{(j)}\right)/\left(mcl_{\tilde{p}_j}^{(j)} - ml_{\tilde{p}_j}^{(j)}\right), 0\right) & \text{if } d_j < mcl_{\tilde{p}_j}^{(j)} \\ 1 & \text{if } mcl_{\tilde{p}_j}^{(j)} \leq d_j \leq mcr_{\tilde{p}_j}^{(j)} \\ \max\left(\left(d_j - mr_{\tilde{p}_j}^{(j)}\right)/\left(mcr_{\tilde{p}_j}^{(j)} - mr_{\tilde{p}_j}^{(j)}\right), 0\right) & \text{if } d_j > mcr_{\tilde{p}_j}^{(j)} \end{cases} \qquad (12)
$$

with $\tilde{p}_j = 1, \ldots, p_j$. Triangular functions can be achieved by setting $mcl_{\tilde{p}_j}^{(j)} = mcr_{\tilde{p}_j}^{(j)}$.

At the left and right side of the interval the functions are continued as constant values of magnitude one :

$$
\mu_1^{(j)}\left(d_j\right) = \begin{cases} 1 & \text{if } d_j \leq mcr_1^{(j)} \\ \max\left(\left(d_j - mr_1^{(j)}\right)/\left(mcr_1^{(j)} - mr_1^{(j)}\right), 0\right) & \text{if } d_j > mcr_1^{(j)} \end{cases} \qquad (13)
$$

and

$$
\mu_{p_j}^{(j)}\left(d_j\right) = \begin{cases} \max\left(\left(d_j - ml_{p_j}^{(j)}\right)/\left(mcl_{p_j}^{(j)} - ml_{p_j}^{(j)}\right), 0\right) & \text{if } d_j \leq mcl_{p_j}^{(j)} \\ 1 & \text{if } d_j > mcl_{p_j}^{(j)} \end{cases} \qquad (14)
$$

The fuzzy sets for d_{gj} are defined analogously. Figure 5 shows the fuzzy sets $\mu_1^{(j)},\ldots,\mu_{p_j}^{(j)}$ and $v_1^{(j)},\ldots,v_{q_j}^{(j)}$.

The linguistic control rules $R_1^{(j)},\ldots,R_{r_j}^{(j)}$, which constitute the rule base, are defined as :

$$R_{\tilde{r}_j}^{(j)} : \textbf{IF } d_j \text{ is } A_{\tilde{p}_j}^{(j)} \textbf{ AND } d_{gj} \text{ is } B_{\tilde{q}_j}^{(j)} \textbf{ THEN } f\left(\Delta v_{\tilde{r}_j}\right) \quad \left(\tilde{r}_j = 1,\ldots,r\right) \tag{15}$$

where the **AND** operations use the t-norm product operator, defined as :

$$\sigma_{\tilde{r}_j} = \mu_{\tilde{p}_j,\tilde{r}_j}^{(j)}\left(d_j\right) \cap v_{\tilde{q}_j,\tilde{r}_j}^{(j)}\left(d_{g_j}\right) = \mu_{\tilde{p}_j,\tilde{r}_j}^{(j)}\left(d_j\right) * v_{\tilde{q}_j,\tilde{r}_j}^{(j)}\left(d_{g_j}\right) \tag{16}$$

Finally, the output of the velocity control unit is given by a weighted average over all rules (Eq. 16) :

$$\Delta v_j = \frac{\displaystyle\sum_{\tilde{r}_j=1}^{r_j} \sigma_{\tilde{r}_j} \cdot \Delta v_{\tilde{r}_j}}{\displaystyle\sum_{\tilde{r}_j=1}^{r_j} \sigma_{\tilde{r}_j}}$$

4.3 Integrated global and local path planning

The path planner and navigator presented in Sections 4.1 and 4.2 is of the reactive type and is used at the local level. The full path following problem involves also a global path planner at a higher level (called geometric level) as shown in Fig. 6. The authors have derived and implemented a global path planner based on the potential field method which was selected for its mathematical elegance and simplicity. However, as it is evident, more complex / complete global path planning techniques can be embedded into our system. Global path planning strategies are more easily planned using a topological map, where the planner can decide the sequence of rooms and corridors to be traversed (Dudek et al., 1991; Ryu & Yang, 1999; Kortenkamp & Weymouth, 1994; Thrun, 1999; Kuipers & Levitt, 1988).
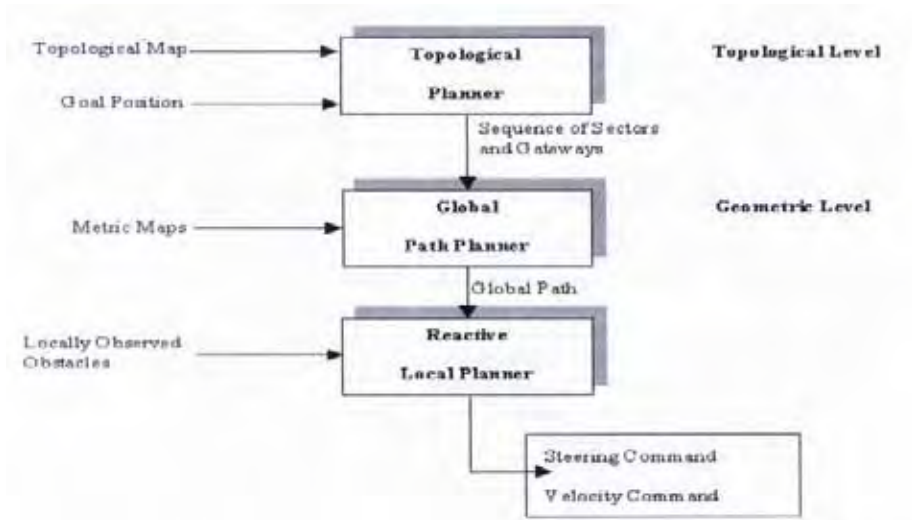
Figure 6. Three – level path planning and navigation.

The topological maps (networks) are placed at the top level of the hierarchical structure. The authors have conducted a number of experiments where both the topological and local metric maps were given to the robot a priori. The enormous compactness of topological maps when compared to the underlying grid-based map, increases substantially the efficiency of global planning. The paths are planned using the abstract topological map of the robot's environment (Fig. 7). Shortest paths on the topological maps can be easily found using one of the standard graph search algorithms, such as Dijkstra's or Floyd and Warshal's shortest path algorithm, the A^* algorithm, or dynamic programming. In our case we have used the Dijkstra's shortest path algorithm. This algorithm finds the best (shortest) path on the topological map going from the sector that contains the current location of the robot to the one which contains the goal. For example, a plan to get from the "start" position in Fig. 7 to an office desk in room R_2 may be :
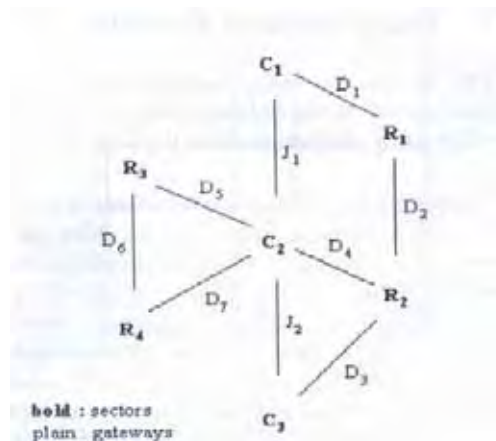
(a)
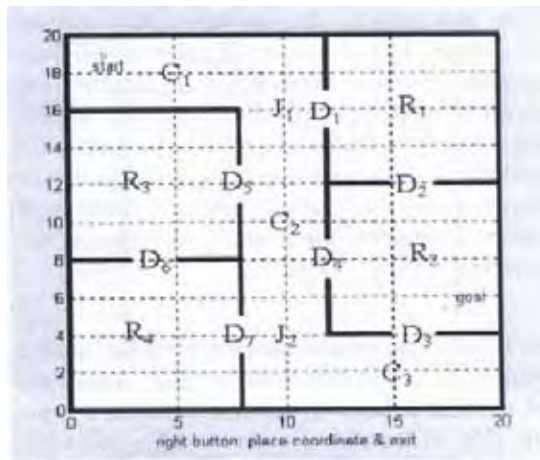
(b)

Figure 7. (a) The topological map used for the experiments, and (b) the corresponding network of the local metric maps.

5. Experimental Results

5.1 Industrial Manipulator

To test the functionality and performance of the basic algorithms, the fuzzy navigator was applied to a simulated 3-dof industrial manipulator (Adept 1). All experiments were carried out on a personal computer (Pentium II, 350 MHz).
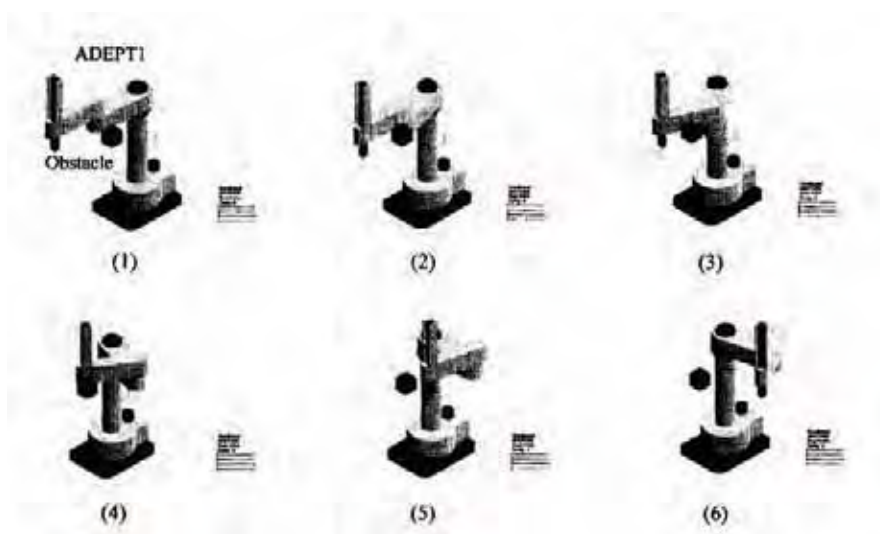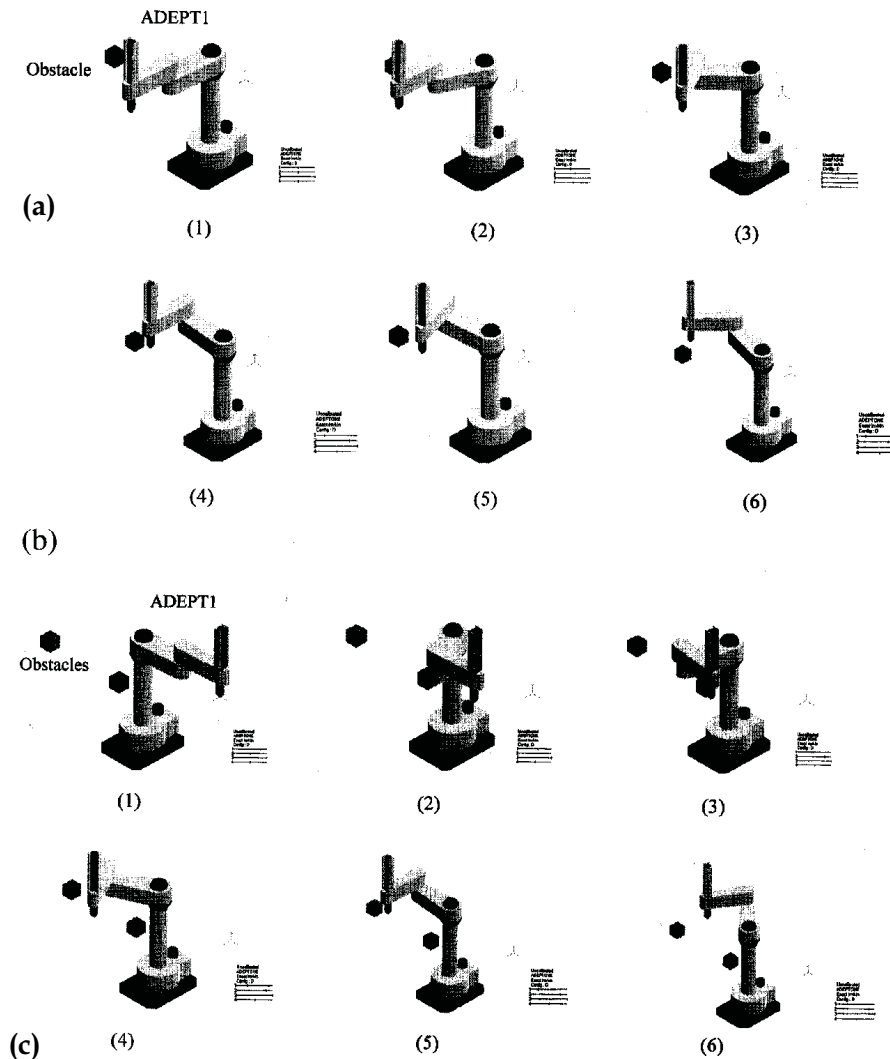
Figure 8. Simulation results using the ADEPT 1 industrial manipulator.

The proposed fuzzy navigator was successfully tested for the path planning / obstacle avoidance problem in three working scenarios with different obstacle constellations (a), (b) and (c), each time providing the manipulator with a collision – free trajectory.

The fuzzy navigator was developed using Microsoft C and for the visualization and animation of the robot's path Workspace 4 software package was used.

The performance of the fuzzy navigator for the Adept 1 industrial manipulator was tested in a variety of environments, with different obstacle constellations and working scenarios, even in dynamic environments with moving obstacles. In all cases, the fuzzy navigator provided the system with a collision free motion. Simulation results obtained in three different working scenarios (with different obstacle constellations) are presented in Figure 8.

5.2 Mobile Robot local path planning

The experimental results were first derived for the simulated omnidirectional mobile robot of the IRAL Laboratory (Robosoft Robuter III) using a Personal Computer (Pentium IV, 1.3 GHz) and the Matlab 5.2 software package. The fuzzy navigator was tested in a variety of environment scenarios (with both static and moving obstacles). In all cases a collision – free motion was obtained. A representative set of results are shown in Fig. 9 for four distinct working scenarios.

Working scenario 1

(a)   (b)

Working scenario 2

(a)   (b)

Working scenario 3

(a)   (b)
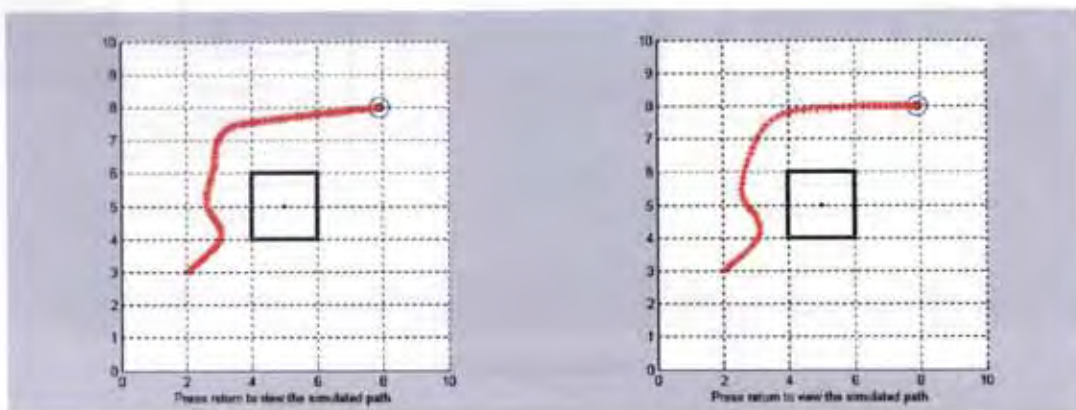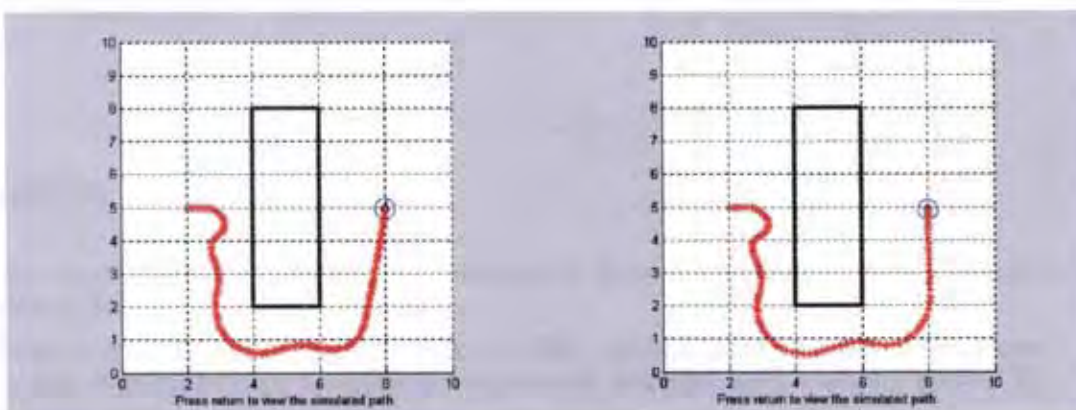
Working scenario 4

(a)   (b)
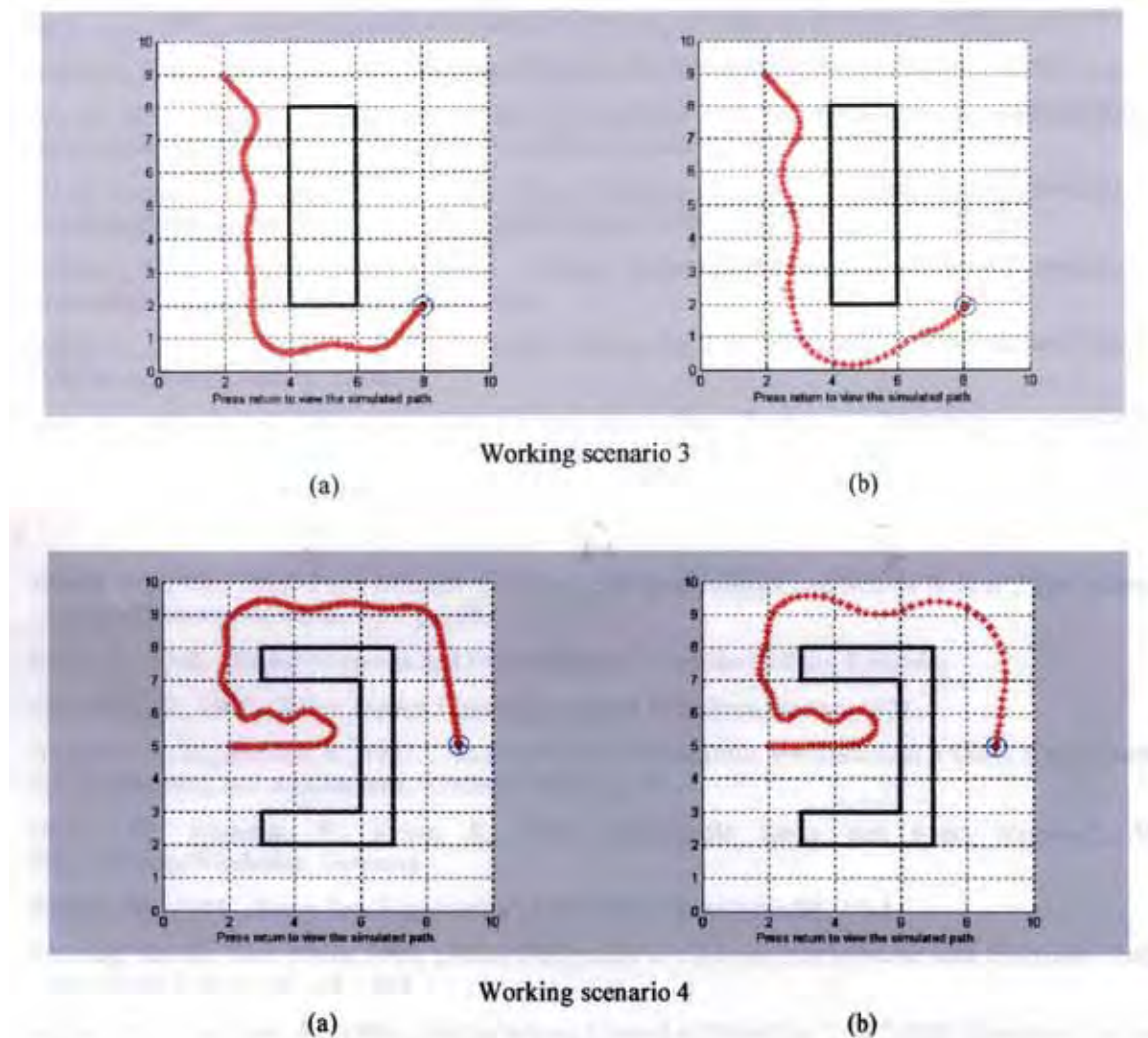
Figure 9. Simulation results using the Robosoft Robuter III mobile robot.

The proposed strategy was successfully tested in four working scenarios with different obstacle constellations each time providing the mobile robot with a collision – free movement. In (a) the path of the robot with only steering control is shown, and in (b) the path with the steering and velocity controllers is shown.

5.3 Mobile robot global path planning

The topological – map based global path planning technique was applied to the simulated Robosoft Robuter III mobile robot using the environment depicted in Fig. 7.

In **experiment 1,** the mobile robot was in corridor C_1 and was instructed to go to an office desk in room R_2. The output sequence / plan of the topological

planner is : first follow corridor C_1 up to junction J_1, then traverse junction J_1, then follow corridor C_2 up to door D_4, get-close-to door D_4, then cross it, then traverse room R_2, and finally get-close-to the desk. The robot moved to its goal configuration following the local paths generated by the path planner.

In **experiment 2,** the mobile robot was in room R_1 and was instructed to go to a position in corridor C_1. The output sequence / plan of the topological planner is : first traverse room R_1 up to door D_1, get-close-to door D_1, then cross it, then reverse junction J_1, then follow corridor C_1 and finally get-close-to goal position. The robot moved to its goal configuration following the local paths generated by the path planner.
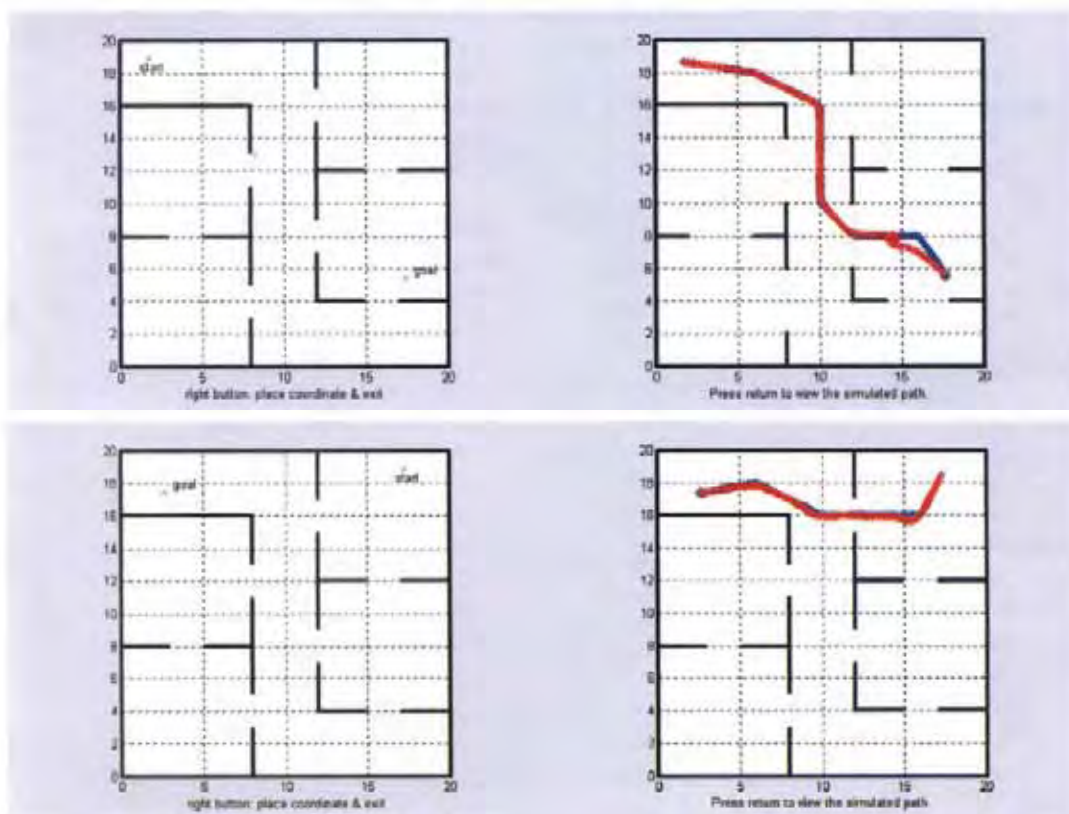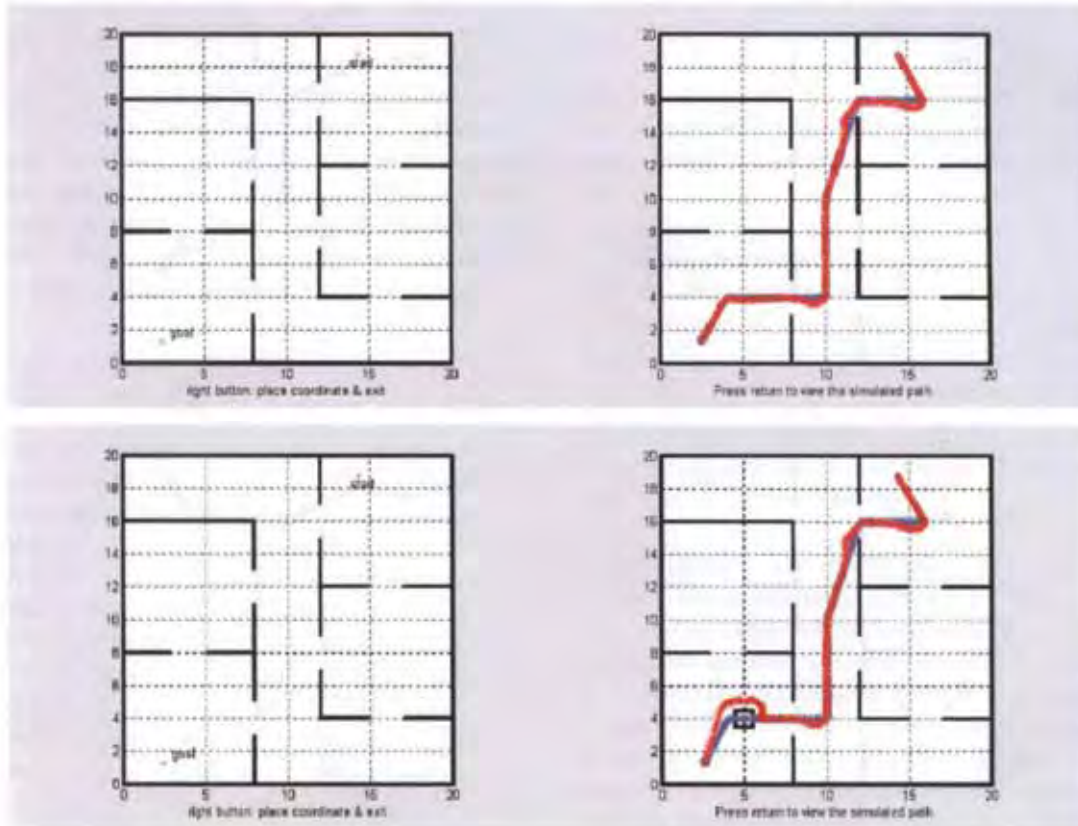
Figure 10. Illustration of mobile robot global path planning experiments.

In **experiment 3,** the mobile robot was in room R_1 and was instructed to go to an office book – case in room R_4. The output sequence / plan of the topological planner is : first traverse room R_1 up to door D_1, get-close-to door D_1, then cross it, then traverse corridor C_2 up to door D_7, then get-close-to door D_7, then cross it, then traverse room R_4, and finally get-close-to the bookcase. The robot moved to its goal configuration following the local paths generated by the path planner.

In **experiment 4,** experiment 3 was repeated. The mobile robot was in room R_1 and was instructed to go to an office bookcase in room R_4. The output sequence / plan of the topological planner is : first traverse room R_1 up to door D_1, get-close-to door D_1, then cross it, then traverse corridor C_2 up to door D_7, then get-close-to door D_7, then cross it, then traverse room R_4, and finally get-close-to the bookcase. The robot moved to its goal configuration following the local paths generated by the path planner. However, an unexpected obstacle occurred in room R_4 and the local reactive planner made the appropriate adjustments to the robot's motion in order to avoid the collision.

6. Conclusions and Directions for Research

In this chapter the problem of collision – free path planning and navigation was studied for both robotic manipulators and omnidirectional mobile robots. A short review of the problem itself was provided along with a number of fundamental and recent references. The methodology followed was based on fuzzy logic and reasoning and a promising fuzzy path planner and navigator was developed, that can be applied to both industrial and mobile robots. This path planner / navigator has been tested by simulation in a variety of working scenarios with different obstacle constellations, both static and dynamic, providing each time a collision-free trajectory for the robotic manipulator. This local planning method has shown a robust and stable performance and the experimental results were very satisfactory. The difference between the presented approach and the existing obstacle avoidance techniques is that the proposed navigator considers only the nearest obstacle to decide upon the robot's next move. This clearly leads to a large reduction in the required remote sensing and computations. A drawback is that, this reduction in information about the robot's environment, leads to an increased possibility of getting trapped in local minima. There may be routes leading to the goal that avoid the obstacles, which our navigator will not be able to find.

Concerning the mobile robots both local and global path planners were developed and tested. The local geometric information was encoded in a set of local sectors, whereas the global topological information was encoded in a network connecting these sectors. Each sector is a Cartesian representation, with its own reference system, that covers a limited area of the environment, like a room, a hall, or a corridor, and includes an approximate geometric description of the boundaries of the objects in the environment. The environment at the local level was represented by local metric maps connected to the topological network. In this way one can use maps that are not metrically consistent on the global scale although they are metrically consistent locally. This structure allows also the combination of abstract global reasoning and precise local geometric computations. To navigate in the environment, the robot uses, at the higher level of abstraction, the topological information to plan a sequence of sectors and gateways to traverse, and, to the lower level, uses the metric information in each sector to perform geometric path planning and navigation, and to locally move within the sector and to the next one. The system has shown a very stable and robust performance, providing each time the mobile robot with a collision free movement. The proposed fuzzy navigator is very fast and can be used in real time.

Future research may be devoted to the application of neural learning mechanisms applied on the fuzzy navigator providing an adaptive neurofuzzy planner (Topalov & Tzafestas, 2001; Brown, 1994; Tzafestas & Zavlangas, 1999; Tzafestas & Zikidis, 2001; Wang, 1994; Stamou & Tzafestas, 2000). The authors'

group is also investigating the motion planning and control problem of mobile robots equipped with manipulator arms, i.e., mobile manipulators (Tzafestas & Tzafestas, 2001; Bayle et al., 2003; Tzafestas et al., 2000; Erden et al., 2004). Other potential directions for future research needed in both industrial and non-industrial (e.g., service, medical) applications include multirobotic systems involving industrial manipulators, mobile robots and mobile manipulators, and the application of behavior-based techniques to both single robotic and multiple cooperating robots (Erden et al., 2004; Hsu & Liu, 2005; Arai et al., 1999; Balch & Arkin, 1998; Yamaguchi et al., 2001; Wang, 1991).

7. References

Althoefer, K. (1996). Fuzzy obstacle avoidance for robotic manipulators, *Neural Network World,* 6(2), 131-142

Althoefer, K. & Fraser, D. A. (1996). Robotic manipulators amidst moving obstacles : Fuzzy – based obstacle avoidance, in *Proc. of EUFIT '96 – The 4th European Congress on Intelligent Techniques and Soft Computing,* Aachen.

Arai, T.; Pagello, E. & Parker, L. E. (1999). (Guest Editorial), *IEEE Trans. on Robotics and Automation,* 15(5), 818-828.

Balch, T. & Arkin, R. C. (1998). Behavior – based formation control for multi-robot teams, *IEEE Trans. Robotics and Automation,* 14(6), 926-939.

Bayle, B.; Renaud, M. & Fourquet, J.-Y. (2003). Nonholonomic mobile manipulators: kinematics,velocities and redundancies, *J. Intell. and Robotic Systems,* 36(1), 45-63.

Beaufrere B. & Zeghloul, S. (1995). A mobile robot navigation method for using a fuzzy based method : Simulation and experimental results, *Intl. J. Robotics and Automation,* 10(3), 106-113.

Belkhous, S.; Azzouz, A.; Saad, M.; Nerguizian, C. & Nerguizian, V. (2005). A novel approach for mobile robot navigation with dynamic obstacles avoidance, *J. Intell. and Robotic Systems,* 44(3), 187-201.

Boem, H. R. & Cho, H. S. (1995). A sensor-based navigation for a mobile robot using fuzzy – logic and reinforcement learning, *IEEE Trans. Systems, Man and Cybernetics,* 25(3), 464-477.

Brooks, R. A. (1986). A robust layered control system for a mobile robot, *IEEE J. Robotics and Automation,* 2(1), 14-23, 1986.

Brown, M. (1994). *Neuro-Fuzzy Adaptive Modelling and Control,* Prentice-Hall, UK.

Dudek, G.; Jenkin, M.; Milios, E. & Wilkes, D. (1991). Robotic exploration as graph construction, *IEEE Trans. on Robotics and Automation, 7(6)*, 859-865.

ajjaji, A. El. (2004). A Ciocan and D. Hamad, Four wheel steering control by fuzzy approach, *J. Intell. and Robotic Systems, 41(2-3)*, 141-156.

Erden, M. S.; Leblebicioglu, K. & Halici, U. (2004). Multi – agent system – based fuzzy controller design with genetic tuning for a mobile manipulator robot in the hand over task, *J. Intell. and Robotic Systems, 39(3)*, 287-306.

Erdmann, M. & Lozano-Perez, T. (1987). On Multiple Moving Obstacles, *Algorithmica 2(4)*, 477-521.

Fibry, J. R. (1987). An Investigation into Reactive Planning in Complex Domains, *Proc. AAAI Conference.*

Fugimura, K. (1991). *Motion Planning in Dynamic Environments,* Springer, Tokyo.

Gat, E. (1991). Reliable Goal-Directed Reactive Control for Real – World Autonomous Mobile Robots, *PhD Dissertation,* Virginia Polytechnic Institute and state University.

Gil de Lamadrid, J. & Gini, M. (1990). Path Tracking through Uncharted Moving Obstacles, *IEEE Transactions on System, Man, and Cybernetics, 20(6)*, 1408-1422.

Griswold, N. C. & Elan, J. (1990). Control of Mobile Robots in the Presence of Moving Obstacles, *IEEE Transactions on Robotics and Automation 6(2).*

Hsu, H. C.-H. & Liu, A. (2005). Multiagent – based multi-team formation control for mobile robots, *J. Intell. and Robotic Systems, 42(4)*, 337-360.

Ishikawa, S. (1995). A method of autonomous mobile robot navigation by using fuzzy control, *Advanced Robotics, 9(1)*, 29-52.

Izumi, K. & Watanabe, K. (2000). Fuzzy behavior – based control trained by module learning to acquire the adaptive behaviors of mobile robots, *Mathem. and Computers in Simulation, 51(3/4)*, 233-243.

Jaitly, R.; Althoefer, K. & Fraser, D. (1996). From vision to path planning : A neural-based implementation, *Proc. 2nd international conference on engineering applications of neural networks,* 209-212, London, UK.

Jaitly, R. & Fraser, D. A. (1996). Automated 3D object recognition and library entry system, *Neural Network world, 6(2)*, 173-183.

Kamon, I. & Rivlin, E. (1997). Sensory-based motion planning with global proofs, IEEE Trans. Robotics and Automation, 13(6).

Khatib, O. (1986). Real – time obstacle avoidance for manipulators and mobile robots, *Internat. J. Robotics Res. 5(1)*, 90-98.

Kortenkamp, D. & Weymouth, T. (1994). Topological mapping for mobile robots using combination of sonar and vision sensing, *Proc. AAAI Conf.,* Menlo Park, CA, USA, 979-984.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence,* Prentice – Hall, Englewood Cliffs, NJ, 2-3, 314-316.

Kuipers, B. & Levitt, T. (1988). Navigation and mapping in large scale space, *AI Magaz.,* 9, 25-43.

Latombe, J. C. (1991). *Robot Motion Planning,* Kluwer, Boston.

Lozano-Perez, T. (1983). Spatial planning : A configuration space approach, *IEEE Trans. Comput.* 32(2), 108-120.

Mamdani, E. H. (1974). Applications of fuzzy algorithms for simple dynamic plant, *Proc. IEE* 121(12), 1585-1588.

Mamdani, E. H. & Assilian, S. (1981). An experiment in linguistic synthesis with a fuzzy logic controller, in : *Fuzzy Reasoning and its Applications, Academic Press, New York,* 311-323.

Martinez, A.; Tunstel, E. & Jamshidi, M. (1994). Fuzzy – logic based collision – avoidance for a mobile robot, *Robotica,* 12 (6), 521-527.

Moustris, G. & Tzafestas, S. G. (2005). A robust fuzzy logic path tracker for non-holonomic mobile robots, *Intl. J. Artif. Intelligence Tools,* 14(6), 935-965.

Parhi, D. R. (2005). Navigation of mobile robots using fuzzy logic, *J. Intell. and Robotic Systems,* 42(3), 253-273.

Pedrycz, W. (1995). *Fuzzy Sets Engineering,* CRC Press, Boca Raton, FL, 3-7.

Ratering, S. & Gini, M. (1995). Robot Navigation in a Known Environment with Unknown Obstacles, *Autonomous Robots,* 149-165.

Rigatos, G. G.; Tzafestas, C. S. & Tzafestas, S. G. (2000). Mobile robot motion control in partially unknown environments using a sliding mode fuzzy controller, *Robotics and Autonomous Systems,* 33(1), 1-11.

Ryu, B. S. & Yang, H. S. (1999). Integration of reactive behaviors and enhanced topological map for robust mobile robot navigation, *IEEE Trans. on Systems, Man and Cybernetics,* 29(5), 474-485.

Seraji, H. (1998). A unified approach to motion control of mobile manipulators, *Intl. J. Robotics Research,* 17(2), 107-118.

Seraji H. & Howard, A. (2002). Behavior – based robot navigation on challenging terrain : A fuzzy logic approach, *IEEE Trans. Robotics and Automation,* 18(3), 308-321.

Stamou G. B. & Tzafestas, S. G. (2000). Neural fuzzy relational systems with a new learning algorithm, *Mathem. and Computers in Simulation,* 51(3-4), 301-314.

Sugeno, M. (1985). An Introductory Survey of Fuzzy Logic, *Information Science,* 36, 59.

Sugeno, M. & Murakami, K. (1984). Fuzzy Parking Control of Model Car, *Proc. 23rd IEEE Conf. on Decision and Control,* 902, Las Vegas, USA.

Sugeno, M. & Nishida, M. (1985). Fuzzy Control of a Model Car, *Fuzzy Sets and Systems,*16, 103-113.

Thrun, S. (1999). Learning metric-topological maps for indoor mobile robot navigation, *Artificial Intelligence,* 1, 21-71.

Topalov, A. V. & Tzafestas, S. G. (2001). Fuzzy – neural – genetic layered multi – agent reactive control of robotic soccer, In : *D. Braha (ed.), Data Mining for Design and Manufacturing,* Kluwer, Dordrecht / Boston, 417-422.

Tzafestas, S. G. (1999) (ed.), Advances in Intelligent Autonomous Systems, *Kluwer,* Dordrecht / Boston.

Tzafestas, S. G; Melfi, A. & Krikochoritis, T. (2000). Modelling and Control of an Omnidirectional Mobile Manipulators, *Proc. BASY '2000 : 4th IEEE/IFIP Intl. Conf. on Inf. Technology for Balanced automation Systems,* Berlin, Sept. 27-29.

Tzafestas, C. S. & Tzafestas, S. G. (2001). Full-state modeling, motion planning and control of mobile manipulators, *Studies in Informatics and Control,* 10(2), 109-127.

Tzafestas, S. G. & Zavlangas, P. G. (1999). Adaptive neuro-fuzzy navigation for industrial manipulators, *Proc. 1999, ICIMS-NOE Advanced Summer Institute : Production Management, Control and Supervision (ASI '99),* Leuven, Belgium.

Tzafestas S. G. & Zikidis, K. C. (2001). NeuroFast : On – line Neuro-Fuzzy ART-based Structure and Parameter Learning TSK Model, *IEEE Trans. Syst., Man, and Cybernetics,* Part B : Cybernetics, 31(5).

Tzafestas, S. G. & Venetsanopoulos, A. N. (1991) (eds.), *Fuzzy Reasoning in Information, Decision and Control Systems,* Kluwer, Dordrecht / Boston, 1994.

Wang, P. K. C. (1991). Navigation strategies for multiple autonomous mobile robots moving in formation, *J. Robotic Systems,* 8(2), 177-195.

Wang, L. X. (1994). *Adaptive Fuzzy Sets and Control : Design and Stability Analysis,* Prentice Hall Int., U.K.

Watanabe, K.; Izumi, K.; Maki, J. & Fujimoto, K. (2005). A fuzzy behavior – based control for mobile robots using adaptive fusion units, *J. Intell. and Robotic Systems,* 42(1), 42-27.

Watanabe, K.; Tang, J.; Nakamura, M.; Koga, S. & Fukuda, T. (1996). A fuzzy – Gaussian neural network and its application to mobile robot control, *IEEE Trans. Control Systems Technol.,* 4(2), 193-199.

Yamaguchi, H.; Arai, T. & Beni, G. (2001). A distributed control scheme for multiple robotic vehicles to make group formations, *Robotics Auton. Syst.,* 36(4), 125-147.

Yen, J. & Pflunger, N. (1992). A Fuzzy Logic Based Robot Navigation System, *Proc. AAAI Fall Symposium on Mobile Robot Navigation,* 195-199, Boston / MA, USA.

Zadeh, L. (1965). Fuzzy sets, *Inform. & Control*, 8, 338-353.

Zadeh, L. (1973). Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Trans. Systems Man Cybernetics,* 1, 28-44.

Zavlangas, P. G. & Tzafestas, S. G. (2000). Industrial robot navigation and obstacle avoidance employing fuzzy logic, *J. Intell. and Robotic Systems,* 27(1-2), 85-97.

Zavlangas, P. G. & Tzafestas, S. G. (2001). Integrated fuzzy global path following and obstacle avoidance for mobile robots, *Systems Science,* 27(4), 85-96.

Zavlangas, P. G. & Tzafestas, S. G. (2002). Integration of topological and metric maps for indoor mobile robot path planning and navigation, *Image Processing and Communications,* 8(1), 55-64.

11

Trajectory Planning and Control of Industrial Robot Manipulators

S. R. Munasinghe and Masatoshi Nakamura

1. Introduction

Industrial robot manipulators are used in various applications in order to achieve fast, precise, and quality production. In pick-and-place operations such as part handling, assembly, etc., the end-effector of the manipulator has to travel between two specific points in the workspace, and the path it takes in between is of no concern. In trajectory tracking applications such as welding, cutting, painting, etc., the end-effector has to follow a specific trajectory in 3-space as closely as possible, while maintaining rated velocity as much as possible (Munasinghe, 2001). In the latter case, planning the trajectory can be complex when there are constraints on the end-effector velocity, joint acceleration, and trajectory error. Trajectories planned without proper consideration to these constraints often result in poor performance such as trajectory over-shoots, end-effector deviations from the planned trajectory, and undue velocity fluctuations (Nakamura, et. al., 2000). Performance could be even more deteriorated especially at sharp corners in the Cartesian trajectory (Nakamura, 20001). Lot of trajectory planning algorithms have been proposed so far starting from simple Cartesian path control (Paul, 1979) to time optimized trajectories (Shin, 1985). However, the industrial systems experience difficulties accommodating most of these methods because of at least two specific reasons; 1) These techniques often require hardware changes in the existing setup and the manufacturing process has to be interrupted for system reconfigurations, which usually takes a longer period of time, and 2) Many of these methods often consider only one constraint, and often they pay less concern about industrial requirements and actual constraints set by applications. Therefore, they find difficulties in industrial implementation.

In this view, we present a new trajectory planning algorithm which considers end-effector velocity limit, joint acceleration limit, and error tolerance set by the application. These are the actual constraints in most industrial applications. Another technical problem in industrial manipulators is their delay dynamics, which causes the end-effector to overshoot at trajectory corners. To remedy this problem, we have designed a feed-forward compensator (Goto,

1997), which slightly alters the corners of the trajectory so that to make sure that the end-effector actually traces the trajectory even with the presence of the delay dynamics. The new trajectory planning algorithm together with the feed-forward compensator appears as a single front end block in the control system, and it can be easily accommodated to existing industrial manipulator systems without taking the risk and time of hardware reconfigurations.

A trajectory planning algorithm can generate position, velocity, and acceleration profiles for all of the joints of the manipulator. In most industrial manipulators, the system input is the joint position data, which are widely known in the industry as taught data. Paul (Paul, 1979) described how homogeneous transformations (Mittal & Nagrath, 2003) can be used in representing position and orientation of a serial link manipulator in order to control it through a Cartesian trajectory. The work by Shin et. al. (Shin et al. 1985) looks similar to ours, however it is difficult to be implemented in industrial systems as it needs to know many link/joint parameters of the manipulator. In industrial manipulator systems, most of these parameters are not precisely known.

In our previous works we have addressed acceleration and velocity constraints for 2-space trajectory planning (Munasinghe, 2001), and in this work we extend it to 3-space, while also considering error tolerance of the trajectory. The proposed method has been tested on a Performer MK-3s industrial manipulator, and its effectiveness has been experimentally verified.

2. Industrial Robot Manipulators

2.1 System Architecture

The industrial robot manipulator Performer MK-3s is shown below.

The reference input generator is a dedicated, or a networked computer which is connected to the servo controller through digital-to-analog (DAC) and analog-to-digital (ADC) converters. Servo controller has motor driver boards to control manipulator joints individually as shown in Fig.1. The reference input generator contains taught data sequences u_j where j=1, 2, 3 stands for the joint. Joint position θ_j is fed back to the reference input generator from the servo controller.
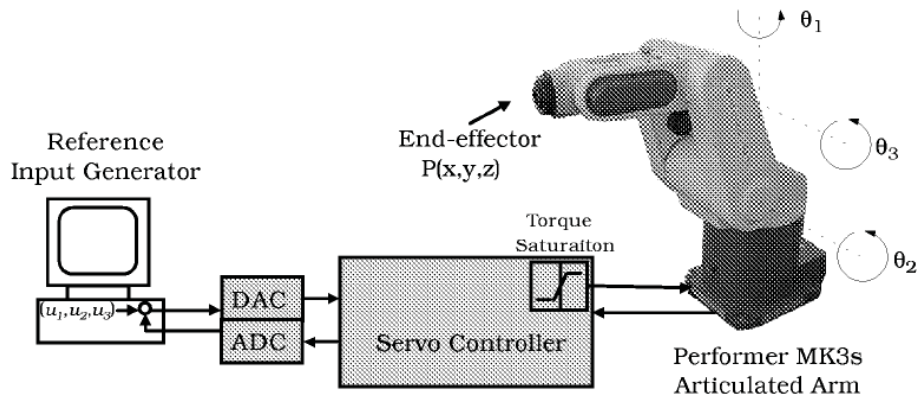
Figure 1. Performer MK-3s industrial robot manipulator

With the taught data and position feedback, reference input generator determines the control commands for each joint, and send those commands to the servo controller, which actuates joint motors accordingly. Refering to Fig.1, kinematics of the manipulator is given by

$$
\begin{aligned}
x &= [L_1 + L_2 \sin\theta_2 + L_3 \sin(\theta_2 + \theta_3)]\cos\theta_1 \\
y &= [L_1 + L_2 \sin\theta_2 + L_3 \sin(\theta_2 + \theta_3)]\sin\theta_1 \\
z &= \quad\quad L_2 \cos\theta_2 + L_3 \cos(\theta_2 + \theta_3)
\end{aligned}
\tag{1}
$$

where (x, y, z) is the end-effector position, and $\theta_1, \theta_2, \theta_3$ is the corresponding joint configuration. L_j is the length of link . By differentiating (1) it is possible to find the velocity relationship between Cartesian velocity and joint velocity as follows:

$$
\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \mathbf{J}(\boldsymbol{\theta}) \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}
\tag{2}
$$

where Jacobean is given by

$$
\mathbf{J}(\boldsymbol{\theta}) = \begin{bmatrix}
-\sin\theta_1[L_1 + L_2\sin\theta_2 + L_3\sin(\theta_2+\theta_3)] & \cos\theta_1[L_2\cos\theta_2 + L_3\cos(\theta_2+\theta_3)] & L_3\cos\theta_1\cos(\theta_2+\theta_3) \\
\cos\theta_1[L_1 + L_2\sin\theta_2 + L_3\sin(\theta_2+\theta_3)] & \sin\theta_1[L_2\cos\theta_2 + L_3\cos(\theta_2+\theta_3)] & L_3\sin\theta_1\cos(\theta_2+\theta_3) \\
0 & -L_2\sin\theta_2 - L_3\sin(\theta_2+\theta_3) & -L_3\sin(\theta_2+\theta_3)
\end{bmatrix}
$$

in that $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix}^T$ is the arm configuration.

2.2 Joint Dynamics of Industrial Robot Manipulators

Industrial robot manipuators are designed to meet the performance level required by the application such as welding, cutting, part handling, etc. The specifications in general are limited only to a certain degree of accuracy, velocity, and complexity. Therefore, most industrial robot manipulators are designed with linear proportional-integral-derivetive (PID) servo controllers with current limiting power amplifiers. This saturating current determines the acceleration limit of the joint. Furthermore, joints are independently controlled, whereas unknown inertia torques, coriolis and centrifugal torques, and torques due to friction and gravity are treated as disturbances to be rejected by the controller. To support this assumption, manipulator links are designed with low inertia, and joints are driven through high gear reductions (Sage et. al. 1999). These controllers are simple, and also provide sufficient robustness. Figure 2 illustrates three degree of freedom decoupled joint dynamic model of an industrial manipulator.
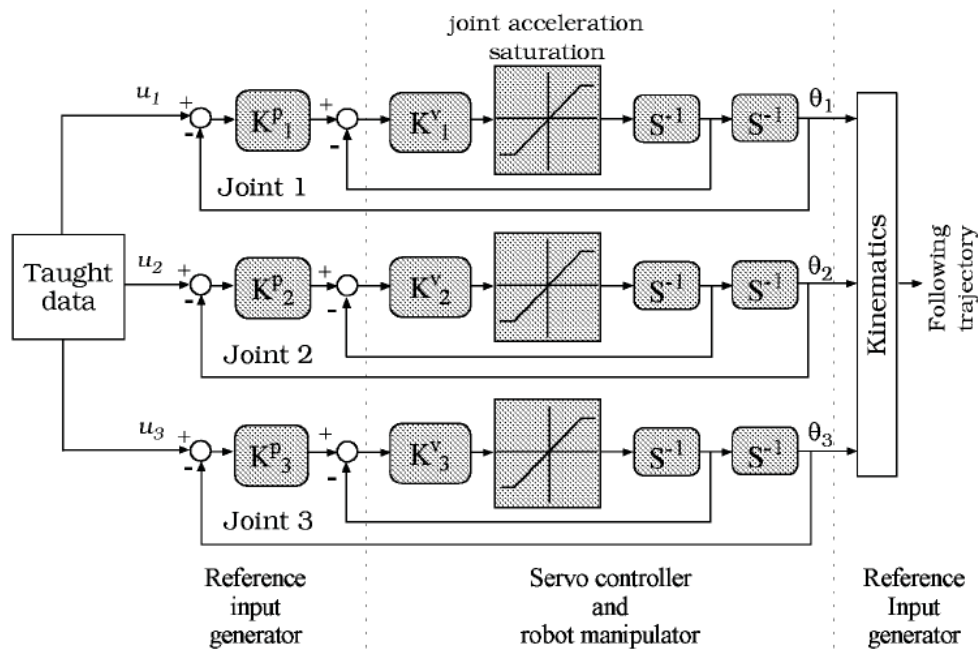
Figure 2. Three degree of freedom joint dynamic model of an industrial robot manipulator

This model also includes power amplifier saturation of joint actuators. K_j^p and K_j^v are the servo controller gains in the position loop and velocity loop of joint j, and these gains are periodically tuned by the trained operators to maintain the level of performance. As only two tunning prameters are involved, controller tuning process is quite simple. Within the linear reagion of joint ac-

celeration, joint dynamics is given by

$$\frac{\Theta_j(s)}{U_j(s)} = \frac{K_j^p K_j^v}{s^2 + 2K_j^v s + K_j^p K_j^v}$$

(3)

And, the joint dynamics when joint acceleration saturates is given by

$$\ddot{\theta}_j(t) = \text{sat}[K_j^v \{K_j^p (u_j(t) - \theta_j(t)) - \dot{\theta}_j(t))\}]$$

(4)

where

$$\text{sat}(z) = \begin{cases} \ddot{\theta}_j^{\max} & z > \ddot{\theta}_j^{\max} \\ z & |z| \le \ddot{\theta}_j^{\max} \\ -\ddot{\theta}_j^{\max} & z < -\ddot{\theta}_j^{\max} \end{cases}$$

in that $\ddot{\theta}_j^{\max}$ is the mximum aceleration of joint j. In this view, the objective of the trajectory planning is to make the best use of joint acceleration capability, while avoiding saturation.

2.3 Problem Statement

In this work, we consider the following three major issues which are practically applicable in industrial robot manipulator applications.

$$|\ddot{\theta}_j| \le \ddot{\theta}_j^{\max}$$

(5)

$$v = \begin{cases} \le v_r & \text{along straight line} \\ \le v_t^{\max} & \text{at corners} \end{cases}$$

(6)

$$e \le \rho$$

(7)

where v, v_r, and $v_t^{\max}$ are end-effector velocity, rated velocity, and maximum tangential velocity (at a rounded corner), respectively. e, and ρ are the trajectory error and error tolerance. Constraint (5) describes the linear region for joint acceleration, within which linear dynamics (3) is maintained. A violation of this coinstraint results in nonlinear joint dynamics (4), which causes the end-effector to deviate from the planned trajectory. Consraint (6) specifies the velocity limit while end-effector moves along straight lines and through corners. Rated velocity of the joint ω_r is given by $\omega_r = 2.\pi.RPM_r /(60.N_G)$, where RPM_r is the rated RPM (revolutions per minute) of the joint and N_G is the gear reduction ratio. Then, the rated velocity $v_r = \omega_r L$, where L is the link length.

At trajectory corners the tangential velocity is lowered heuristically to maintain centripetal acceleration within constraint (5), and it can also be theoretiaclly determined as described in (Munasinghe & Nakamura, 2002)

3. Trajectory Planning

3.1 The Algorithm

The proposed trajectory planning algorithm is illustrated in Fig. 3.
The objective trajectory $O(s)$ is specified by the application, and it is segmented into a sequence of a) corners and b) straight line segments. Corners are planned in Cartesian space using specified tangential velocity $v = v_t^{max}$, and transformed into joint space using inverse kinematics. Straight line segments are generated in joint space as piecewise in that every straight line segment has three pieces; forward(acceleration), middle(uniform velocity), and reverse(deleleration). Forward/reverse pieces are planned in such a way that at least one joint moves with its maximum acceleration/deceleration as long as end-effector velocity constraint (6) is not violated (b1 and b2 in Fig.3).
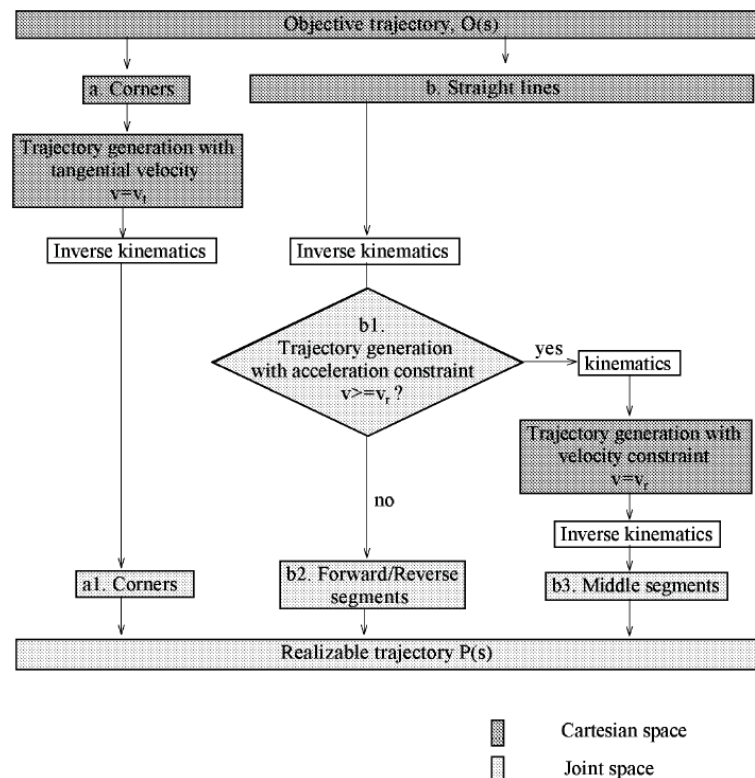
Figure 3. Proposed trajectory planning algorithm

Middle segment is planned in Cartesian space maintaining uniform end-effector velocity $v = v_r$, and then it is transformed into joint space (b3 in Fig. 4). Finally, all corners and straight line segments in joint space are merged in the correct sequence. This trajectory in joint space is called the realizable trajectory *P(s)*.

3.2 Trajectory Planning for a Corner

Figure 4(a) illustrates a sharp corner of the objective trajectory *O(s)* with the error tolerance (dashed line). Error tolerances are quite common in industrial applications, and it can be used to effectively plan the realizable trajectory *P(s)*. It is required however, to make sure that the realizable trajectory is contained within the error tolerance. Referring to Fig. 4, the largest possible circular arc should pass through point *R*, and it should be tangential to the section of the tolerance cylinder on the plane of ΔABC. In order to construct this curve, points $A', B',$ and C' are determined from $A, B,$ and C according to the following procedure:

In ΔABC $AB = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}$,

and $\beta = \cos^{-1}\{(AB^2 + BC^2 - AC^2)/2.AB.BC\}$. A' is located using point coordinates of A and C. For exmple, x coordinate of A' can be determined by $x_{A'} = (x_A - x_C)\varepsilon_{A'} / AC + x_A$, where $\varepsilon_{A'} = \rho / \sin \hat{A}$. Adopting the same procedure B' could be located with point coordina tes of $F, B,$ and $\varepsilon_{B'} = \rho / \sin(\beta / 2)$. C' could also be located with point coordinates of *A, C,* and $\varepsilon_{CB} = \rho / \sin \hat{C}$.

F is located with point coordinates

$A, C,$ and $\varepsilon_F = BC \tan \hat{C} \tan(\beta / 2) / \{\tan \hat{C} + \tan(\beta / 2)\}$.

Figure 4(b) illustrates the circular arc constructed at a corner with radius *r* is given by

$$r = 2\rho / \{1 - \sin(\beta / 2)\} \tag{8}$$
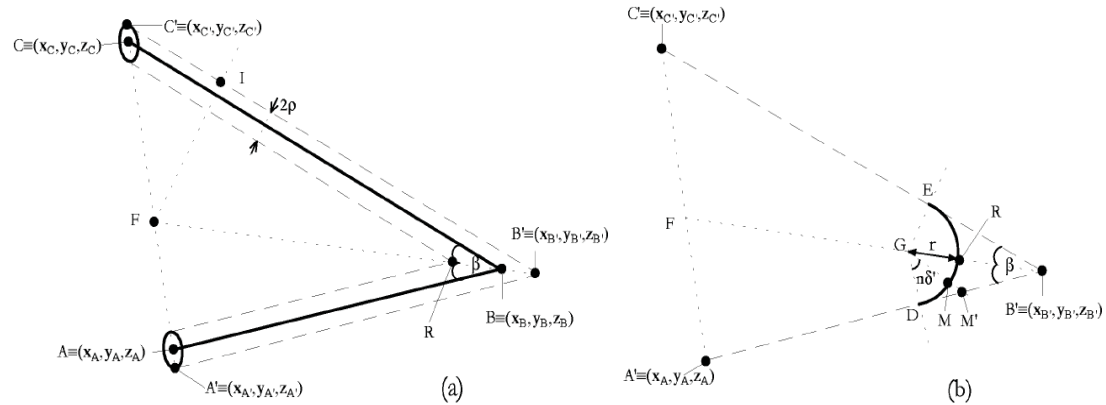
Figure 4 (a) A sharp corner of the objective trajectory, and (b) A planned corner of the realizable trajectory

D and E are the terminal points of the circular arc and they could be located on $A'B'$ and $B'C'$ as $DB' = EB' = \{r + 2\rho / \sin(\beta/2)\}\cos(\beta/2)$. Along the circular arc, from D to E trajectory is sampled at each δ as marked by M. Sampling angle is given by

$$\delta = v_t^{max} t_s / r \qquad (9)$$

where t_s is the sampling interval. Number of sampling points is determined by upward roounding of $N = (\pi - \beta)/2\delta$ to the closest whole number. Then, sampling angle is readjusted by $\delta' = (\pi - \beta)/2N$. M' could be located on BD' as $DM' = r\tan(n\delta')$. M could be located on GM' since the ratio r/GM' is known.

3.3 Trajectory planning for a Straight Line

3.3.1 Forward and Reverse Segments

Figure 6 illustrates details of straight line trajectory planning. P_1P_2 is the straight line segment of the objective trajectory, for which a realizable trajectory has to be planned. The two end points of the straight line are either start and end ponts of the objective trajectory, or terminal points of a circular arc. Either way, position and velocity at these points are known. From P_1 to P_2, trajectory is segmented by equidistance via points indexed by $k = 0,1,2 \ldots$ The forward trajectory is planned from P_1 to P_2, whereas the reverse trajectory is generated from P_2 to P_1. Both segments are planned in joint space in the two directions using the same algorithm described below.
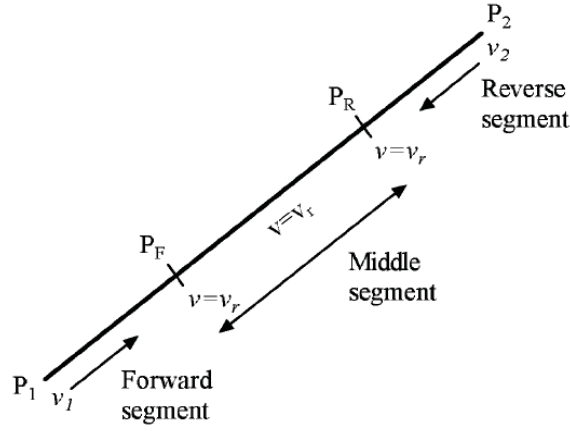
Figure 5. Trajectory planning for straight line segments

The minimum time $t_j^{\min}(k)$ each joint takes to move from k th via point to $(k+1)$th via point is given by

$$t_j^{\min}(k) = \begin{cases} \dfrac{\sqrt{\dot{\theta}_j^2(k) + 2\ddot{\theta}_j^{\max}\Delta\theta_j(k)} - \dot{\theta}_j(k)}{\ddot{\theta}_j^{\max}} & \text{if} \quad \Delta\theta_j(k) > 0 \\[3mm] \dfrac{\sqrt{\dot{\theta}_j^2(k) + 2\ddot{\theta}_j^{\max}\Delta\theta_j(k)} + \dot{\theta}_j(k)}{\ddot{\theta}_j^{\max}} & \text{if} \quad \Delta\theta_j(k) < 0 \end{cases} \qquad (10)$$

where $\Delta\theta_j(k) = \theta_j(k+1) - \theta_j(k)$. The minimum feasible time between the two via points without letting any of the joints to saturtion is the longest $t_j^{\min}$ considering all of the joints as follows.

$$t^{\min}(k) = \max_j \left\{ t_j^{\min}(k) \right\} \qquad (11)$$

Using $t^{\min}(k)$ for planning the trajectory between the two via points guarantees minimum time motion, and resulting joint accelerations are

$$\ddot{\theta}_j(k) = \frac{2(\Delta\theta_j(k) - \dot{\theta}_j(k)t^{\min}(k))}{\{t^{\min}(k)\}^2} \qquad (12)$$

Then, the trajectory is planned in joint coordinates as follows.

$$\dot{\theta}_j(k,t) = \dot{\theta}_j(k) + \ddot{\theta}_j(k)t \qquad\qquad ; kt < t < (k+1)t \qquad (13)$$

$$\theta_j(k,t) = \theta_j(k) + \dot{\theta}_j(k)t + 0.5\ddot{\theta}_j(k)t^2 \; ; kt < t < (k+1)t \qquad (14)$$

This algorithm continues as via point advances $k=$ 0, 1, 2…, and in the same time end-effector velocity is calculated using (2) together with (13) and (14). When the end-effector velocity reaches rated velocity the algorithm terminates (b1 in Fig. 3). As illustrated in Fig. 5, end-effector reaches the rated velocity at P_F in the forward direction, and at P_R in the reverse direction.

3.3.2 Middle Segment

Referring to Fig. 5 $P_F P_R$ is the middle segment of the straight line. This segment is planned in Cartesian space by maintaining rated velocity v_r as follows.

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} v_r^x \\ v_r^y \\ v_r^z \end{bmatrix} + \begin{bmatrix} x(P_F) \\ y(P_F) \\ z(P_F) \end{bmatrix} \tag{15}$$

where v_r^x, v_r^y, and v_r^z are the velocity components of v_r along major axes, and $x(P_F)$, $y(P_F)$ and $z(P_F)$ are the cartesian position coordinates of P_F. Middle segment is transformed into joint coordinates using inverse kinematics.

3.4 Compensation of Delay Dynamics

Planned corners and straight lines are merged to form the realizable trajectory. As shown in Fig. 3, taught data is obtained by compensating realizable trajectory for delay dynamics. In (Goto et. al., 1997) pole placement with linear state feedback were used to develop a feed-forward delay compensator as described by

$$F_j(s) = -\frac{a_3 s^3 + a_2 s^2 + a_1 s + a_0}{(s - \mu_1)(s - \mu_2)(s - \gamma)} \tag{16}$$

Where

$$a_o = -\mu_1 \mu_2 \gamma$$

$$a_1 = (K_j^v + \gamma)(\mu_1 + \mu_2) + K_j^{v^2} + \mu_1 \mu_2 + K_j^v \gamma - \mu_1 \mu_2 \gamma / K_j^p$$

$$a_2 = \frac{1}{K_j^p} \left\{ (K_j^v + \gamma)(\mu_1 + \mu_2) + K_j^{v^2} + \mu_1 \mu_2 + K_j^v \gamma \right\} - \frac{\mu_1 \mu_2 \gamma}{K_j^p K_j^v}$$

$$a_3 = \frac{1}{K_j^p K_j^v} \left\{ (K_j^v + \gamma)(\mu_1 + \mu_2) + K_j^{v^2} + \mu_1 \mu_2 + K_j^v \gamma \right\}$$

in that μ_1 and μ_2 are the regulator poles, and γ is the observer pole. These poles can be tentatively tuned for better performance. A theoretical determination of compensator poles can be found in (Munasinghe & Nakamura, 2003)

4. Results and Discussion

4.1 Experimental Conditions

The objective trajectory was set as follows: start (0.35, 0, 0.1)[m], first corner(0.41, 0.1, 0.15)[m], second corner (0.28, -0.1, 0.3)[m], and end (0.35, 0, 0.35)[m]. Rated velocity and tangential velocity were set to $v_r = 0.15$ [m/s²] and $v_t^{max} = 0.02$ [m/s²]. Maximum joint acceleration for all joints were set to $\ddot{\theta}_j^{max} = 0.72$ [rad/s²]. Trajectory error tolerance was set to $\rho = 0.001$ [m]. Servo controller gains were set to $K_j^p = 15$ [1/s] and $K_j^v = 15$ [1/s]. In the delay compensator, regulator poles were set to $\mu_1 = \mu_2 = -60$, and the observer pole was set to $\gamma = -200$. In order to compare performance of the new method, we simulated a conventional trajectory planning algorithm in that uniform end-effector velocity of 0.05 [m/s] was used to plan the trajectory in Cartesian space through the mentioned objective trajectory.

4.2 Evaluation of Results

Figure 6 illustrates the results.
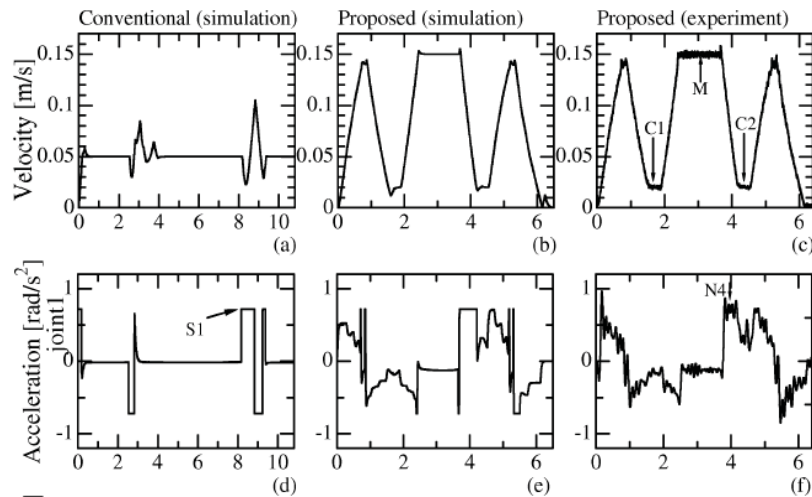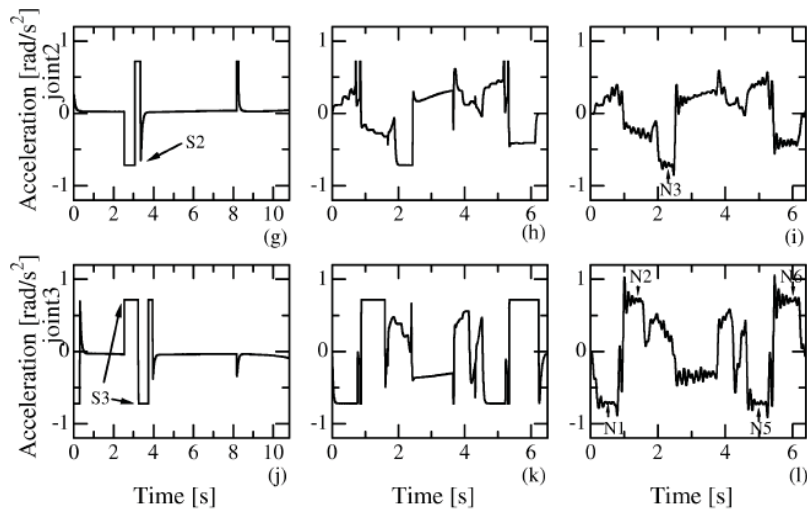
Figure 6. End-effector velocity and joint acceleration profiles under the control of conventional and proposed methods.

One important observation to be made is the close similarity between simulation and experimental results of the proposed method, i.e., the experiment produces end-effector velocity and joint acceleration profiles that are very similar to what is obtained by the simulation under the assumption of linear decoupled dynamics. This conveys the validity of the trajectory planning and delay compensation used in the proposed method.

The arrow sequence ↑N1, ↓N2, ↑N3, ↓N4, ↑N5, ↓N6 confirms that at least one of the three joints moves with its maximum acceleration or deceleration within the entire motion, except at corners C1, C2, and middle segment M. End-effector velocity has been kept on or below the rated velocity within the entire motion. On the contrary, the conventional method shows in its simulation a significant saturation in joint acceleration profiles as indicated by S1, S2, and S3.

Figure 7 shows the motion of the end-effector in 3-space with projections to X-Y, Y-Z, and Z-X planes. In Fig. 7(b), huge trajectory errors are observed at corner C1 as a result of acceleration saturation in joint 2 and joint 3 as indicated by S2 and S3. Similar errors are resulted at corner C2 due to acceleration saturation in joint 1 as indicated by S1. On the other hand, proposed method has made the end-effctor accurately follow the objective trajectory.

4.3 Discussion

The proposed trajectory planning algorithm takes the crude objective trajectory and the constraints for velocity, acceleration and error tolerance, and plans the realizable trajectory. The realizable trajectory is compensated for delay dynamics. The proposed method brings the best possible performance as it

always maintains at least one of the given constraints (5), (6), or (7) within the entire motion.
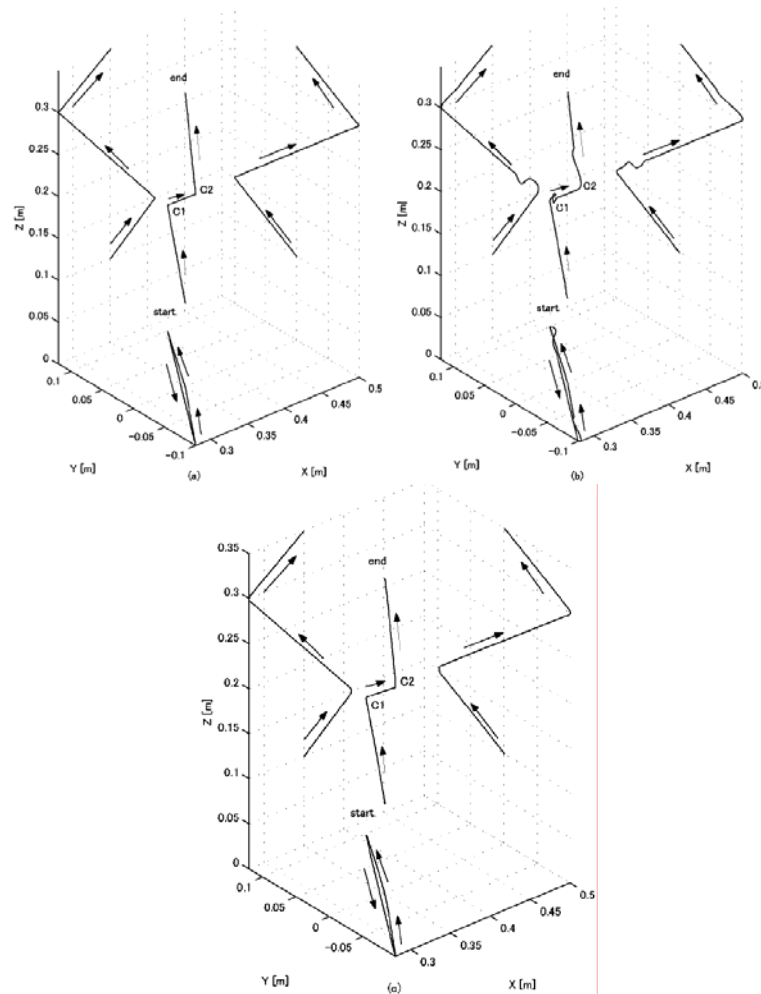
Figure 7. Profiles of the end-effector in 3-space, (a) objective trajectory, (b) resulting motion under conventional method (simulation), and (c) resulting motion under proposed method.

5. Conclusion

This chapter presented a new trajectory planning algorithm for industrial robot manipulators. This algorithm considers joint acceleration constraint, rated end-effector velocity, and trajectory error tolerance, and plans the realizable trajectory accordingly so that all these constraints are maintained in the best possible manner during the entire motion. A feed-forward compensator is also used to compensate the realizable trajectory against delay dynamics of the joints. The method was successfully tested on Performer MK-3s industrial robot manipulator using a complex three dimensional trajectory, in that very ac-

curate motion was realized without violating any of the constraints. The proposed method appears as a single feed-forward block in the control system, therefore, it could be conveniently incorporated into existing industrial manipulators without undertaking a significant cost or risk.

6. References

Goto, S.; Nakamura, M., & Kyura, N.; (1997). Modified taught data method for industrial mechatronic servo-controller to achieve accurate contour control performance, *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 525B, June 1997

Mittal, R. K. & Nagrath I. J. (2003). Chapter 2, In: *Robotics and Control,* 35-69, Tata McGraw Hill, 0-07-048293-4, New Delhi

Munasinghe, S. R. & Nakamura, M. (2001). Optimum contouring of industrial robot arms under assigned velocity and torque constraints. *IEEE Transactions on Systems, Man, and Cybernetics-Part C,* Vol. 31, No. 2., (May, 2001) 159-167

Munasinghe, S. R. & Nakamura, M.; (2002). Determination of maximum tangential velocity at trajectory corners in robot manipulator operation under torque constraint, *Proceedings of the Annual Conference of the Society of Instrumentation and Control Engineers(SICE),* MA17-2, 1170-1175, August, 2002

Munasinghe, S. R., Nakamura, M., Goto S., & Kyura, N., (2003). Pole selection of feedforward compensators considering bounded control input of industrial mechatronic systems. *IEEE Transactions on Industrial Electronics,* Vol. 50, No. 6, (December, 2003) 1199-1206

Nakamura, M., Munasinghe, S. R., Goto, S., and Kyura, N., (2000). Enhanced contour control of SCARA robot under toque saturation constraint. *IEEE/ASME Transactions on Mechatronics,* Vol. 5, No. 4, (December, 2000) 437-440

Paul, R. (1979). Manipulator Cartesian path control. *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 9, No. 11., (Nov., 1979) 702-711

Sage, H. G., De Mathelin M. F., & Ostertag F., (1999). Robust control of industrial manipulators: a survey. *International Journal of Control,* Vol. 72, No. 16, (Nov., 1999) 1498-1522

Shin, K. G., & Mckay N. D., (1985). Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control,* Vol. 30, No. 6, (Jun., 1985) 531-541

Collision Free Path Planning
for Multi-DoF Manipulators

Samir Lahouar, Said Zeghloul and Lotfi Romdhane

1. Introduction

Path planning is a very important issue in robotics. It has been widely studied for the last decades. This subject has gathered three interesting fields that were quite different in the past. These fields are robotics, artificial intelligence and control. The general problem of path planning consists of searching a collision free trajectory that drives a robot from an initial location (position and orientation of the end effector) to a goal location. This problem is very wide and it has many variants such as planning for mobile robots, planning for multiple robots, planning for closed kinematic chains and planning under differential constraints. It includes also time varying problems and molecular modeling, see (LaValle, 2006) for a complete review. In this study we focus on the case of multi-Degrees of Freedom (DoF) serial manipulators.

The first works on serial manipulators path planning began in the seventies with Udupa (Udupa, 1977), then with Lozano-Pérez and Wesley (Lozano-Pérez & Wesley, 1979) who proposed solving the problem using the robot's configuration space (CSpace). Since then, most of path planning important works have been carried out in the CSpace. There are two kinds of path planning methods: Global methods and Local methods. Global methods (Paden et al., 1989; Lengyel et al., 1990; Kondo, 1991) generally act in two stages. The first stage, which is usually done off-line, consists of making a representation of the free configuration space (CSFree). There are many ways proposed for that: the octree, the Voronoï diagram, the grid discretization and probabilistic roadmaps. For each chosen representation, an adapted method is used in order to construct the CSFree, see (Tournassoud, 1992; LaValle, 2006). The representation built in the first stage is used in the second one to find the path. This is not very complicated since the CSFree is known in advance. Global methods give a good result when the number of degrees of freedom (DoF) is low, but difficulties appear when the number of DoF increases. Moreover, these methods are not suitable for dynamic environments, since the CSFree must be recomputed as the environment changes. Local methods are suitable for robots with a high number of DoF and thus they are used in real-time applications. The

potential field method proposed by Khatib (Khatib, 1986) is the most popular local method. It assumes that the robot evolves in a potential field attracting the robot to the desired position and pushing its parts away from obstacles. Because of its local behavior these methods do not know the whole robot's environment, and can easily fall in local minima where the robot is stuck into a position and cannot evolve towards its goal. Constructing a potential field with a single minimum located in the goal position, is very hard and seems to be impossible, especially if there are many obstacles in the environment.

Faverjon and Tournassoud proposed the constraint method (Faverjon & Touranssoud, 1987), which is a local method acting like the potential field method in order to attract the end effector to its goal and dealing with the obstacles as constraints. Although it yields remarkable results with high DoF robots, this method suffers from the local minima problem.

Probabilistic methods were introduced by Kavraki et al. (Kavraki et al., 1996) in order to reduce the configuration free space complexity. These methods generate nodes in the CSFree and connect them by feasible paths in order to create a graph. Initial and goal positions are added to the graph, and a path is found between them. This method is not adapted for dynamic environments since a change in the environment causes the reconstruction of the whole graph. Several variants of these methods were proposed: Visibility based PRM (Siméon et al., 2000), Medial axis PRM (Wilmarth et al., 1999) and Lazy PRM (Bohlin & Kavraki, 2000).

Mediavilla et al. (Mediavilla et al., 2002) proposed a path planning method for many robots cooperating together in a dynamic environment. This method acts in two stages. The first stage chooses off-line, a motion strategy among many strategies generated randomly, where a strategy is a way of moving a robot. The second stage is the on-line path planning process, which makes each robot evolve towards its goal using the strategy chosen off-line to avoid obstacles that might block its way.

Helguera et al. (Helguera & Zeghloul, 2000) used a local method to plan paths for manipulator robots and solved the local minima problem by making a search in a graph describing the local environment using an A* algorithm until the local minima is avoided.

Yang (Yang 2003) used a neural network method based on biology principles. The dynamic environment is represented by a neural activity landscape of a topologically organized neural network, where each neuron is characterized by a shunting equation. This method is practical in the case of a 2-DoF robot evolving in a dynamic environment. It yields the shortest path. However, the number of neurons increases exponentially with the number of DoF of the robot, which makes this method not feasible for realistic robots.

Here, we propose two methods to solve the path planning problem. The first method (Lahouar et al., 2005a ; Lahouar et al., 2005b) can be qualified as a

global method. It is suitable for serial robot manipulators in cluttered static environments. It is based on lazy grid sampling. Grid cells are built while searching for the path to the goal configuration. The proposed planner acts in two modes. A depth mode while the robot is far from obstacles makes it evolve towards its goal. Then a width search mode becomes active when the robot gets close to an obstacle. This mode ensures the shortest path to go around an obstacle. This method reduces the gap between pre-computed grid methods and lazy grid methods. No heuristic function is needed to guide the search process. An example dealing with a robot in a cluttered environment is presented to show the efficiency of the method.

The second method (Lahouar et al., 2006) is a real-time local one, which is used to solve the path planning problem for many manipulator robots evolving in a dynamic environment. This approach is based on the constraints method coupled with a procedure to avoid local minima by bypassing obstacles using a boundary following strategy. The local planner is replaced by the boundary following method whenever the robot gets stuck in a local minimum. This method was limited to 2-DoF mobile robots and in this work we show how it can be applicable to a robot with n degrees of freedom in a dynamic environment. The path planning task is performed in the configuration space and we used a hyperplane in the n dimensional space to find the way out of the deadlock situation when it occurs. This method is, therefore, able to find a path, when it exists and it avoids deadlocking inherent to the use of the local method. Moreover, this method is fast, which makes it suitable for on-line path planning in dynamic environments.

2. Sampling and construction of the CSpace

Many planning algorithms need samples of CSpace in order to compute a trajectory. There are many ways of sampling; the easiest way is to use a grid with a given resolution. The number of the grid cells grows exponentially according to the number of DoF of the robot. In the same way, the time and the memory space required to compute and store the grid increase. Random sampling was introduced in order to reduce the number of samples needed to represent the CSpace. It consists of choosing random configurations and constructing a graph representing feasible paths between them. This method needs a long time of computation.

We give an example of sampling using a grid with a low resolution and we define constraints used to detect if there is a free path between two neighboring cells. On one hand, these constraints make the path between two neighboring cells in the CSfree safe even if the step is quite large, and on the other hand they speed up the collision checking process as the constraints computed in a cell are useful to check all the neighboring cells. There is no need to check for collision in all cells of the grid before starting to search for a path. The con-

straints calculated in a cell allow us to judge whether a path exists to a neighboring cell or not.
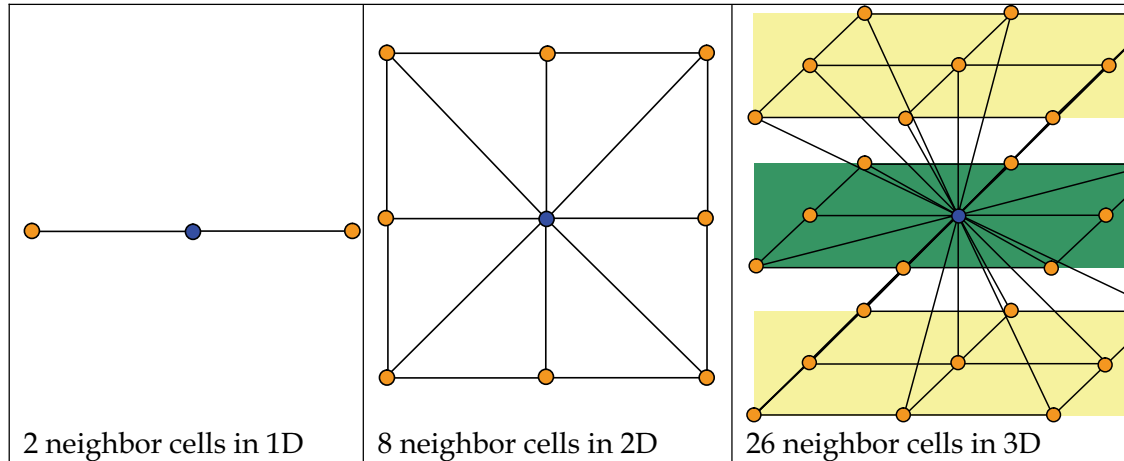
Figure 1. Each cell has 3^N-1 neighbors

Therefore, the constraints-calculating process is equivalent to 3^N-1 times the collision checking process, as a cell has 3^N -1 neighbors (Fig. 1). The number N represents the number of DoF of the robot.

3. Non-collision constraints

Here, we define non-collision constraints necessary to accelerate the global method (see paragraph 4) and useful for the local planner of the second method (see paragraph 5). Non-collision constraints as proposed by Faverjon and Tournassoud are written as follows:

$$\dot{d} \geq -\xi \frac{d - d_s}{d_i - d_s} \quad \text{if} \quad d \leq d_i \tag{1}$$

With d is the minimal distance between the robot and the object and $\dot{d}$ is the variation of d with respect to time. d_i is the influence distance from where the objects are considered in the optimization process, d_s is the security distance and ξ is a positive value used to adjust the convergence rate.
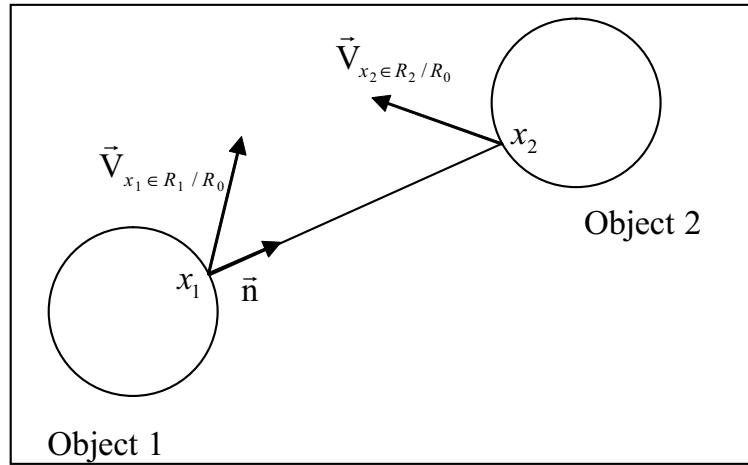
Figure 2. Two objects evolving together
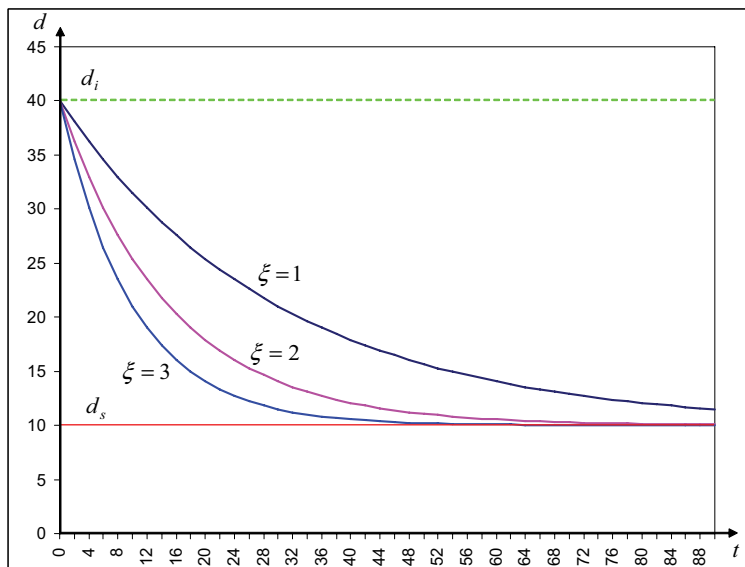
Figure 3. Evolution of the distance according to the convergence rate

If we consider two mobile objects in the same environment as shown in Fig. 2, $\dot{d}$ can be written as follows:

$$\dot{d} = V_{(x_2 \in R_2 / R_0)}^{T} . \mathbf{n} - V_{(x_1 \in R_1 / R_0)}^{T} . \mathbf{n} \tag{2}$$

Where $V_{(x_i \in R_i / R_0)}$ is the velocity vector evaluated at the point x_i of object i having the minimal distance with the second object and $\mathbf{n}$ is the unit vector on the line of the minimal distance.

The non-collision constraints, taking into account the velocities of objects, are written as:

$$V^T_{(x_1 \in R_1 / R_0)}.\mathbf{n} - V^T_{(x_2 \in R_2 / R_0)}.\mathbf{n} \le \xi \frac{d - d_s}{d_i - d_s} \tag{3}$$

A robot evolving towards an obstacle, if it respects constraints given by equation (1), it will evolve exponentially to the security distance without going closer than this distance (see Fig. 3).

Fig. 4 shows a PUMA robot placed next to a static obstacle. The constraint corresponding to that obstacle is written as:

$$V^T_{(x_1 \in R_1 / R_0)}.\mathbf{n} \le \xi \frac{d - d_s}{d_i - d_s} \tag{4}$$

By introducing $\mathbf{J}_{x_1}(q)$, the Jacobian matrix of the robot in configuration q defined in point x_1, we get:
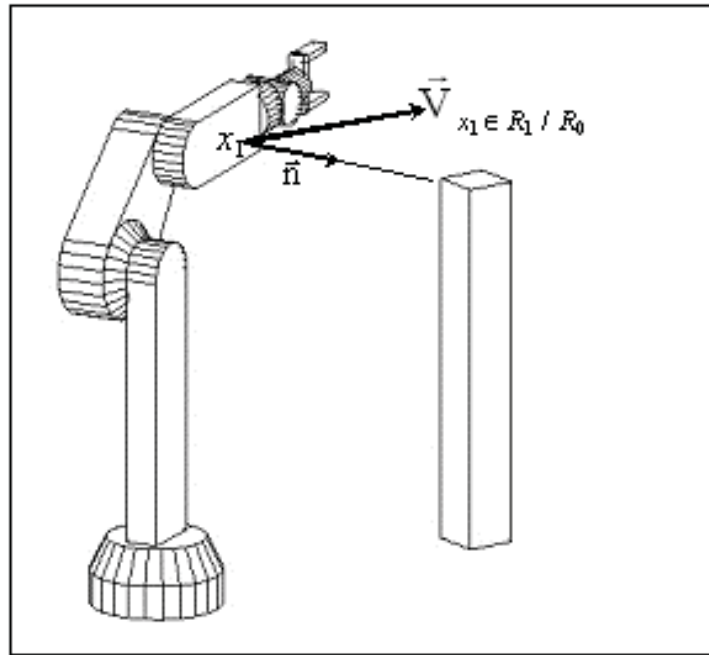
Figure 4.The distance between a robot and an obstacle

$$\mathbf{n}^T \mathbf{J}_{x_1}(q) \, \Delta q \le \xi \frac{d - d_s}{d_i - d_s} \tag{5}$$

Condition (5) will be written in the following manner:

$$[a_1 \quad \cdots \quad a_N][\Delta q_1 \quad \cdots \quad \Delta q_N]^T \le b \tag{6}$$

with $\begin{bmatrix} a_1 & \cdots & a_N \end{bmatrix} = \mathbf{n}^T \mathbf{J}$, $\Delta q = \begin{bmatrix} \Delta q_1 & \cdots & \Delta q_N \end{bmatrix}^T$ and $b = \xi \dfrac{d - d_i}{d_s - d_i}$

Figure 5. shows two PUMA robots evolving together. We consider that each robot is controlled separately.
In that manner, each robot is considered as a moving obstacle by the other one.
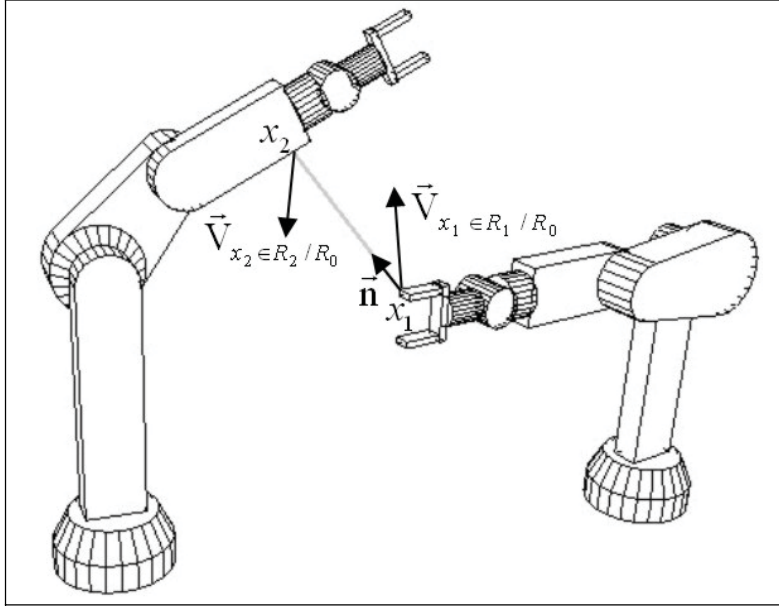
Figure 5. Two PUMA robots working in the same environment

The motion of the two robots must satisfy the following conditions:

$$V_{(x_1 \in R_1 / R_0)}^T . \mathbf{n} \le \xi' \frac{d - d_i}{d_s - d_i} \text{ and } -V_{(x_2 \in R_2 / R_0)}^T . \mathbf{n} \le \xi' \frac{d - d_i}{d_s - d_i} \tag{7}$$

Where $\xi' = \frac{1}{2}\xi$. While adding the two conditions of equation (7), we notice that the non-collision constraint defined by (3) is satisfied. So with a suitable choice of the parameters ξ, d_i and d_s, it is possible to use only condition (5) to avoid collisions with all objects in the environment.
In the next paragraph, we propose an approach that does not construct the whole grid, representing the CSpace. Only cells necessary to find the path to the goal position are checked for collision.

4. Path planning in static cluttered environments

The planner we propose uses two modes. The first one makes the robot evolve towards its goal position if there is no obstacle obstructing its way and the second mode is active near the obstacles and enables the robot to find the best way to avoid them. This latter mode is the most important as it needs to generate all the cells near the obstacle until it is avoided. For this reason, we do not have to store all the cells but just the ones near the obstacles which are sufficient to describe the CSfree.

4.1 Definitions

In order to explain the algorithm of this method, we need to define some terms.

A Cell
The algorithm we propose is based on a "Cell" class in terms of object oriented programming. A cell c_i is made of:

***A pointer to the parent cell (c_i.parent)*:**
the path from the initial configuration to the goal is made of cells. Each one of these cells has a pointer to the parent cell, that generated it previously. Starting from a cell, the next one in the path is the one that is closest to the goal and respecting the non-collision constraints. When the goal cell is reached the algorithm stops and the path is identified by all the selected cells.

***A configuration defining a posture of the robot*:**
each cell corresponds to a point in the CSpace. If a cell configuration is written as $q_1 = \begin{bmatrix} q_1^1 & \cdots & q_N^1 \end{bmatrix}^T$ where N is the number of DoF of the robot, and let Δq be the step of the grid, the neighboring cells are then defined as the configurations belonging to the following set:

$$Vic(q_1) = \left\{ q = \begin{bmatrix} q_1^1 + s_1\Delta q & \cdots & q_N^1 + s_N\Delta q \end{bmatrix}^T ; (s_1,\ldots,s_N) \in \{-1,0,1\}^N /(0,\ldots,0) \right\} \qquad (8)$$

***A distance to the goal(c_i.distance_to_goal)*:**
it represents the distance in configuration space between the goal configuration and the cell configuration. This distance allows the planner's first mode to choose the closest cell to the goal configuration. While the robot is far from obstacles, the shortest path to the goal configuration is a straight line in CSpace.

***A boolean "collision" variable (c_i.collision)*:**
it takes false if the cell verifies the constraints and true if it does not.

A boolean "computed" variable(c_i.computed):
used by the planner in order to know whether the cell has already been used to search for the path or not.
A boolean "near an obstacle" variable (c_i.near_an_obstacle):
used by the second mode of the planner allowing it to stay stuck to the obstacle while performing its width search in order to find the best direction to go around the obstacle.

Queue

Another important item in our approach is the Queue, Q, which is defined as an ordered set of cells. The first cell in the Queue is named head and denoted h(Q). While the last cell is the tail of the Queue and denoted t(Q). If the Queue is empty we write $h(Q) = t(Q) = \emptyset$.

In order to handle the Queue Q, we use some operators that we define here.
$h^+(Q, c_1)$ adds the cell c_1 to the head of Q.
$t^+(Q, c_1)$ adds c_1 to the tail of Q.
$h^-(Q)$ removes the head cell from Q.
$t^-(Q)$ removes the tail cell from Q.

Stop Condition

We define the stop condition as the condition for which we judge that the goal position has been found. We write this condition as follows:

$$\left\| q_{goal} - q \right\| < \Delta q \tag{9}$$

where q_{goal} is the goal configuration, q is the configuration of the cell verifying the stop condition and Δq is the step of the grid.
If the algorithm can no longer evolve and the stop condition is not satisfied, it means that there is no possible solution for the given density of the grid.

4.2 Algorithm

The algorithm outlined in Fig. 6, starts by constructing the initial cell in step 1. It sets the parent pointer to NULL and evaluates the distance to the goal. The algorithm uses a variable c representing the cell searched by the algorithm. $\aleph$ is the set of explored cells and $\aleph_1$ is the set of unexplored cells in the vicinity of cell c.
Step 6 computes non-collision constraints using distances between obstacles and robot parts evaluated in the posture defined by cell c. Steps 8 to 13 construct unexplored cells in the vicinity of cell c. For each cell the parent pointer

is set to c, the distance to goal is evaluated and the non-collision constraints are checked. A cell is considered a collision if it does not verify constraints given by equation (3).

Step 15 determines the nearest cell to the goal in the vicinity of c, using the distance to goal already evaluated. If that cell is not an obstacle, it is placed in the head of the queue Q at step 17. This makes the planner perform a depth search since there is no obstacle bothering the robot.

However, if the cell computed by step 15 is a collision, all non-collision cells in the vicinity of c that are close to collision cells are placed in the tail of the queue Q by step 22. This makes the planner perform a depth search until the obstacle is bypassed.

---

1. *Construct initial cell* c_1
2. *Set* $c = c_1$
3. *Let* $\aleph = \{c_1\}$
4. *While* $c \neq \emptyset$ *and c does not satisfy the stop condition do*
5. *c.computed=true*
6. *Compute non-collision constraints for the configuration represented by the cell c*
7. $\aleph_1 = Vic(c) \setminus \aleph$
8. *For each cell* $c_2 \in \aleph_1$ *do*
9. *Set c₂.parent = c*
10. *Evaluate c₂.distance_to_goal*
11. *Verify the non-collision constraints and determine c₂.collision*
12. *Set c₂.computed to false*
13. *End for*
14. $\aleph = \aleph \cup \aleph_1$
15. *Choose c₃ in* $\aleph_1$ *with the minimal distance to goal*
16. *If c₃.collision=false then*
17. $h^+(Q, c_3)$
18. *Else (c₃.collision=true)*
19. *For each* $c_2 \in Vic(c)$ *such as c₂.collision=true do*
20. *For each* $c_3 \in Vic(c_2) \cap \aleph$ *set*   *c₃.near_an_obstacle=true*
21. *End for*
22. *For each* $c_2 = Vic(c) \setminus Q$ *such as c₂.Near_an_obstacle = true and c₂.collision=false and* *c₂.computed=false do* $t^+(Q, c_2)$
23. *For each* $c_2 \in Q$ *such as* $Vic(c_2) \subset \aleph$ *remove c₂ from the Queue Q and set* *c₂.computed=true*
24. *End if*
25. $c = h(Q)$
26. $h^-(Q)$
27. *End while*

Figure 6. Pseudo-code of the method
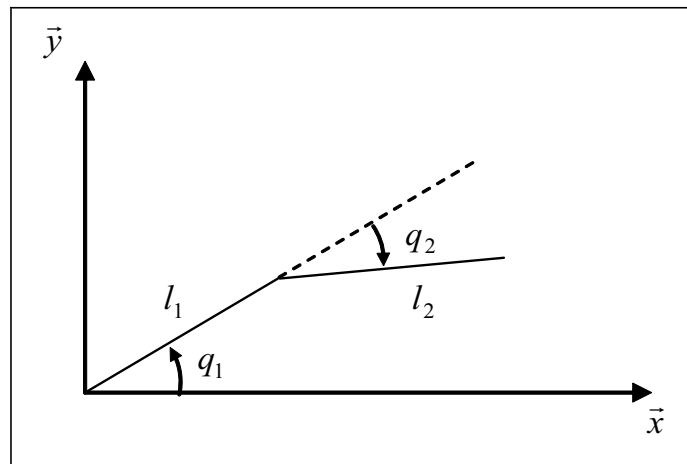
Figure 7. A 2 DoF robot

Steps 19 to 21 evaluate the "near an obstacle" property. This property is set to false when the cell is constructed. Then for each cell in the vicinity of a collision cell, itself in the vicinity of the cell c, this property is set to true.

Step 23 removes from the queue Q all cells for which their vicinity has been already explored and sets their computed property to true, so they do not return to the queue when the algorithm evolves. The search procedure is stopped when a cell verifying the stop condition is found and the path is done by joining this cell to the initial cell by going back through the parent cells using the pointer of each cell. The procedure can also be stopped if the Queue Q is empty, in that case there is no possible path for the chosen resolution of the grid.
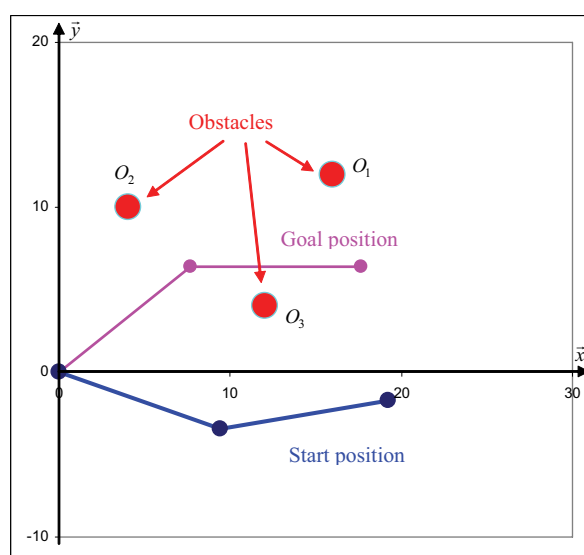
Figure 8. Path planning consists of moving the robot from the start position to the goal position while avoiding obstacles

Obstacle	O_1	O_2	O_3
x	16	4	10
y	12	10	4

Table 1. Position of obstacles

4.3 A planar example

In order to illustrate the proposed algorithm we consider a 2D example, of a 2R robot (Fig. 7) evolving among point obstacles. The simulations are made using three point obstacles defined by table 1.

The start configuration is $q_s = \begin{bmatrix} -20° & 30° \end{bmatrix}^T$ and the goal configuration is $q_g = \begin{bmatrix} 50° & -45° \end{bmatrix}^T$. Fig. 8 shows the robot in its starting and goal positions, respectively, and the three point obstacles. We set the lengths of the arms of the robot $l_1 = l_2 = 10$.

Fig. 9 shows the CSpace of the robot, the dark regions correspond to CSpace obstacles.
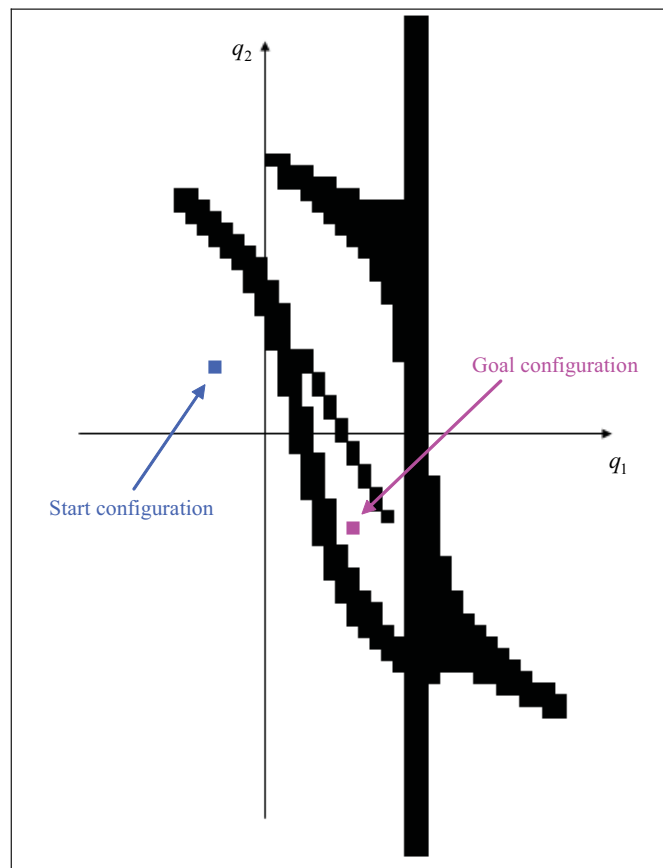
Figure 9. The configuration space
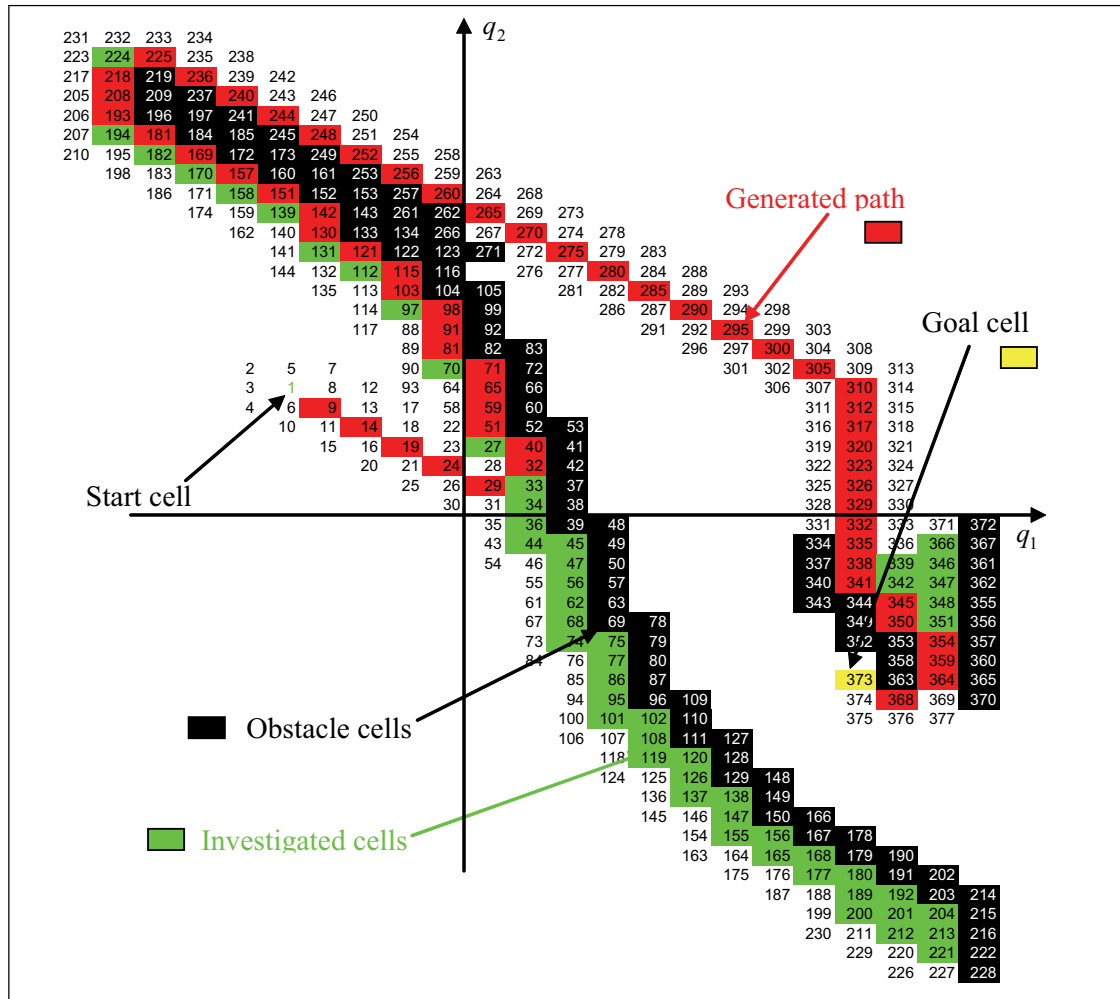
Figure 10. Cell generation order

The construction order of cells is shown in Fig. 10. The algorithm evolves towards its goal using the depth-search mode while there is no obstacle bothering it. When an obstacle is detected the algorithm uses the width-search mode. The algorithm overlaps the obstacle in order to find the best direction to bypass it. When the obstacle is avoided the depth search mode is resumed. The algorithm gives the best way to go around the C obstacle (which is the portion of CSpace corresponding to a collision with one obstacle).
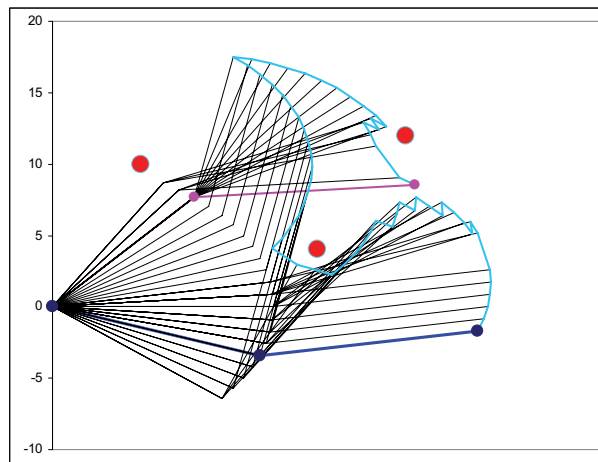
Figure 11. Simulation results for the planar robot
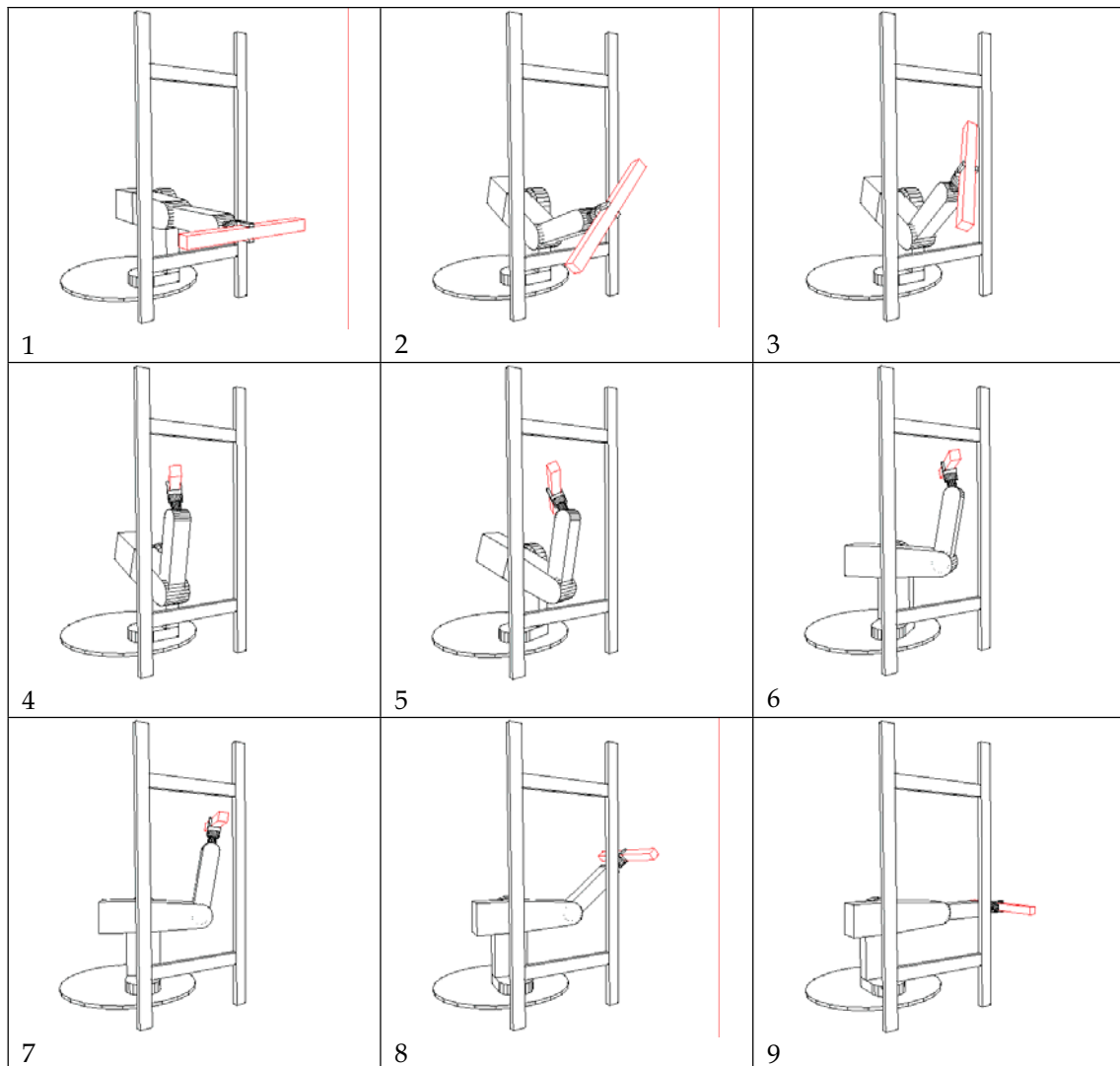
Figure 12. Simulation results for the PUMA robot

The result of the simulation is shown in Fig. 11. Moreover, out of 5329 cells, which corresponds to 73 points on each axis, only 375 cells were computed. This represents less than 10% of the whole workspace.

4.4 Simulation and results

The simulation has been performed on a robotic-oriented-Software named SMAR (Zeghloul et al., 1997). This software is made of two modules: a modeling module and a simulation one. The modeling module is used to generate a model of the robot in its environment. The simulation module is used to simulate the motion of the robot in its environment. It contains a minimal distance feature we used to implement our algorithm.

Fig. 12 shows the simulation results of a 5 DoF ERICC robot carrying a large object and standing in an environment containing ladder-shaped obstacles. The planner determines the path in 20 steps. The robot is carrying a beam whose length is greater than the width of the ladder-shaped obstacle. Regular local path planners would be stuck in the initial position. The proposed method explores all possible configurations capable of going around the obstacle and chooses the one that yields the minimum distance to the goal. The sequence of frames shown in Fig. 12, shows the solution found by the proposed planner. In this case the total number of cells is 12252303 while the number of computed cells is only 220980, which represents less than 2% of the whole workspace.

5. Real-time path planning in dynamic environments

The method described above is useful in the case of cluttered static environments. It can be used offline to generate repetitive tasks. In many cases robots evolve in dynamic environments, which are unknown in advance. That is why we propose to solve the path planning problem for many manipulator robots evolving in a dynamic environment using a real-time local method. This approach is based on the constraints method coupled with a procedure to avoid local minima by bypassing obstacles using a boundary following strategy.

5.1 Local Method

In this method, we use a local planner based on an optimization under constraints process (Faverjon & Touranssoud, 1987). It is an iterative process that minimizes, at each step, the difference between the current configuration of the robot and the goal configuration. When there are no obstacles in the way of the robot, we consider that it evolves towards its goal following a straight line in the CSpace. The displacement of the robot is written as follows:

$$\Delta q_{goal} = \frac{q_{goal} - q}{\left\| q_{goal} - q \right\|} \Delta q_{max} \quad \text{if} \quad \left\| q_{goal} - q \right\| > \Delta q_{max} \tag{10}$$

$$\Delta q_{goal} = q_{goal} - q \qquad\qquad \text{if} \quad \left\| q_{goal} - q \right\| \le \Delta q_{max} \tag{11}$$

where q_{goal} is the goal configuration of the robot, q is the current configuration of the robot and Δq_{max} is the maximum variation of each articulation of the robot. If there are obstacles in the environment, we add constraints (defined in paragraph 3) to the motion of the robot in order to avoid collisions. Path planning becomes a minimization under constraints problem formulated as:

$$\text{Minimize} \left\| \Delta q - \Delta q_{goal} \right\| \quad \text{Under non - collision constraints} \tag{12}$$

where Δq is the change of the robot joints at each step. We can formulate then the planning problem as follows:

$$\text{Minimize} \left\| \Delta q - \Delta q_{goal} \right\| \quad \text{Under linear constraints } \mathbf{n}^T \mathbf{J}\, \Delta q \le \xi \frac{d - d_i}{d_s - d_i} \tag{13}$$

The local planner can be represented by an optimization problem of a nonlinear function of several parameters, subject to a system of linear constraint equations. In order to solve this problem, we use Rosen's gradient projection method described in (Rao, 1984). When the solution of the optimization problem Δq corresponds to the null vector, the robot cannot continue to move using the local method. This situation corresponds to a deadlock. In this case, the boundary following method is applied for the robot to escape the deadlock situation.

In the next section, we define the direction and the subspace used by the boundary following method.

5.2 Boundary following method

Before explaining the method in the general case of an n-DoF robot, we present it for the 2D case. The proposed approach to escape from the deadlock situation is based on an obstacle boundary following strategy.

The 2D case
This method was first used in path planning of mobile robots (Skewis & Lumelsky, 1992; Ramirez & Zeghloul, 2000).
When the local planner gets trapped in a local minimum (see Fig. 13), it becomes unable to drive the robot farther. At this point the boundary following

method takes over and the robot is driven along the boundary of the obstacle until it gets around it. The robot in this case has the choice between two directions on the line tangent to the obstacle boundary or on the line orthogonal to the vector to the goal (Fig. 13). It can go right or left of the obstacle. Since the environment is dynamic and unknown in advance, we have no idea whether going left or going right is better. The choice of the direction is made randomly. Once the obstacle is avoided the robot resumes the local method and goes ahead towards the goal configuration.
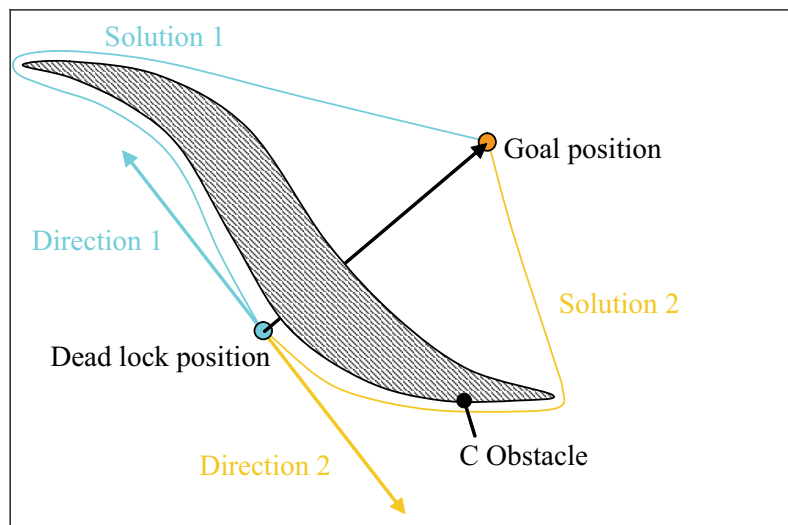
Figure 13. Two possible directions to bypass the obstacle in the case of a 2DoF robot
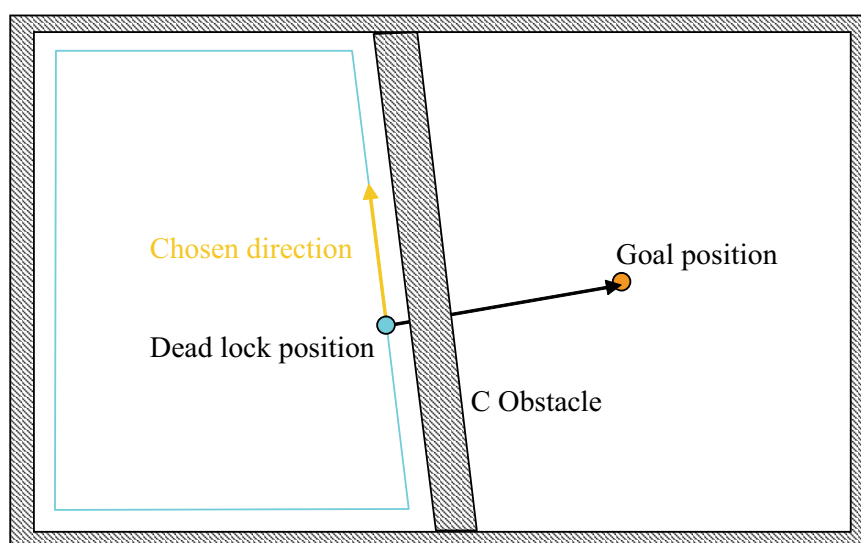
Figure 14. The case where there is no feasible path to the goal

If the boundary following method drives back the robot to the original dead-lock position, one can conclude that there exists no feasible path to reach the goal (Fig. 14) and the process is stopped.

Fig. 15 shows the two feasible paths leading the robot to the goal position. Each path corresponds to one choice of the direction of the motion to follow the boundary of the obstacle. Therefore, and since the environment can be dynamic, the choice of the direction (left or right) is made once and it stays the same until the goal is reached. This unique choice guarantees a feasible path in all situations whenever a deadlock position is found by the local planner (even if in certain cases the choice seems to be non optimal as it is the case for the path 2 using the left direction in Fig. 15).
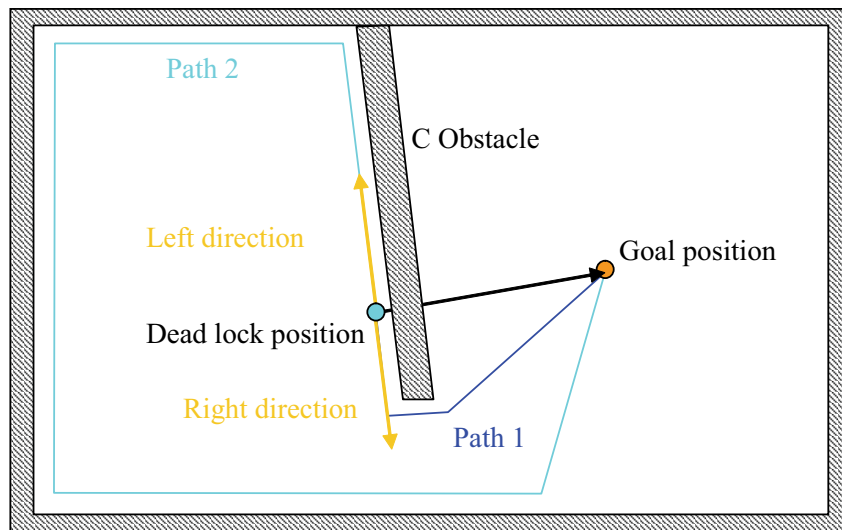
Figure 15. If a solution exists any chosen direction will give a valid path
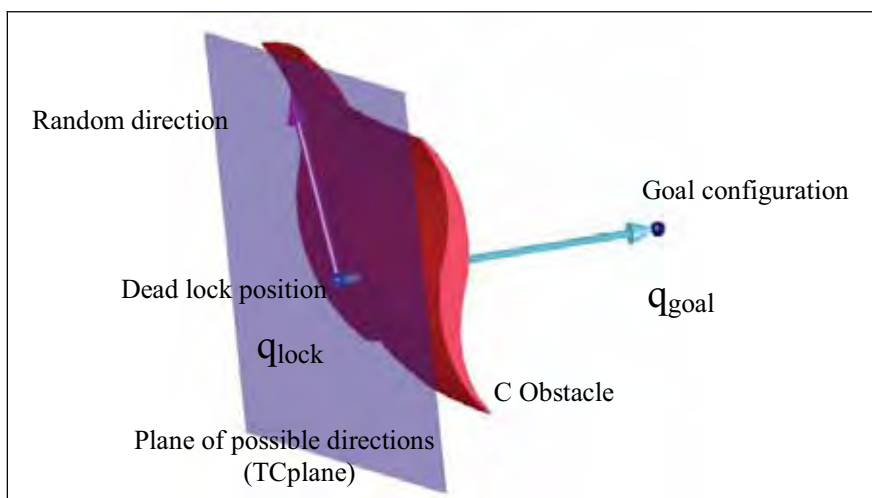
Figure 16. Definition of the TCplane

The n-dimensional case

In the case of a 3-DoF robot, the choice of a direction avoiding the obstacle becomes more critical. Indeed, the directions perpendicular to the vector pointing towards the goal configuration are on a hyperplane of the CSpace, which is in this case, a plane tangent to the obstacle and normal to the vector pointing to the goal position (Fig. 16).

This plane will be called TCplane (Tangent C plane). The path planner can choose any direction among those included in this plane.

As in the case of 2-DoF case, we have no idea about the direction to choose in order to avoid the obstacle. In this case, an earlier method, proposed by Red *et al.* (Red et al., 1987), consists of using the 3D space made of the robots primary DoF. Then, by using a graphical user interface (GUI), the user moves the screen cursor to intermediate interference free points on the screen. A path is then generated between the starting and the final configurations going through the intermediate configurations.
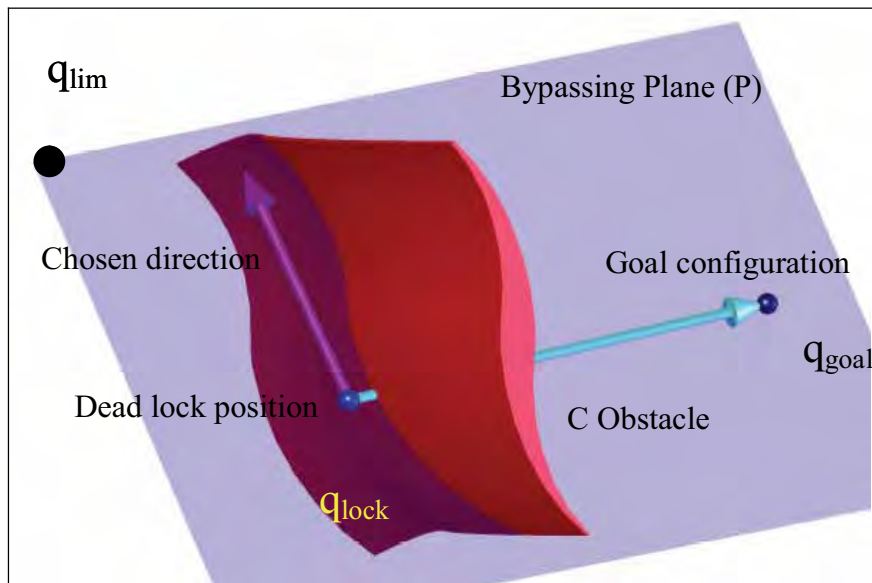
Figure 17. Definition of the Bypassing Plane

This method is applicable only to the primary 3-DoF case when the 3D graphical model can be visualized. Also, the user can choose paths using only the primary DoF, which eliminates other possibilities using the full DoF of the robot. Moreover, this method cannot be applied in real-time applications.

One possible strategy is to make a random choice of the direction to be followed by the robot in the TCplane. This strategy can lead to zigzagged paths and therefore should be avoided. In our case, whenever the robot is in a deadlock position, we make it evolve towards its upper joint limits or lower joint limits, defined by the vector $q_{\lim}$. This strategy allowed us to find a consistent

way to get out of the deadlock position. This chosen direction is defined by the intersection of the TCplane and the bypassing plane (P) containing the three points: $q_{\lim}$, q_{lock} and q_{goal} (Fig. 17).

In the general case of robots with many DoF, the TCplane becomes a hyperplane which is normal to the vector pointing from q_{lock} to q_{goal} and containing q_{lock}. The direction chosen to go around the obstacle is defined by the intersection of the TCplane and the plane (P) defined by the three points : $q_{\lim}$, q_{lock} and q_{goal}. New constraints, reducing the motion of the robot to the plane (P), are defined with respect to non-collision constraints.

The boundary following method will follow these constraints until the obstacle is avoided. This plane (P) will be characterized by two vectors U_1 and U_2, where U_1 is the vector common to all possible subspaces pointing out to the goal configuration. Vector U_1 is given by:

$$U_1 = \frac{q_{goal} - q}{\left\| q_{goal} - q \right\|} \tag{14}$$

U_2 is the vector that defines the direction used by the robot in order to avoid the obstacle. This vector is defined by the intersection of plane (P) and the TCplane. It is not the only possible direction, any random direction can define a bypassing plane that can be used in the boundary following method. The systematic use of $q_{\lim}$ in the definition of U_2 avoids the problem of zigzagged paths. As the robot evolves in a dynamic environment, it has no prior knowledge of obstacles and of their motion and it can not compute the best direction to bypass obstacles. In order to define U_2 we use the vector V given by:

$$V = \frac{q_{\lim} - q}{\left\| q_{\lim} - q \right\|} \tag{15}$$

where $q_{\lim} = q_{\inf}$ if the chosen strategy makes the robot move towards the lower limits of its joints, and $q_{\lim} = q_{\sup}$ if the chosen strategy makes the robot move towards the upper limits of its joints. U_2 is the unit vector orthogonal to U_1 and located in the plane (U_1, V). Vector U_2 is given by:

$$U_2 = \frac{V - \left(U_1^T V\right) U_1}{\left[\left(V - \left(U_1^T V\right) U_1\right)^T \left(V - \left(U_1^T V\right) U_1\right) \right]^{\frac{1}{2}}} \tag{16}$$

While avoiding the obstacle, the robot will move in the defined subspace (P), and Δq could be written as

$$\Delta q = \Delta u_1 \, U_1 + \Delta u_2 \, U_2 \qquad (17)$$

Where, Δu_1 is the motion along the U_1 direction and Δu_2 is the motion along the U_2 direction.

Whenever an object is detected by the robot, which means that the distance between the robot and the obstacle is less then the influence distance, a constraint is created according to equation (6). Constraints are numbered such that the ith constaint is written as:

$$\begin{bmatrix} a_{i1} & \cdots & a_{iN} \end{bmatrix} \begin{bmatrix} \Delta q_1 & \cdots & \Delta q_N \end{bmatrix}^T \leq b_i \qquad (18)$$

If we replace Δq by its value in the subspace, we get

$$\begin{bmatrix} a_{i1} & \cdots & a_{iN} \end{bmatrix} \left(\Delta u_1 \, U_1 + \Delta u_2 \, U_2 \right) \leq b_i \qquad (19)$$

Let

$$au_{i1} = \begin{bmatrix} a_{i1} & \cdots & a_{iN} \end{bmatrix} U_1 \qquad (20)$$

$$au_{i2} = \begin{bmatrix} a_{i1} & \cdots & a_{iN} \end{bmatrix} U_2 \qquad (21)$$

The projected constraints on the bypassing plane are written as

$$A_i^T \, \Delta u \leq b_i \qquad (22)$$

with

$$A_i = \begin{bmatrix} au_{i1} & au_{i2} \end{bmatrix}^T \qquad (23)$$

$$\Delta u = \begin{bmatrix} \Delta u_1 & \Delta u_2 \end{bmatrix}^T \qquad (24)$$
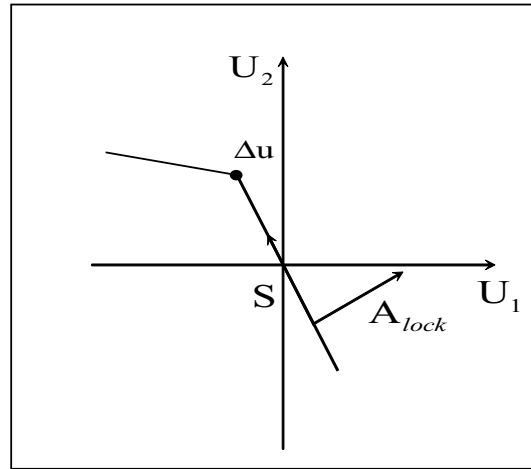
Figure. 18. Boundary following result
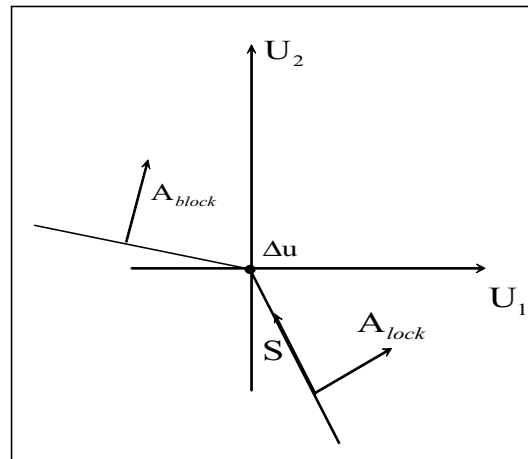
Figure 19. Constraint switching

In order to escape from deadlocks, we follow the projected constraints corresponding to the obstacles blocking the robot. To do so, we use the Boundary following method described in the next section.

The boundary following Algorithm
This method uses the distance function defined as:

$$\mathbf{V}(q) = \left\| q - q_{goal} \right\| \tag{25}$$

which is the distance from the current position of the robot to the goal position. The value of the distance function is strictly decreasing when the robot is evolving towards its goal using the local planner. When a deadlock is detected, we define $d_{lock} = \left\| q_{lock} - q_{goal} \right\|$ as the distance function in the deadlock configura-

tion. While the robot is going around the obstacles using the boundary following method, the distance function, $V(q)$, is continuously computed and compared to d_{lock}. When the value of the distance function is lower than d_{lock}, the robot has found a point over the C obstacle boundary that is closer to the goal than the deadlock point. At this moment, the robot quits the boundary following method and continues to move towards the goal using the local planner. The vector of the followed constraint is named A_{lock}. It corresponds to the vector of the projected constraint blocking the robot. The boundary following method can be stated as follows:

1. Initiate the parameters A_{lock} and d_{lock}
2. Evaluate the distance function. If it is less than d_{lock} then quit the boundary following method and resume the local planner
3. Find and update the followed constraint A_{lock}
4. Find the vertex enabling the robot to go around the obstacle
5. Move the robot and go to step 2
6. Fig. 18 shows the followed vertex Δu. It is the point on the constraint A_{lock} in the direction of S and it satisfies all the projected constraints.

$$S = \begin{bmatrix} -au_{lock2} & au_{lock1} \end{bmatrix} \qquad (26)$$

where

$$A_{lock} = \begin{bmatrix} au_{lock1} & au_{lock2} \end{bmatrix}^T \qquad (27)$$

At each step the algorithm tracks the evolution of the followed constraint among the set of the projected constraints. The tracked constraint is the one maximizing the dot product with A_{lock}. In certain cases the resultant vertex Δu is null when there is another projected constraint blocking the robot (Fig. 19). This is the case of point B in Fig. 20. In this case, the robot switches the followed constraint. It uses the blocking constraint to escape from the deadlock. Fig. 20 shows the case of a point robot moving from point S to point q_{goal}. The robot moves from point S to point q_{lock1} using the local planner.
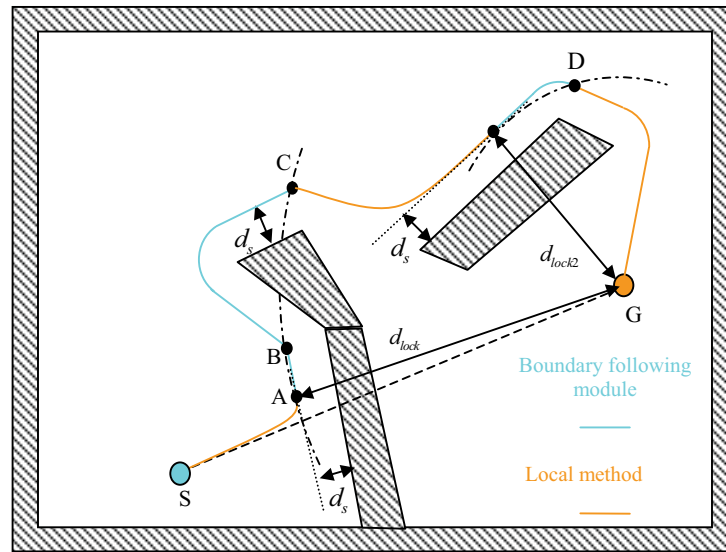
Figure 20. An illustrating example

The point q_{lock1} corresponds to a deadlock position where the robot can no longer move towards the obstacle while respecting the security distance, d_s, from the obstacle. This point corresponds also to a local minimum of the distance function, $\mathbf{V}(q) = d_{lock1}$. At this point, the robot starts to follow the boundary of the blocking obstacle and the distance function $\mathbf{V}(q)$ is continuously compared to d_{lock1}. In point B there is another obstacle preventing the robot from following the first one. In that case, the boundary following module changes the path of the robot to follow the new obstacle. In point C the distance to the goal has decreased and becomes equal to d_{lock1}, which means that the robot bypassed the obstacle and the local planner is resumed. When reaching point q_{lock2}, a second deadlock position occurs. Therefore, the boundary following module is activated again until point D is reached, which corresponds to a distance from the goal equal to d_{lock2}. At this point the local method is resumed to drive the robot to its goal position.
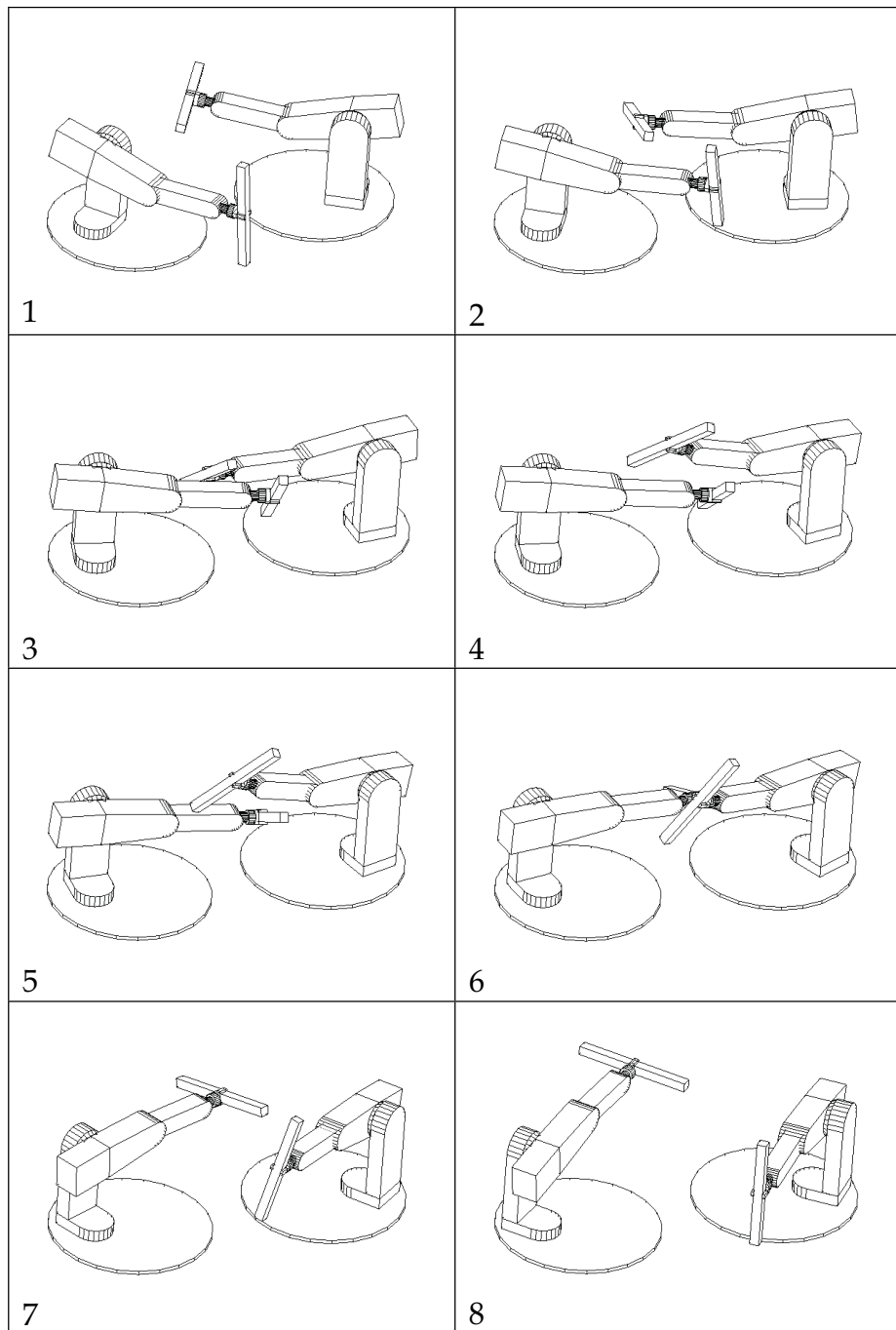
Figure 21. Results using two 5-DoF robots

5.3 Simulation and results

In order to evaluate the efficiency of the method, we present several examples. All the simulations have been performed on SMAR (Zeghloul et al., 1997). This method was added to the simulation module. All the following examples were simulated on a Pentium IV. Path planning was performed in real time and did not slow down the motion of the robot compared to the case without obstacles. The first example is made of two 5-DoF robots, where each one is taking an object from an initial position to a final one (Fig. 21). The two robots come closer to each other and they have to avoid collision.

Frames 4, 5 and 6 show the two robots following the boundary of each other by keeping a security distance. This task would not be possible if we used only the local planner, because it would be stuck as soon as two faces of the two objects become parallel, which happens in Frame 3.

Fig. 22 shows the results using three PUMA robots. Each one of the three robots considers the two other robots as moving obstacles. Each robot moves towards its goal, once a deadlock position is detected, the robot launches the boundary following method. Until Frame 3 the local planner is active for the three robots. As soon as the robots get close to each other the boundary following module becomes active (Frame 4).

When each robot finds a clear way to the goal the local planner takes over (Frame 13) to drive each robot to its final position.

In these simulations, robots anticipate the blocking positions. If the value of the joint velocity given by the local method is less than 30% of the maximum joint velocity, the robot starts the boundary following method. Elsewhere, the boundary following method is stopped and local method is resumed when the distance function is less then 0.8 d_{lock}. These values are found by performing some preliminary simulations. Anticipating the deadlock position makes the resultant trajectories smoother, as the robot does not wait to be stopped by the deadlock position in order to begin the boundary following method.
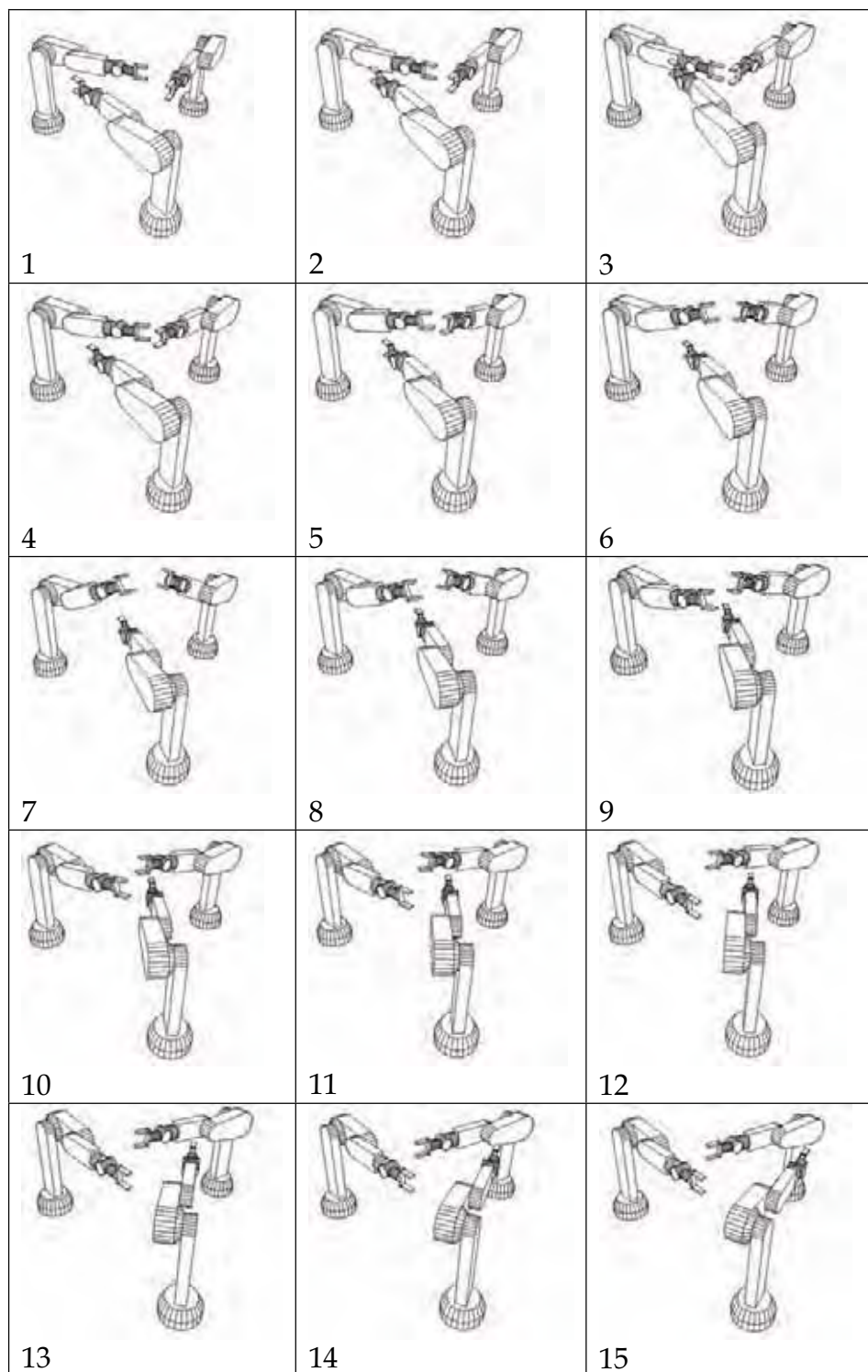
Figure 22. Results using three PUMA robots

6. Conclusion

In this paper, we presented two methods of free path planning. The first one is based on lazy grid methods. It searches for a path without using a heuristic function. This method reduces the gap between classic grid methods where all the grid cells must be computed before searching for a path, and lazy grid methods where the grid is computed while searching for the path. The proposed planner is very general and is guaranteed to find a path, if one exists, at a given resolution. However, this algorithm depends on the resolution of the grid. The higher this resolution is, the closer the robot can squeeze between obstacles. This method reduces the number of computed cells and gives the best direction to go around a C obstacle. It can be combined with quasi-random methods and it replaces the A* searching module, where quasi-random sampling of the CSpace appears to offer performance improvements in path planning, see for instance (Branicky et al., 2001).

The second part of this work was concerned with a novel method for path planning suitable for dynamic environments and multi-DoF robots. This method is a combination of the classical local method and the boundary following method needed to get the robot out from deadlock positions in which the local method gets trapped. The local path planner is based on non-collision constraints, which consists of an optimization process under linear non-collision constraints. When a deadlock, corresponding to a local minimum for the local method, is detected, a boundary following method is launched. A similar method can be found for the 2D cases, and we show in this work how it can be applied to the case of multi-DoF robots. When the robot is stuck in a deadlock position, we define the direction of motion of the robot, in the configuration space, as the intersection of a hyperplane, called TCplane, a plane defined by the vector to its goal and a vector to its joint limits. This direction of motion allows the robot to avoid the obstacle by following its boundary until it finds a path to the goal, which does not interfere with the obstacle. Starting from this point the classical local planner takes over to drive the robot to its goal position. This method is fast and easy to implement, it is also suitable for several cooperating robots evolving in dynamic environments.

7. References

Bohlin, R. & Kavraki, L. (2000). Path planning using lazy prm, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 521–528, San Francisco, CA, April 2000.

Branicky, M.; LaValle, S.; Olson, K. & Yang, L. (2001). Quasi-randomized path planning, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1481-1487, Seoul, Korea, May 2001.

Faverjon, B. & Touranssoud, P. (1987). A local based approach for path planning of manipulators with a high number of degrees of freedom, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1152–1159, Raleigh, March 1987.

Helguera, C. & Zeghloul, S. (2000). A local-based method for manipulators path planning in heavy cluttered environments, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3467–3472, San Francisco, CA, April 2000.

Kavraki, L.; Svestka, P.; Latombe, J-C. & Overmars M. (1996). Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, (august 1996), pp. 566–580, ISSN 1042-296X.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, Vol. 5, No. 1, (1986), pp. 90–99.

Kondo, K. (1991). Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 267–277, Sacramento, CA, April 1991.

Lahouar, S.; Zeghloul, S.; Romdhane, L. (2005a). Path planning for manipulator robots in cluttered environments. *WSEAS Transactions On Systems*, Vol. 4, No. 5, (May 2005) pp. 555-560. ISSN 1109-2777.

Lahouar, S.; Zeghloul, S.; Romdhane, L. (2005b). Path planning for manipulator robots in cluttered environments. *International Design Engineering Technical Conferences & Computers and Information In Engineering Conference*, DETC2005-84993. Long Beach, CA, September 2005.

Lahouar, S.; Zeghloul, S.; Romdhane, L. (2006). Real-time path planning for multi-DoF robot manipulators in dynamic environment. *International Journal of Advanced Robotic Systems*, Vol. 3, No. 2, (June 2006) pp. 125-132. ISSN 1729-8806.

LaValle, S. (2006). *Planning Algorithms*, Cambridge University Press, ISBN 0521862051.

Lengyel, J.; Reichert, M.; Donald, B. & Greenberg D. (1990). Real-time robot motion planning using rasterizing computer graphics hardware. *Computer Graphics*, Vol. 24, No. 4, (August 1990), pp. 327–335.

Lozano-Pérez, T. & Wesley, M. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, Vol. 22 , No. 10, (October 1979), pp. 224–238, ISSN 0001-0782.

Mediavilla, M.; Gonzalez, J.; Fraile, J. & Peran, J. (2002). Reactive path planning for robotic arms with many degrees of freedom in dynamic environments, *Proceedings of 15 th Triennial Word Congress*, Barcelona, Spain, 2002.

Paden, B.; Mess, A. & Fisher, M. (1989). Path planning using a jacobian-based free space generation algorithm. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1732–1737, Scottsdale, Arizona, 1989.

Ramirez, G. & Zeghloul, S. (2000). A new local path planner for nonholonomic mobile robot navigation in cluttered environments, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2058–2063, San Francisco, CA, April 2000.

Rao, S.S. (1984). *Optimization theory and applications*, Wiley, ISBN 0470274832, New York.

Red, W.; Troung-Cao, H. & Kim, K. (1987). Robot path planning in three-dimensions using the direct subspace. ASME Journal of Dynamics, Measurement and Control, Vol. 109, pp.238–244.

Siméon, T.; Laumond, J-P. & Nissoux, C. (2000). Visibility-based probalistic roadmaps for motion planning. *Advanced Robotics Journal*, Volume 14, No. 6, pp. 477--494, 2000, ISSN 0169-1864.

Skewis, T. & Lumelsky, V. (1992). Experiments with a mobile robot operating in a cluttered unknown environment, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1482–1487, Nice, France, May 1992.

Tournassoud, P. (1992). *Planification et contrôle en robotique: Application aux robots mobiles et manipulateurs*, Hermes, ISBN 2866013220, Paris.

Udupa, S. (1977). *Collision Detection and Avoidance in Computer Controlled Manipulators*, PhD thesis, Dept. of Electrical Engineering, California Institute of Technology, 1977.

Wilmarth, S.; Amato, N. & Stiller, P. (1999). Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1024–1031, Leuven, Belgium, May 1999

Yang, S. (2003). Biologically inspired robot behavior engineering, *Biologically inspired neural network approaches to real-time collision free robot motion planning*, pp. 143–172, Springer Verlag, ISBN 3790815136.

Zeghloul, S.; Blanchard, B. & Ayrault, M. (1997). Smar: A robot modeling and simulation system. *Robotica*, Vol. 15, No. 1, (january 1997), pp. 63–73, ISSN 0263-5747.

Determination of Location and Path Planning Algorithms for Industrial Robots

Yung Ting and Ho-Chin Jar

1. Introduction

Before path planning, it is significant to determine the robot location, which very few researches have addressed in this topic. Determination of a suitable robot location is influential to prepare for the subsequent path search with better solution or even to ensure the possibility of finding a path. In particular, the environment with complex obstacles, the inspection is demanding. In this article, a method by use of the intersection theorem (Danielle & Mark, 2001) is proposed to determine the robot location.

Path planning has been studied with numerous researches on the topics of minimum time, minimum energy, and obstacle avoidance, etc. Obstacle avoidance is probably the most distinguished one investigated for many application purposes. Distance maps is one of the earlier method to divide the space by grids with equal distance. The obstacle is mapped in this 2D diagram. The rest of the area is considered to be passable and marked with integer numbers, which indicates the distance to the obstacle (Latombe, 1991; Pobil et al., 1992; Jarvis, 1993). Wave expansion method is derived based on the distance maps. It starts to mark the passable nodes with sequential integer numbers from the selected initial position to expand outward, and then begins the path search at the final position (Barraquand et al., 1992; Ralli & Hirzinger, 1994).

Configuration space concept is proposed by (Lozano-Perez, 1987; Banski, 1996; Red & Truong-Cao, 1996). It attempts to illustrate the robot manipulation geometry in terms of the joint space. For an n degree-of-freedom robot, there is n dimensional vector in the configuration space, where the collision occurs in the workspace can be expressed.

In this study, three path-planning methods, the neighboring search method, the depth-first search method, and the extensile search method, are developed. The path searching capability, manipulation steps and time are discussed with comparison.

A practical automobile baggage trunk welding process in association with an industrial robot, ABB IRB1400, is selected as an example to be investigated with simulation on the Robot Studio S4-lite software. The proposed extensile

neighboring search method, in particular, is more reliable to find a path and shows autonomous capability of reducing manipulation steps.

2. Determination of Robot Location

Inappropriate location of the robot may cause inconvenient operation, or even unexpected damage. Especially, it may provide no solution for path planning when dealing with complex obstacles in the working environment. Therefore, investigating the feasible location area of the robot in the Cartesian coordinate system before path planning is the primary task.

The shapes of miscellaneous obstacles are difficult to express by simple mathematical description. An easy way is to segment the obstacle into analyzable geometric shapes such as triangle or rectangle. The unavoidable error due to this approximation approach is tolerable because it does not affect the determination of robot location and the following path planning obviously. For example, the top view of an automobile baggage trunk in 2D is shown in Figure 1. The solid line represents the boundary of the baggage trunk. The segmented rectangular areas bounded by the dashed lines replace the original practical trunk shapes. For instance, the robot needs to pass the four (A,B,C,D) working points. The possible location area to cover each of the passing points (A,B,C,D) is represented R_A, R_B, R_C, and R_D, respectively. Via inspection on the intersection with the obstruction area of the obstacle, the possible location area is obtained and can be mathematically described by

$$R = (R_A \cap \overline{O}) \cap (R_B \cap \overline{O}) \tag{1}$$

where "O" represents the obstruction area of one of the segmented rectangular obstacles, and "R" represents the inspected possible region that the robot can be located for the working points A and B. To check each rectangular shape in sequence, the possible location areas are searched, so that the robot location is determined.

Similarly, the passable area for considering all of the segmented obstacles (O_1, O_2, …, O_n) etc., are defined as

$$R = [(R_A \cap \overline{O_1}) \cap (R_B \cap \overline{O_1})] \cap [(R_A \cap \overline{O_2}) \cap$$
$$(R_B \cap \overline{O_2})] \cap \cdots \cap [(R_A \cap \overline{O_n}) \cap (R_B \cap \overline{O_n})] \tag{2}$$

Concerning all of the desired working passing points (A,B,C,D, …), the possible location region R is inspected with the same process in (2). In case that the intersection area is none, that is, $R =$[ ], then, there may not have suitable robot

location for the subsequent path planning. On the other hand, large space of R may provide better solution for searching a suitable path.

As shown in Figure 1, each of the fours bigger circles represents the possible location area while the robot stretches out to pass each of the points A,B,C,D, respectively. Similarly, the other four smaller circles represent the possible location area while the robot withdraws to pass the points A,B,C,D, respectively. Similar tests are also carried on in the other Y-Z and Z-X planes, which are not addressed in detail here. It is concluded that the shaded area is the possible region for robot location. Via numerical analysis, the point E in Figure 1 is selected as the robot location.
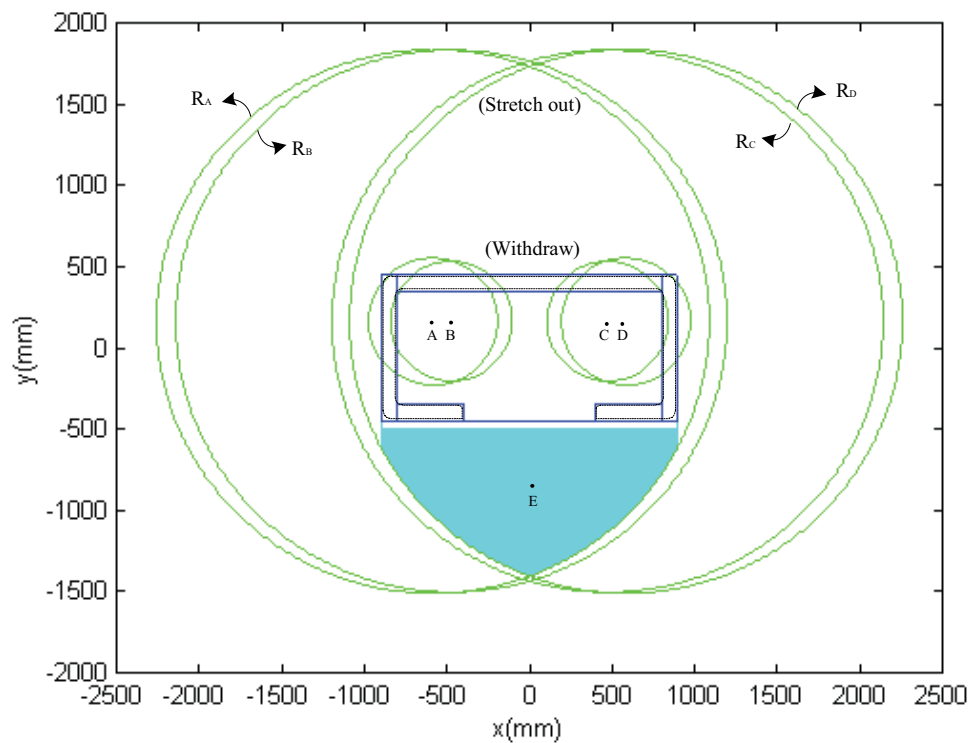
Figure 1. Segmented Obstacles and Robot Location

3. Collision Inspection

The robot and the obstacle need to be defined in the geometric space. Since the last three joints inherited with limited manipulation space for most of the six DOF industrial robots, the robot structure can be lumped to express by the first three arms of the elbow (Ting et al., 2002).

The robot and the obstacle can be described in a discretized space by the distance maps method. In this 2D map, the perpendicular grids intersect on the nodes and cut the space into numerous equivalent squares (Pobil et al., 1992).

Configuration space method is a tool to transfer the robot manipulation geometry into the joint space so that robot collision inspection can be achieved in the same space. The configuration space is established by the joint variables, which is tantamount to the dimension of degree-of-freedom of the robot. Thus, it is convenient to transform the robot and the obstacle structure in the distance maps into the configuration space for further investigation.

According to the robot shape, the boundary function Plane (P) is used to check an arbitrary point P whether the collision appears (Ting et al., 2002). Via collision inspection, the nodes of the obstacle and the unreachable region of the robot in the configuration space are marked –1, and those of the movable range are marked 0.

4. Path Planning

Wave expansion method provides an appropriate approach for path planning (Pobil et al., 1992; Ting et al., 2002). Via the previous collision inspection results, the passable area marked 0 can be expanded outward either from the initial position or the final position and marked with a specified integer number in the configuration space. The number is marked with 1 at the chosen start node, and gradually increased to n at the end node (Ting et al., 2002). For example, the passable nodes are marked with numbers shown in Figure 2.
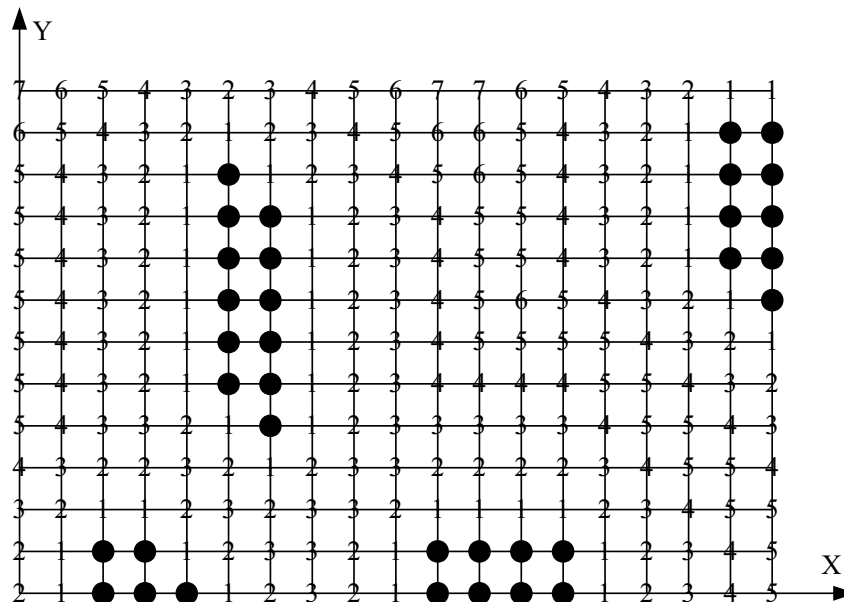
Figure 1. Marked numbers on the nodes

4.1 Neighboring Search Method

While searching for suitable path, it is suggested to start from the node of final position with the largest marked number since there is no guarantee to have a solution to start on the initial position. All of the passable nodes with one unit distance around the start node, 8 nodes at most for two dimension and 26 nodes at most for three dimensions, are checked to see whether there exists at least one passable neighboring node. If the marked number of the start node is n, the node around it with the smallest number, for example, n-1 or n-2, is the desired node as the next passable destination. Then, from that node, the subsequent search is continued until the marked number of the desired passable node is 1. To conjoin these searched nodes, thus determines the path. In case that the chosen passable node does not eventually search a path successfully, it needs to switch to another node with smaller number, and then continues the path search from there again.

4.2 Depth-first Search Method

The depth-first method is derived based upon the data structure concept of the computer system. It starts at the node of the final position with the largest marked integer number n, and searched the neighboring node whose marked number (n–1) must be smaller than it by one. For instance, the searched path in the tree structure is illustrated in Figure 3. The white nodes are the likely nodes to pass through, and the black nodes indicate the obstructive area or the robot unreachable area. The start node means the initial position to begin wave expansion, and the end node means the final position to begin the marked number. The Null indicates the termination of search with no solution.
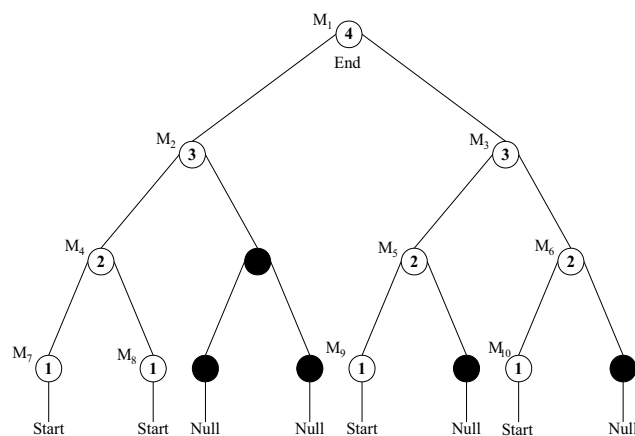
Figure 3. Depth-first search method

As the depth-first search method is used in data structure, a path is obtained from the top to the bottom and from left to right on the tree structure (Tarjan, 1972; Ting & Lei, 1999). In this example, the path searching process is followed by

$M_1 \rightarrow M_2 \rightarrow M_4 \rightarrow M_7 \rightarrow M_8 \rightarrow M_3 \rightarrow M_5 \rightarrow M_9 \rightarrow M_6 \rightarrow M_{10}$

Hence, several likely paths are concluded as below.

(1) $M_1 \rightarrow M_2 \rightarrow M_4 \rightarrow M_7$ (2) $M_1 \rightarrow M_2 \rightarrow M_4 \rightarrow M_8$

(3) $M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_9$ (4) $M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_{10}$

To reduce the robot manipulation steps, an easy way is to merge the nodes in the same manipulation direction (Lei, 1999).

4.3 Extensile Neighboring Search Method

Via many experimental testing, the neighboring search method does not obtain a good path in comparison with the depth-first search method. It is interesting to dig out why the former one searching 26 neighboring nodes along various directions cannot obtain better outcome than the latter one searching only 6 neighboring nodes. It is attempted to develop an extensile neighboring search method that outperforms the depth-first search method by solving the defects of the neighboring search method as described below.

While using the wave expansion method, the start and the end nodes are selected to be either the initial or the final positions. Once the initial position is chosen, the wave expansion begins at that nodal point. An investigation is carried out to examine whether there is different wave expansion solution by exchanging the initial and the final destinations. As illustrated in Figure 4, two paths, the solid and the dotted lines, are obtained by interchanging the initial with the final destinations. The bold line represents the passable region of both cases. It is obviously to see that the searched two paths are not the same.
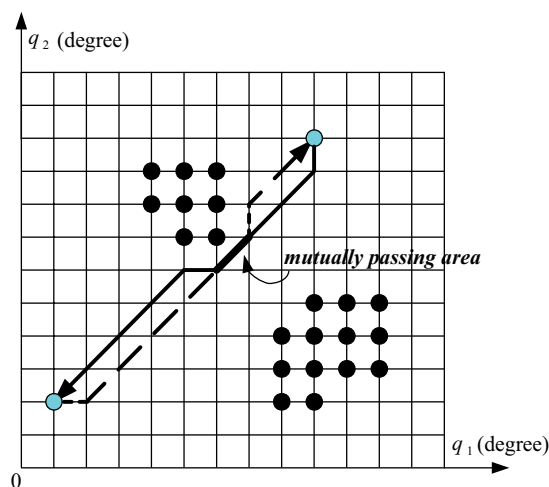
Figure 4. Two searched paths by exchanging the initial with the final positions

Therefore, the selection of the start node for wave expansion and the end point for backward numbering is important, and may seriously affect the search result. This situation is more obvious in the 3D environment for the search neighboring nodes are increased from 8 (2D) to 26 directions. To double-check on the searched path by interchanging the initial with the final position is necessary.

In Figure 4, a solid line is searched on condition that the furthest left node is selected as the initial position to start wave expansion process, and the furthest right node is the final position to begin the path search. On contrary, the initial and the final positions are exchanged to obtain the dotted path. It is seen that the difference of the searched paths between the two cases appears two parallelogram areas. In these areas, it is for sure that no collision occurs. Thus, it is likely to merge the two searched paths into one as shown in Figure 5.
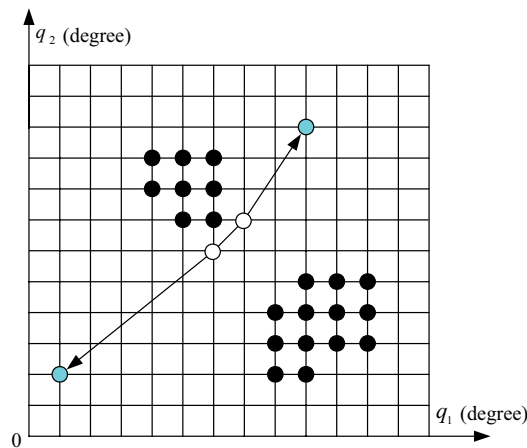
Figure 6. Merge of the two paths into one solution

While using the neighboring search method, the robot path must pass the grid nodes of the same square so that a saw tooth type of path is likely obtained. Thus, it limits the searched path no matter how small the grid is planned. This defect may result in the searched path is not flexible and the manipulation time is elongated because of passing more nodes. This phenomenon is illustrated in Figure 6.
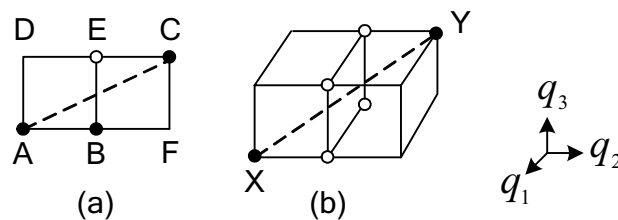
Figure 6. Reduction of path in 2D and 3D

On the assumption that the path $A \rightarrow B \rightarrow C$ shown in Figure 6(a) is the original searched path, it can be reduced to $A \rightarrow C$. It is not solvable by the neighboring search method since the searched path has to move along the grid nodes in the range of a unit square. The nodes D and E on the unit square ABED are considered to inspect collision (Ting et al., 2002). Since the triangle $\triangle ABE$ includes the dotted line across the parallelogram, it is necessary to check only the node E. To extend to the 3D environment shown in Figure 6(b), the path $X \rightarrow Y$ is desired, and it only needs to check the area is enveloped in the grid nodes includes the dotted line. This method is useful to reduce the path. For example, once the circled grid nodes shown in Figure 7 are ensured no collision occurs, then the path is defined to be $P \rightarrow Q$, which thus saves many manipulation steps.
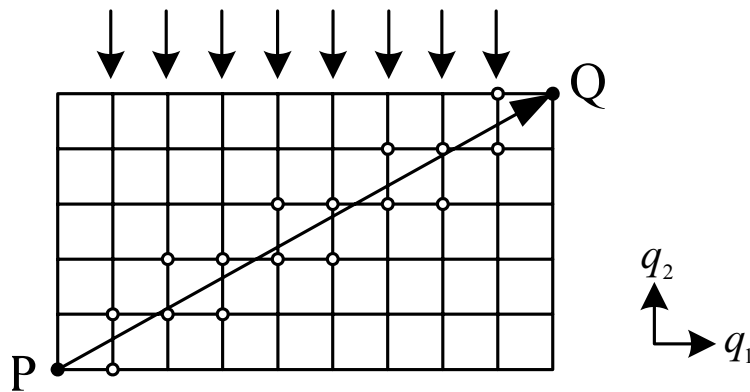
Figure 7. Further reduction of searched path by checking the circled nodes

It is quite difficult to intuitively choose the grid nodes for checking in the 3D space. A recursive program is developed to systematically inspect the collision of the nodes around the unit square where the considered path intersects. The algorithm is described as below.

Recursive Algorithm:
Assuming the kth grid node is located at (a,b,c), and the $(k+2)^{th}$ grid point is at (x,y,z). Provided that the distance along the X-Y-Z direction is defined with $|x-a| \geq |y-b| \geq |z-c|$, and the grid unit distance is designated with d degree. Let the grid number N is defined as $N = |x-a|/d$, and the checked nodes (p,q,r) is defined as $(p,q,r)=(x-a,y-b,z-c)/N$, and the plus or minus symbol of the checked nodes is defined by $(u,v,w)=sgn(p,q,r)$, respectively. By use of the above definitions, three conditions are defined as below.

1. While $|p|=|q|=|r|$, then inspects the nodes at (a+du, b+dv, c+dw),…, (a+(N-1)du, b+(N-1)dv, c+(N-1)dw).

2. While $|p|=|q| \neq |r|$, then inspects nodes at (a+du, b+dv, c), (a+du, b+dv, c+dw),…, (a+(N-1)du, b+(N-1)dv, c+(N-2)dw), (a+(N-1)du, b+(N-1)dv, c+(N-1)dw).

3. While $|p| \neq |q| \neq |r|$, then inspects (a+du, b, c), (a+du, b+dv, c), (a+du, b, c+dw), (a+du, b+dv, c+dw),…, (a+(N-1)du, b+(N-2)dv, c+(N-2)dw), (a+(N-1)du, b+(N-1)dv, c+(N-2)dw), (a+(N-1)du, b+(N-2)dv, c+(N-1)dw), (a+(N-1)du, b+(N-1)dv, c+(N-1)dw).

After the inspection procedures described above, the original path k→k+1→k+2 can be simplified to be k→k+2 on condition that there is no collision occurs at the checked nodes around the unit square. Thus, if the path is searched to pass n nodes, it may be reduced at most (n-1) nodes.

The extensile neighboring search method is able to obtain a path near the obstacle. It is advantageous to deal with a complicated environment such as complex obstacles or limited passable area. Also, unlike the neighboring and the depth-first search methods need intuitive decision to reduce the manipulation steps along the same moving direction, it is autonomous to complete path search by the developed recursive algorithm. Moreover, there may be a situation that the depth-first search method is infeasible. This method inspects the up, down, left, and right directions in 2D. For instance, the obstacle is transferred into the configuration space as depicted in Figure 8.
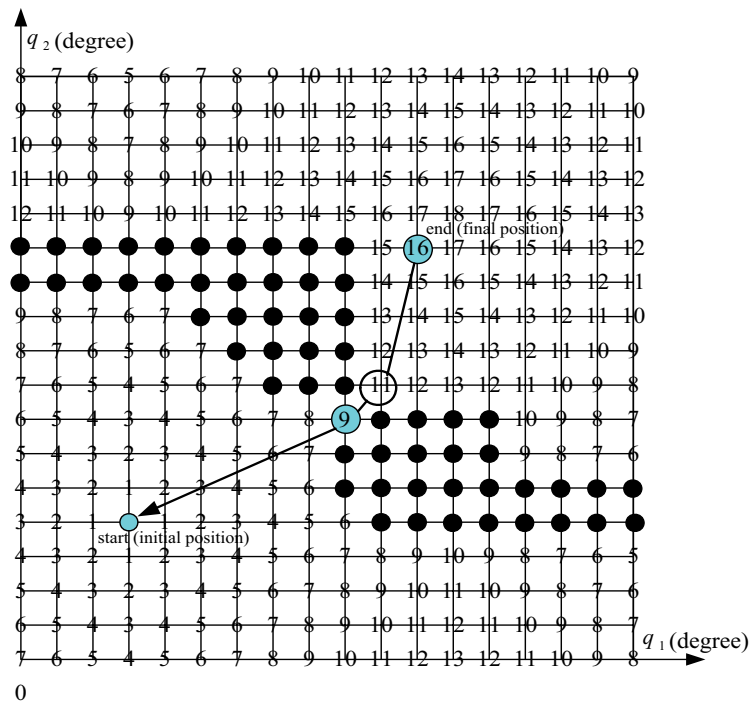
Figure 8. Unsuccessful path planning by depth-first search

It is seen that there is no subsequent node to continue with when the path search meets the grid number 11. That is, the depth-first search method is not workable if the grid node of number (n-1), number 10 in this example, does not neighbor to the current searched node. On the other hand, the extensile neighboring search method checks the surrounding nodes with number (n-2), number 9 in this example, so that it is able to find a path.

5. Manipulation Time

As a matter of fact, the manipulation time is more concerned than the manipulation steps. In general, any type of motion profile can be planned by motion programming method (Tesar & Matthew, 1976). In this study, for example, according to the driving motor of ABB IRB-1400 industrial robot (ABB, 1996), a polynomial form to express the motion profile is given by

$$S(t)=C_5t^5+ C_4t^4+ C_3t^3+ C_2t^2+ C_1t^1+ C_0 \tag{3}$$

Two cases of motion profiles are presented in Figure 9. In Figure 9(a), the initial time is at t_0. The time between t_0 and t_1 is the time needs to arrive at the maximum speed. The time between t_1 and t_2 is the time to manipulate with the maximum speed. The time between t_2 and t_3 is the time needs to reduce the speed to zero. In case the motor of the robot does not reach its maximum speed due to too short distance of the manipulation step, the motion profile is illustrated in Figure 9(b). Once the coefficients in (3) are defined, the manipulation time following the designed motion profile is then computed for both cases. It is noted that the distance of each piece of straight line in the joint space formed by the passable nodes may be different. For example, some passable nodes are connected to form a straight line. That is, more manipulation steps are accompanied with. Therefore, the motion profile planned for each piece of straight line is different. The entire manipulation time is the sum of the manipulation time of each piece of the passing straight line.
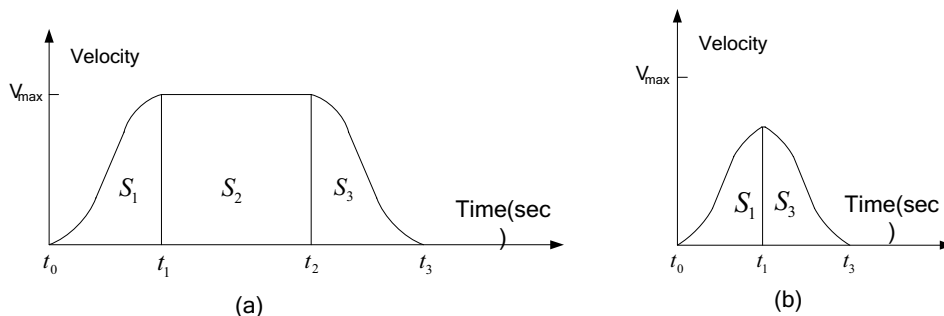
Figure 9. Motion profile for different situation

6. Simulation and Results

An ABB IRB-1400 industrial robot with 6 DOF is selected as an example to investigate the path planning. The maximum motor speed of the robot is limited to be 30 cm/sec. The shape of the first three links is assumed to be cylinder with radius r_1, r_2, r_3 and corresponding link length l_1, l_2, l_3. The last three joints of the wrist of most the industrial robots have limited manipulation range. For example, the workspace of the fifth joint of ABB IRB-1400 industrial robot is negligible because of short link length and small joint working area, and ineffective to the rest two joints. The radius r_3 of the third link is intended to expand to include the workspace of the fifth link. Hence, the simplified first three joints with joint variables (q_1, q_2, q_3) of the elbow are considered for the path planning (Ting et al., 2002).

A practical automobile baggage trunk is depicted with simplified picture shown in Figure 10. The dimension of this obstacle is about 900mm×1800mm× 500mm. The designated passing points are assumed to be A, B, C, D, E, F, where points A and F are the start and the end positions, respectively, and the rest of them are the welding positions. These points mapped into the configuration space in terms of the three joints are (90°, 0°, 0°), (30°, 20°, 10°), (25°, 15°, 10°), (-25°, 15°, 10°), (-30°, 20°, 10°), and (-90°, 0°, 0°), respectively, in reference to the robot base.

In this example, it is quite uneasy to find a solution by robot teaching method from many practical experimental tests. The searched path is critical and inefficient for the robot. Especially, inappropriate location of the robot may cause the path planning unlikely. The obstacles are arbitrarily segmented into several rectangular pieces with a 2D top view shown in Figure 2. The location of the robot is investigated by (2) with all the passed points. According to the off-line computation, the robot is located at (0, -850) in reference to the center of the obstacle (0,0) in the X-Y Cartesian coordinate.

The results of path planning via the neighboring, the depth-first and the extensile neighboring methods are presented in Figures 11, 12 and 13, respectively. Though, these methods are able to search a path, the first one needs 33 manipulation steps, the second one needs 16 manipulation steps, and the third one needs 13 steps. In terms of the manipulation time by (3), the first one spends 18.7661 seconds, the second one spends 9.6716 seconds, and the third one spends 8.5054 seconds. It is obvious that the extensile neighboring method saves more manipulation steps, even better than the depth-first search method.
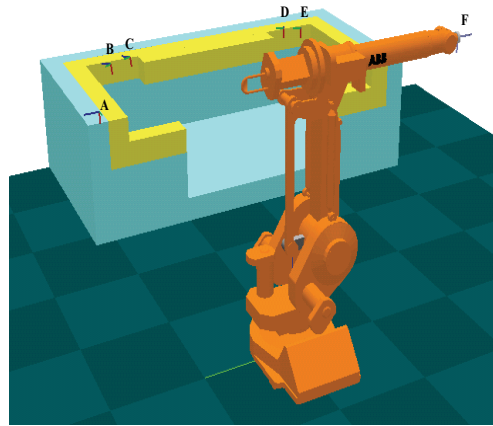
Figure 10. Diagram of automobile baggage trunk
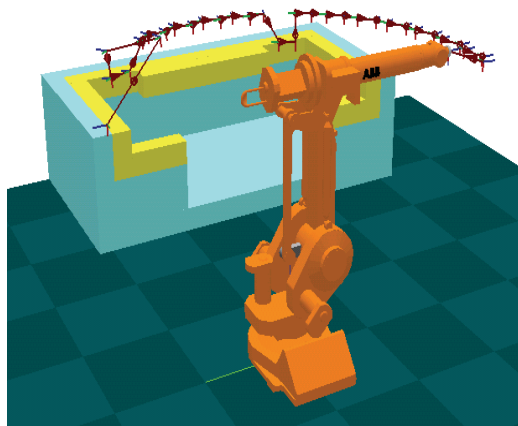
Figure 11. Path planning by neighboring search
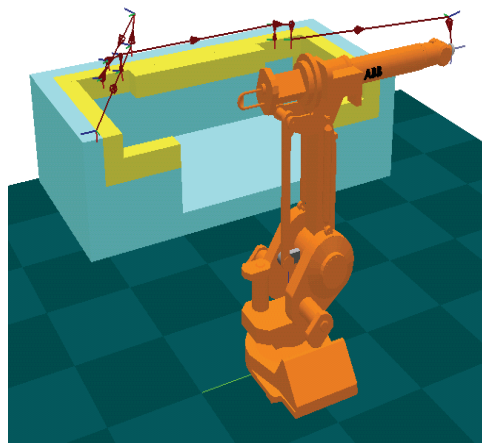
Figure 12. Path planning by depth-first search
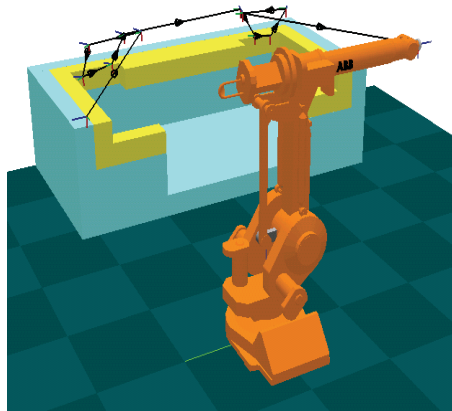
Figure 13. Path planning by extensile neighboring search

7. Conclusion

Investigation of robot location is a necessary procedure previous to path planning. Checking the passable region R by (2) is significant that ensures whether the later path planning is feasible. As to the measure of R in terms of a performance index, large space of R may provide more selections of robot location.

Via the wave expansion method, the passable nodes are marked with numbers. Three methods are proposed for path planning of the industrial robots. The plain neighboring search method is expected to find the path with more manipulation steps. The depth-first method can search a path with fewer manipulation steps, however, it may fail when there does not exist a neighboring node with marked number fewer than the number of the current node. Extensile neighboring search method provides a further reduction of manipulation steps. In general, the searched path needs fewer manipulation steps implies less manipulation time. Also, it is convenient to use the recursive search algorithm to find a path with fewer manipulation steps without the need of intuitively merging the path in the same direction either by the neighboring or the depth-first search methods (Ting et al., 2002). This method, above all, has better performance on searching a path successfully.

A practical automobile baggage trunk is studied to show the capability of determination of robot location and path planning with the developed methods. The extensile neighboring search method not only can trace 26 directions in 3D space, but also can autonomously reduce manipulation steps; therefore, it is an ideal candidate for path planning of industrial robots.

9. References

ABB Robotics, Inc. (1996). *Training Manual S4Lite Robot Studio*

Banski, B. (1996). Local Motion Planning Manipulators Based on Shrinking and Growing Geometry Models, *IEEE International Conference on Robotics and Automation*, pp. 3303-3308, ISSN:1050-4729, Minneapolis, Apr 22-28, USA

Barraquand J.; Langlois B. & Latombe J. C. (1992). Numerical Potential Field Techniques for Robot Path Planning, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, No. 2 (March/April), pp. 222-241, ISSN:0018-9472

Danielle, S. & Mark, H. O. (2001). Motion Planning in Environments with Danger zones, *IEEE International Conference on Robotics and Automation*, pp.1488-1493, ISSN:1050-4729, Seoul, May 21-26, Korea

Jarvis, R. A. (1993). Distance Transform Based Path Planning for Robot Navigation, *Recent Trends in Mobil Robotics*

Latombe, J. C. (1991). *Robot Motion Planning*, Kluwer Academic Publication, Boston

Lei W.-I. (1999). *The Study of Obstacle Avoidance Methods for Industrial Robots*, Master Thesis, Department of Mechanical Engineering, Chung Yuan Christian University, Taiwan

Lozano-Perez, T. (1987). A Simple Motion Planning Algorithm for General Robot Manipulators, *IEEE Journal of Robotics and Automation*, Vol. 3, No. 3 (June), pp. 224-238, ISSN:0882-4967

Pobil, A. P. D. ; Serna, M. A. & Liovet, J. (1992). A New Representation for Collision Avoidance and Detection, *IEEE International Conference on Robotics and Automation*, pp. 246-251, Nice, May 12-14, France

Ralli E. & Hirzinger G. (1994). Fast Path Planning for Robot Manipulators Using Numerical Potential Fields in the Configuration Space, *Proceedings of IEEE/RSJ IROS*, pp.1922-1929, Munich, Sep 12-16, Germany

Red, W. E. & Truong-Cao, H. V. (1996). Unifying Configuration Space and Sensor Space for Vision-Based Motion Planning, *IEEE International Conference on Robotics and Automation*, pp. 3572-3577, Minneapolis, Apr 22-28, USA

Tarjan R. (1972). Depth-first Search and Linear Graph Algorithms, *SIAM Journal of Computing*, Vol. 1, No. 2, pp.146-149

Tesar, D. & Matthew, G. K. (1976). *The dynamic synthesis, analysis, and design of modeled cam systems*, Lexington, MA: Lexington Books, USA

Ting Y. & Lei W.-I. (1999). Using the Hierarchy Tree Method for Robot Path Planning, *IASTED International Conference on Robotics and Applications*, pp. 153~157, Santa Barbara, USA

Ting, Y. ; Lei, W.-I. & Jar, H.-C. (2002). A Path Planning Algorithm for Industrial Robots, *International Journal of Computers & Industrial Engineering*, Vol. 42, No. 2-4 (April), pp. 299~308, ISSN:0360-8352

Homogeneous Approach for Output Feedback Tracking Control of Robot Manipulators

Luis T. Aguilar

1. Introduction

The problem of robust tracking control of electromechanical systems has been studied and solved by many different approaches within the robot control community (see e.g. Sage et al., 1999; Kelly et al., 2005 and references therein) in order to ensure accurate motion in typical industrial tasks (painting, navigation, cutting, etc). In the last decade, the homogeneity approach attracted considerable interest from the research and engineering communities (see e.g. Lebastard et al., 2006; Ferrara et al., 2006; Bartolini et al., 2006) because it was demonstrated that homogeneous systems with homogeneity degree $\eta < 0$ exhibit robustness and finite-time convergence properties (Bhat & Bernstein, 1997; Hong et al., 2001; Orlov, 2005).

Control laws based on the homogeneity approach (Bhat & Bernstein, 1997; Hermes, 1995; Orlov, 2003a; Rosier, 1992) are attractive in robotic applications because they can cope with many mechanical perturbations, including external vibrations, contact forces, and nonlinear internal phenomena such us Coulomb and viscous friction, dead zone and backlash, while it is possible to ensure exact tracking to continuously differentiable desired trajectories.

Several homogeneous controllers and studies have been proposed in the literature. For example, Rosier (1992) constructed a homogeneous Lyapunov function associated with homogeneous dynamic systems. Hermes (1995) addressed the homogeneous stabilization control problem for homogeneous systems. Bhat and Bernstein (1997) examined the finite time stability of homogeneous systems. Levant (2005a, 2005b) developed robust output-feedback high-order sliding mode controllers that demonstrate finite-time convergence (see also (Fridman & Levant, 1996; Fridman & Levant, 2002)) where the controller design is based on homogeneity reasoning while the accuracy is improved in the presence of switching delay, and the chattering effect is treated by increasing the relative degree. Orlov et al. (2003a, 2003b) proposed applying homogeneous controller to solve the set-point problem dealing with mechanical imperfections such as Coulomb friction, viscous friction, and backlash. Orlov et al., (2005) extended the finite time stability analysis to nonlinear nonautonomous switched systems.

Motivated by the above-mentioned features, and taking into account that only incomplete and imperfect state measurements are available, the main objective of this paper is introduce an output feedback homogeneous controller for tracking the trajectories of robot manipulators. The control design proposed here is inspired by the passivity-based approach, which consists of an observer part, a precomputed reference trajectory, and a controller part, but is augmented with a relay part in both the controller and the observer, which yields a certain degree of robustness under disturbances and finite-time stability of the closed-loop system. We note that the passivity-based approach, which is represented by the Slotine and Li + observer controller, allows semi-global stability and the addition of the relay terms ensures robustness, despite the presence of the observer without destroying the closed-loop stability.

This paper is organized as follows: In Section 2 homogeneous systems are defined. Section 3 states the problem and introduces the Euler-Lagrange representation of the robot manipulator, along with some fundamental properties of the dynamic model. Section 4 presents the homogeneous controller and its stability analysis. Section 5 provides a simulation study for a 2-DOF robot manipulator using the controller described in Section 4 as well as performance of the controller for external perturbations. Section 6 establishes final conclusions. The following notation will be used throughout the paper. The norm $|x|_2$, with $x \in R^n$, denotes the Euclidean norm and $|x|_1 = |x_1| + \ldots + |x_n|$ stands for the sum norm. The minimum and maximum eigenvalues of a matrix $A \in R^{n \times n}$ are denoted by $\lambda_{\min}\{A\}$ and $\lambda_{\max}\{A\}$, respectively. The vector sign(x) is given by sign(x)=[sign(x_1),...,sign(x_n)]^T where the signum function is defined as

$$sign(y) = \begin{cases} 1 & if \quad y > 0, \\ [-1,1] & if \quad y = 0, \forall y \in R. \\ -1 & if \quad y < 0, \end{cases} \qquad (1)$$

2. Basic definitions

Let us begin by recalling some definitions of homogeneity for nonautonomous nonlinear systems governed by

$$\dot{x} = f(x,t) \qquad (2)$$

where $x = (x_1, \ldots, x_n)^T$ is the state vector, t is the time, and $f = (f_1, \ldots, f_n)^T$ is a piece-wise continuous function (Orlov, 2005). The function $f : R^{n+1} \mapsto R^n$ is piece-wise continuous if and only if R^{n+1} is partitioned into a finite number of

domains $G_j \subset R^{n+1}$, $j = 1, \ldots, N$, with disjoints interiors and boundaries ∂G_j of measure zero such that f is continuous within each of these domains and for all $j = 1, \ldots, N$ it has a finite limit $f^j(x, t)$ as the argument $(x^*, t^*) \in G_j$ approaches a boundary point $(x, t) \in \partial G_j$. Throughout, the precise meaning of the differential equation (2) with a piece-wise continuous right-hand side is defined in the sense of Filippov (Filippov, 1988). An absolutely continuous function $x(\cdot)$ defined on an interval I, is said to be a solution of (2) if it satisfies the differential inclusion

$$\dot{x} \in F(x, t) \tag{3}$$

almost everywhere on I.

Definition 1 (Orlov, 2005): A piece-wise continuous function $f : R^{n+1} \to R^n$ is said to be homogeneous of degree $\eta \in R$ with respect to dilation $(r_1 \ldots r_n)$ where $r_i > 0$, $i = 1, \ldots, n$ if there exists a constant $c > 0$ and a ball $B_\delta \subset R^n$ such that

$$f_i(c^{r_1} x_1, \ldots, c^{r_n} x_n, c^{-\eta} t) = c^{\eta + r_i} f_i(x).$$

for all $c > c_0$ and almost all $(x, t) \in B_\delta \times R$.

When continuous, a globally homogeneous time-invariant vector field $f(x)$ of degree $\eta < 0$ with respect to dilation $(r_1, \ldots, r_n)$ is known to be globally finite time stable whenever it is globally asymptotically stable (Orlov 2005, Thm 3.1) and an upper estimate of the settling time is given by

$$T(t_0, x^0) \leq \tau(x^0, E_L) + \frac{1}{1 - 2^\eta} \left(\delta L^{-1} \right)^\eta s(\delta)$$

where

$$x(t_0) = x^0 \text{ and }$$

$$\tau(x^0, E_L) = \sup_{x(\cdot, t_0, x^0)} \inf \left\{ T \geq 0 : x(t, t_0, x^0) \in E_L \forall t_0 \in R, t \geq t_0 + T \right\}$$

and

$$s(\delta) = \sup_{x^0 \in E_\delta} \tau \left(x^0, E_{\frac{1}{2}\delta} \right),$$

where E_L denotes an ellipsoid of the form

$$E_L = \left\{ x \in R^n : \sqrt{\sum_{i=1}^{n} \left(\frac{x_i}{L^{r_i}} \right)^2} \leq 1 \right\},$$

E_L is located within a homogeneity ball, $\delta \geq c_0 L$, and $c_0 > 0$ is a lower estimate of the homogeneity parameter.

3. Dynamic model and problem statement

We here present a homogeneous tracking control for an n-degrees-of-freedom rigid serial links robot manipulator governed by the following equation of motion (Spong, 1989):

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = U + w \tag{4}$$

where q is the $n \times 1$ vector of joint positions and is considered to be the only information available for feedback; U is the $n \times 1$ vector of applied joint torques; w is the $n \times 1$ unknown perturbation vector; $M(q)$ is the n×n symmetric positive-definite inertia matrix; $C(q,\dot{q})\dot{q}$ is the $n \times 1$ vector of centripetal and Coriolis forces; and $g(q)$ is the $n \times 1$ vector of gravitational torques. The dynamic equation (4) has the following properties, which will be used in the closed-loop stability analysis (Kelly et al., 2005):

- The inertia matrix $M(q)$ is bounded above and below for all $q \in R^n$; that is, $m_1 I \leq \|M(q)\| \leq m_2 I$ where m_1 and m_2 are positive scalars and I is the identity matrix.
- The matrix $C(q;\dot{q})$ is chosen such that the relation $\dot{q}^T[M(q) - 2C(q;\dot{q})]\dot{q} = 0$ holds for all $q, \dot{q} \in R^n$.
- The vector $C(q,x)y$ satisfies $C(q,x)y=C(q,y)x$ and $C(q,z+\alpha x)y=C(q,z)y+\alpha C(q,x)y$ for all $q,x,y,z \in R^n$ and $\alpha \in R$.
- The matrix $C(q,x)$ satisfies $\|C(q,x)\| \leq k_c\|x\|$ for all $x, q \in R^n$ and k_c is a positive constant.

We further assume

- A known constant, $W > 0$, is the upper boundary for the perturbation vector, w, that is, $|w| \leq W$.

The control objective is established formally as follows: given bounded and continuously differentiable desired joint trajectories $q_d(t) \in R^n$, we must design a discontinuous control law U such that the joint positions $q(t)$ reach the desired trajectories $q_d(t)$ asymptotically; that is,

$$\lim_{t \to \infty} |q_d(t) - q(t)| = 0. \tag{5}$$

4. Homogeneous controller

In this section, we present the nonadaptive Slotine-and-Li controller (Slotine & Li, 1987) augmented with a homogeneous part to achieve asymptotic stability of the closed-loop system equilibrium point.

Proposition 1. *Consider the equation of the robot (2) along with the following control law:*

$$\begin{cases} U &= M(q)\ddot{q}_r + C(q,\dot{q}_0)\dot{q}_r + g(q) - K_D(\dot{q}_0 - \dot{q}_r) \\ &\quad - K_P e - K_\alpha sign(e) - K_\beta sign(\dot{q}_0 - \dot{q}_r) \\ \dot{q}_r &= \dot{q}_d - \Lambda(\hat{q} - q_d) \\ \dot{q}_0 &= \dot{\hat{q}} - \Lambda z \end{cases} \tag{6}$$

and the homogeneous observer,

$$\begin{cases} \dot{\hat{q}} &= p + (\Lambda + l_d I)z \\ \dot{p} &= \ddot{q}_r + l_d \Lambda z + M^{-1}(q)K_P[z - e + \gamma sign(z)] \\ &\quad - M^{-1}(q)[K_\alpha sign(e) + 2K_\beta sign(\dot{q}_0 - \dot{q}_r)] \end{cases} \tag{7}$$

where $e=q-q_d$ is the n×1 tracking error vector; $\hat{q}$ is the n×1 estimated velocity vector; $z = q - \hat{q}$ is the n×1 observation error vector; Λ, K_P, K_D, K_α, K_β, γ are n×n diagonal positive definite matrices and l_d is a positive constant. Then, for any initial condition some sufficiently large gains of the controller (6), (7) always exist such that (5) holds.

Proof. First, the equations of the closed-loop system [(4), (6), (7)] must be introduced in terms of the tracking and observation errors, which are given, respectively, by

$$\frac{d}{dt}\begin{bmatrix} e \\ z \\ s \\ r \end{bmatrix} =$$

$$\begin{bmatrix} \dot{e} \\ \dot{z} \\ M^{-1}(q)[K_D r - C(q,r)\dot{q}_r - C(q,\dot{q})s - K_P e + w - K_D s - K_\alpha sign(e) - K_\beta sign(s-r)] \\ M^{-1}(q)[K_D r - C(q,r)\dot{q}_r - C(q,\dot{q})s - K_D s - K_P z + w - l_d M(q)r + K_\beta sign(s-r) - \gamma sign(z)] \end{bmatrix} \qquad (8)$$

where $s = \dot{q} - \dot{q}_r = \dot{e} + \Lambda(e-z)$ and $r = \dot{q} - \dot{q}_0 = \dot{z} - \Lambda z$.

It should be noted that the nonlinear nonautonomous closed-loop system (8) is a differential equation with a right-hand discontinuous side. Thus, the precise meaning of solutions of the differential equation with the discontinuous functions is defined in the Filippov sense (Filippov, 1988), as for the solutions of a certain differential inclusion with a multi-valued right-hand side. In fact, the control law (6)-(7) can be seen as a second-order sliding mode.
To conclude that the origin is asymptotically stable, consider the following Lyapunov function candidate for the closed-loop system (8):

$$V(x,t) = \frac{1}{2}e^T K_P e + \frac{1}{2}s^T M(q)s + \frac{1}{2}z^T K_P z + \frac{1}{2}r^T M(q)r + \sum_{i=1}^{n} K_{\alpha i}|e_i| + \sum_{i=1}^{n} \gamma_i |z_i| \qquad (9)$$

where $K_{\alpha i}$ and γ_i are the elements of the main diagonal of K_α and γ, respectively. The time derivative of $V(x,t)$ along the solution of (8) yields

$$\begin{aligned} \dot{V}(x,t) &= e^T K_P \dot{e} + s^T M(q)\dot{s} + \frac{1}{2}s^T \dot{M}(q)s + z^T K_P \dot{z} + r^T M(q)\dot{r} + \frac{1}{2}r^T \dot{M}(q)r \\ &\quad + \dot{e}^T K_\alpha sign(e) + \dot{z}^T \gamma sign(z). \end{aligned}$$

Substituting equations (8) in $\dot{V}(x,t)$ and employing properties H2 and H3, it follows that

$$\begin{aligned} \dot{V}(x,t) &= -e^T \Lambda K_P e + e^T \Lambda K_P z - z^T \Lambda K_P z - s^T K_D s - r^T[M(q)l_d - K_D]r + s^T C(q,r)[s - \dot{q}] \\ &\quad + r^T C(q,s)[r - \dot{q}] - e^T \Lambda K_\alpha sign(e) - (s-r)^T K_\beta sign(s-r) - z^T \Lambda \gamma sign(z) \\ &\quad + z^T \Lambda K_\alpha sign(e) + (s+r)^T w. \end{aligned} \qquad (10)$$

Using properties H1, H2, and H5; and employing the well-known inequality the following boundary is obtained

$$2\|g\|\|h\| \le \|g\|^2 + \|h\|^2, \qquad\qquad g, h \in R^n, \qquad\qquad (11)$$

$$
\begin{aligned}
\dot{V}(x,t) \;=\; & -\frac{1}{2}\begin{bmatrix}\|e\|\\\|z\|\end{bmatrix}^T \underbrace{\begin{bmatrix}\lambda_{\min}\{\Lambda K_P\} & 0 \\ 0 & \lambda_{\min}\{\Lambda K_P\}\end{bmatrix}}_{Q_1}\begin{bmatrix}\|e\|\\\|z\|\end{bmatrix} - \Big[\underbrace{\lambda_{\min}\{\Lambda\gamma\}-\lambda_{\max}\{\Lambda K_\alpha\}}_{Q_2}\Big]\|z\| \\
& -\begin{bmatrix}\|s\|\\\|r\|\end{bmatrix}^T \underbrace{\begin{bmatrix}\lambda_{\min}\{K_D\}-k_c\|r\| & k_c\|\dot{q}\| \\ k_c\|\dot{q}\| & m_1 l_d - \lambda_{\max}\{K_D\}-k_c\|s\|\end{bmatrix}}_{Q_3}\begin{bmatrix}\|s\|\\\|r\|\end{bmatrix} \\
& -\lambda_{\min}\{\Lambda K_\alpha\}\|e\| - \lambda_{\min}\{K_\beta\}\|s-r\| + \lambda_{\max}\{W\}\|s+r\|.
\end{aligned}
\qquad (12)
$$

We now derive sufficient conditions for $\dot{V}(x,t)$ to be locally negative definite. Firstly, Q_1 will be positive definite by selecting positive definite matrices Λ and K_P. Notice that Q_2 will be positive definite if

$$\lambda_{\min}\{\Lambda\gamma\} > \lambda_{\max}\{\Lambda K_\alpha\}.$$

Finally, Q_3 is positive definite if

$$\frac{1}{k_c}\lambda_{\min}\{K_D\} > \|x\|$$

and

$$m_1 l_d > \lambda_{\max}\{K_D\}.$$

Thus it is always possible to find some controller gains to ensure that all the above inequalities hold. Therefore, (12) is locally negative definite almost everywhere and the equilibrium point is exponentially stable.

Finally, we define the domain of attraction and prove that it can be enlarged by increasing the controller gains. For this, we first find some positive constant α_1, α_2 such that

$$\alpha_1\|x\|_2^2 \le V(x,t) \le \alpha_2\|x\|_2^2. \qquad (13)$$

Notice from (8) that

$$V > \frac{1}{2}\Big[\lambda_{\min}\{K_P\}\|e\|^2 + m_1\|s\|^2 + \lambda_{\min}\{K_P\}\|z\|^2 + m_1\|r\|^2\Big]$$

so we define α_1 as

$$\alpha_1 = \frac{1}{2} \min\{\lambda_{\min}\{K_P\}, m_1\}.$$
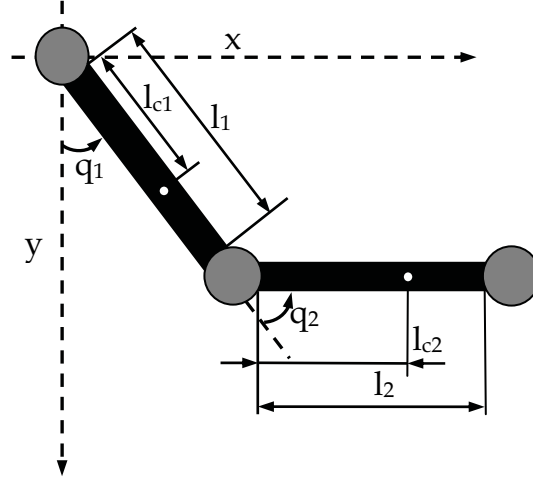
Figure 1. Schematic diagram of the 2-DOF robot manipulator.

In a similar manner, an upper bound on (8) is

$$
\begin{aligned}
V \quad \leq \quad & \frac{1}{2}\Big[(\lambda_{\max}\{K_P\} + 2n\lambda_{\max}\{K_\alpha\})\|e\|^2 + m_2\|s\|^2 + m_2\|r\|^2 \\
& + (\lambda_{\max}\{K_P\} + 2n\lambda_{\max}\{\gamma\})\big]\|z\|^2
\end{aligned}
$$

so we define

$$\alpha_2 = \max\{(\lambda_{\max}\{K_P\} + 2n\lambda_{\max}\{K_\alpha\}), m_2, (\lambda_{\max}\{K_P\} + 2n\lambda_{\max}\{\gamma\})\}.$$

From (12), (13) we conclude that the domain of attraction contains the set

$$\|x\| \leq \frac{1}{k_c} \lambda_{\min}\{K_D\} \sqrt{\frac{\alpha_1}{\alpha_2}}. \tag{14}$$

5. Simulation results

To study the performance of the controllers, we generate simulations in which a two-link manipulator was required to follow the joint trajectory

$$q_{di} = \pi + \frac{\pi}{2}\cos(2\pi ft), \qquad\qquad i = 1,2$$

where f=10 Hz. The position and velocity initial conditions were set to $q(0) = \dot{q}(0) = 0 \in R^2$. The motion of the 2-DOF manipulator with rotational joints, depicted in Figure 1, was governed by (4) where

$$M(q) = \begin{bmatrix} 8.77 + 2.02\cos(q_2) & 0.76 + 0.51\cos(q_2) \\ 0.76 + 0.51\cos(q_2) & 0.62 \end{bmatrix},$$

$$C(q,\dot{q}) = \begin{bmatrix} -0.51\sin(q_2)\dot{q}_2 & -0.51\sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ 0.51\sin(q_2)\dot{q}_1 & 0 \end{bmatrix},$$

And

$$g(q) = 9.8\begin{bmatrix} 7.6\sin(q_1) + 0.63\sin(q_1 + q_2) \\ 0.63\sin(q_1 + q_2) \end{bmatrix}$$

were taken from (Berghuis & Nijmeijer, 1993a). In the simulation, the control gains were selected as follows:

K_P=diag{10,10} K_D=diag{5,5}
K_α=diag{5,5} K_β=diag{2,2}
γ=diag{12,12}.

The resulting position and observation errors of the closed-loop system (8) for the unperturbed and perturbed case are depicted in Figure 2. This figure also shows the control input. The figure demonstrates that the homogeneous controller asymptotically stabilizes the manipulator in the desired trajectory, thus satisfying the control objective (5). Figure 3 shows chattering due to the relay part of the control law (last two terms of (6)):

$$u_h = -K_\alpha sign(e) - K_\beta sign(\dot{q}_0 - \dot{q}_r).$$

For the sake of comparison, both controllers (6)-(7) and the controller without a

relay part ($K_\alpha = K_\beta = \gamma = 0$) that were proposed by Berghuis and Nijmeijer (1993b) were simulated assuming permanent disturbances ($w=10$). In contrast to the proposed controller, the simulations results depicted in Figure 4 show that the continuous controller drives position of each joint of the manipulator to a steady-stable error of 0.3 [rad]. We repeated the last simulations for a reference trajectory consisting of a circle in task space with radius $r=0.25$ [m] and center (1,1/2) which is given by

$$
\begin{cases}
x_d = 1 + \dfrac{1}{4}\cos(0.1t) \\
y_d = \dfrac{1}{2} + \dfrac{1}{4}\sin(0.1t);
\end{cases}
$$

where x_d and y_d are coordinates in the Cartesian space and the initial position in task space is fixed at $x(0) = 2$ [m] and $y(0) = 0$ [m]. The corresponding trajectory in the joint space is

$$
q_{d1} = \tan^{-1}\left(\frac{x_d}{y_d}\right) - \tan^{-1}\left(\frac{l_2 \sin(q_{d2})}{l_1 + l_2 \cos(q_{d2})}\right), \qquad q_{d2} = \cos^{-1}\left(\frac{x_d^{\,2} + y_d^{\,2} - l_1^{\,2} - l_2^{\,2}}{2l_1 l_2}\right)
$$

where l_1 and l_2 are the lengths of links 1 and 2 respectively. Figure 5 illustrates the effectiveness of the controller.
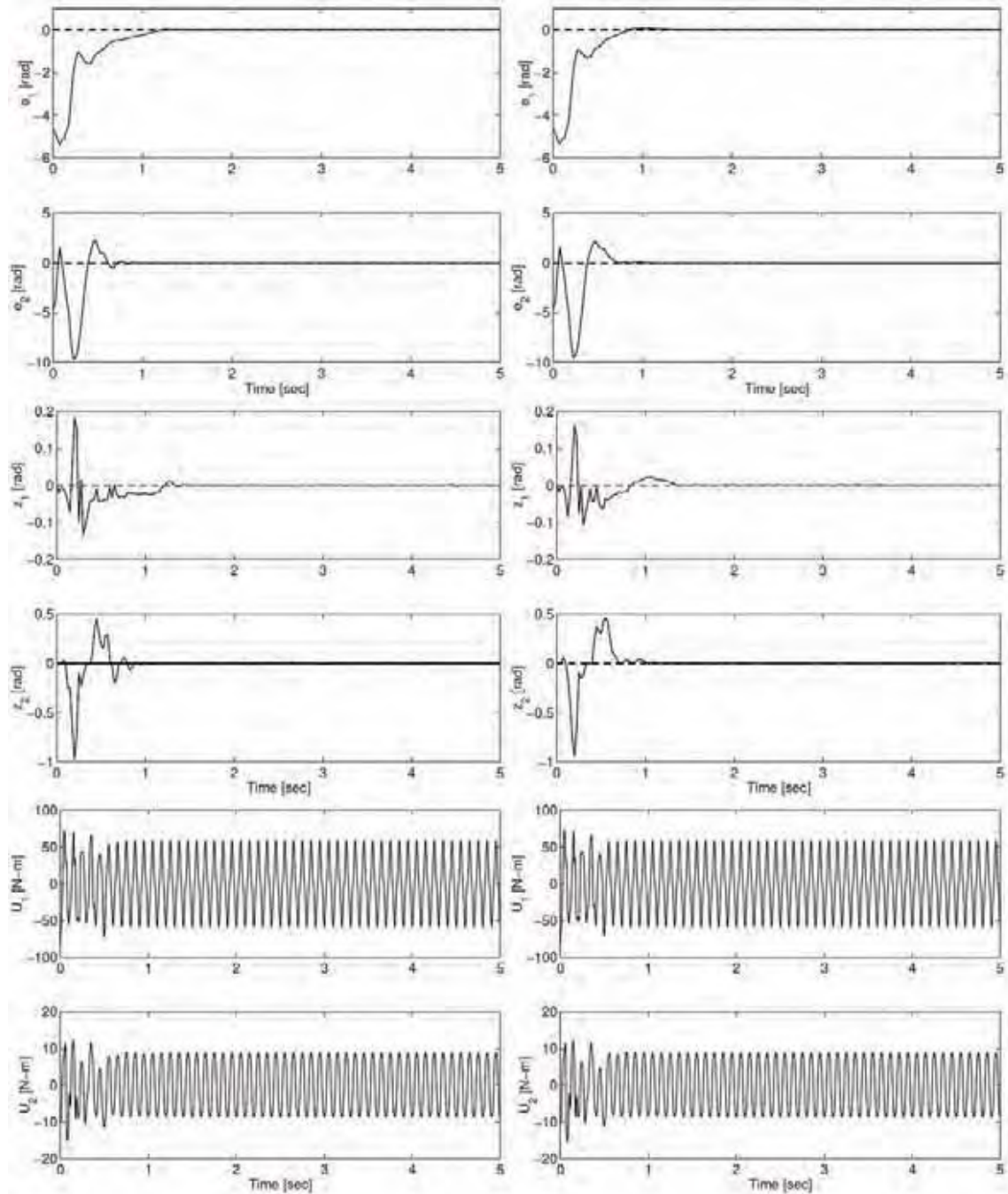
Figure 2. Joint position errors, observation errors, and control input for the unperturbed case (*left column*) and perturbed case (*right column*).
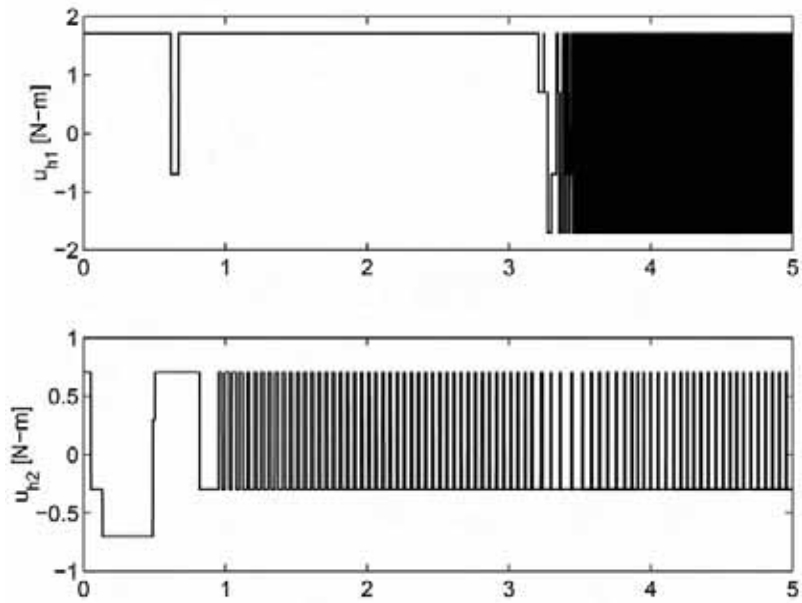
Figure 3. Chattering signal due to discontinuous terms in the input control.
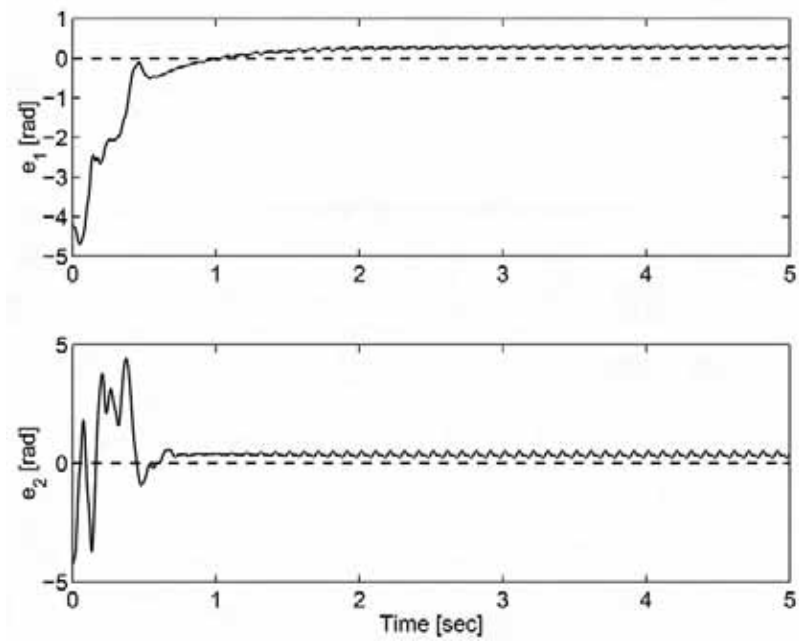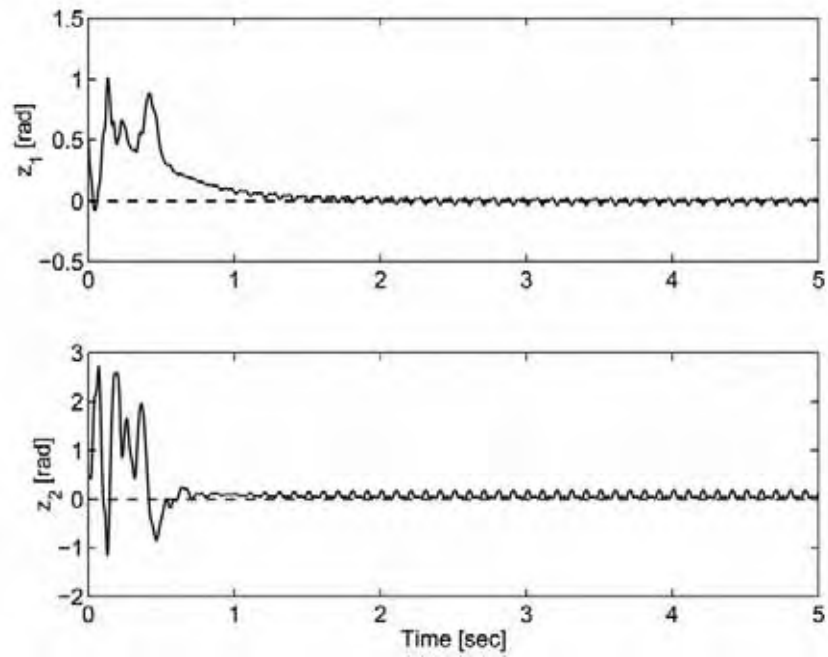
Figure 4. Joint position errors for the continuous controller (i.e., $K_\alpha=K_\beta=\gamma=0$): The perturbed case.

(a)

(b)
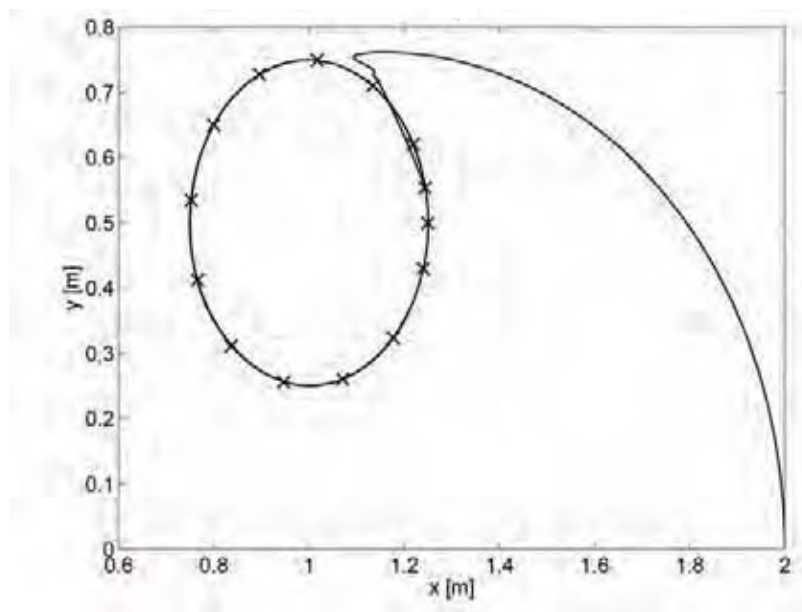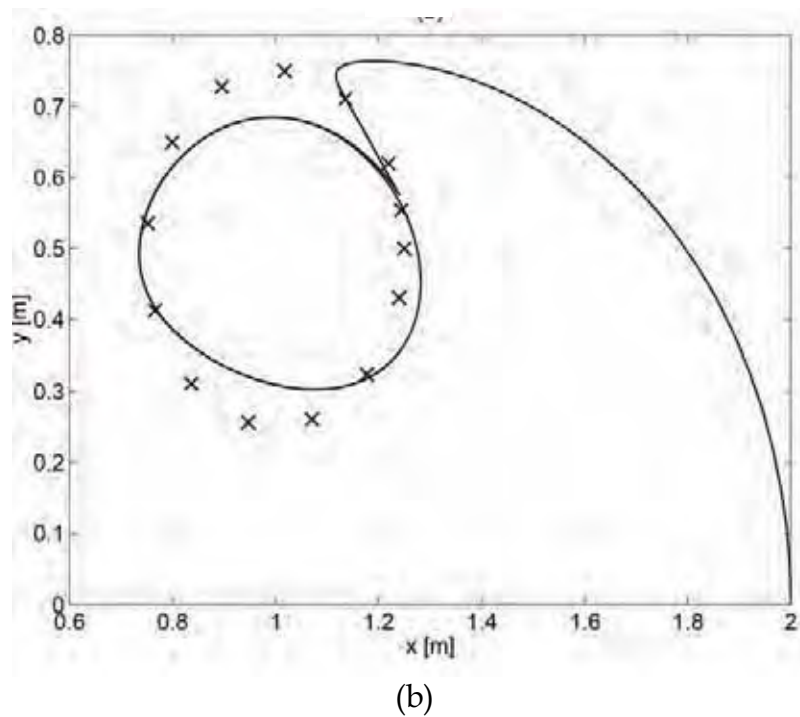
Figure 5: Motion of the end effector of the manipulator following a circular path (x) using the homogeneous controller (*a*) and the continuous controller (*b*).

6. Conclusions

We developed a controller that exploits the advantages of homogeneity theory and that is applicable to the tracking control problem for *n*-degrees of freedom robot manipulators, assuming that position information is the only available feedback data. The basis of this work is the passivity-based approach, which consists of a tracking controller plus an observer complemented with a relay part that yields a certain degree of robustness in comparison with its continuous counterpart. Stability analysis was developed within the nonsmooth Lyapunov function framework (Baccioti & Rosier, 2005) where semi-global asymptotical stability has been concluded. The effectiveness of the controller was supported by simulations made for a two degrees-of-freedom robot manipulator taking into account the unperturbed and perturbed cases.

7. References

Baccioti, A. & Rosier, L. (2005). *Liapunov functions and stability in control theory: 2nd edition*, Springer, 978-3-540-21332-1, Berlin.

Bartolini, G., Orani, N.; Pisano, A. & Usai, E. (2006). Higher-order sliding mode approaches for control and estimation in electrical drives, In: *Advances in variable structure and sliding mode control*, C. Edwards, E. Fossas & L. Fridman, (Ed.), 423-445, Springer, 3-5403-2800-9, London.

Berghuis, H. & Nijmeijer, H. (1993a). Global regulation of robots using only position measurements. *Systems and Control Letters*, Vol. 21, 289-293, 0167-6911.

Berghuis, H. & Nijmeijer, H. (1993b). A passivity approach to controller-observer design for robots. *IEEE Transactions on Automatic Control*, Vol. 9, No. 6, 740-754, 0018-9286.

Bhat, S. & Bernstein, D. (1997). Finite time stability of homogeneous systems, *Proceedings of the 1997 American Control Conference*, pp. 2513-2514, 0-7803-3832-4, Alburquerque, USA, June 1997, IEEE.

Ferrara, A.; Giacomini, L. & Vecchio, C. (2006). Control of nonholonomic systems with uncertainties via second order sliding modes, *Proceedings of the 2006 American Control Conference*, pp. 5384-5389, 1-4244-0210-7, Minneapolis, USA, June 2006, IEEE.

Filippov, A.F. (1988). *Differential equations with discontinuous right-hand sides*, Kluwer academic, 9-0277-2699-X, Dordrecht.

Fridman, L. & Levant, A. (1996). Higher order sliding modes as a natural phenomenon in control theory, In *Robust Control via Variable Structure and Lyapunov Techniques* Garofalo & Glielmo, (Ed.), Lectures notes in control and information science, 217, 107-133, Springer, 3-5407-6067-9, Berlin.

Fridman, L. & Levant, A. (2002). Higher order sliding modes, In *Sliding mode control in engineering*, W. Perruquetti and J.P. Barbot, (Ed.), 53-102, Marcel Dekker, 0-8247-0671-4, New York.

Hermes, H. (1995). Homogeneous feedback control for homogeneous systems. *Systems and Control Letters*, Vol. 24, 7-11, 0167-6911.

Hong, Y.; Huang, J. & Xu, Y. (2001). On an output feedback finite-time stabilization problem. *IEEE Transactions on Automatic Control*, Vol. 46, No. 2, 305-309, 0018-9286.

Kelly, R.; Santibañez, V. & Loría, A. (2005). *Control of robot manipulators in joint space*, Springer, 1-8523-3994-2, London.

Lebastard, V.; Aoustin, Y.; Plestan, F. & Fridman, L. (2006). Absolute orientation estimation based on high order sliding mode observer for a five link walking biped robot. *Proceedings. of the International Workshop on Variable Structure Systems*, pp. 373-378, 1-4244-0208-5, Alghero, Italy,.

Levant, A. (2005a). Homogeneity approach to high-order sliding mode design. *Automatica*, Vol. 41, 823-830, 0005-1098.

Levant, A. (2005b). Quasi-continuous high-order sliding-mode controllers. *IEEE Transactions on Automatic Control*, Vol. 50, No. 11, 1812-1816, 0018-9286.

Orlov, Y.; Aguilar, L. & Cadiou, J.C. (2003a). Switched chattering control vs. backlash/friction phenomena in electrical servo-motors. *International Journal of Control*, Vol. 76, 959-967, 0020-7179.

Orlov, Y.; Alvarez, J.; Acho, L. & Aguilar, L. (2003b). Global position regulation of friction manipulators via switched chattering control. *International Journal of Control*, Vol. 76, 1446-1452, 0020-7179.

Orlov, Y. (2005). Finite time stability and robust control synthesis of uncertain switched systems. *SIAM J. Control Optim.*, Vol. 43, No. 4, 1253-1271, .

Rosier, L. (1992). Homogeneous Lyapunov function for homogeneous continuous vector field. *Systems & Control Letters*, Vol. 19, 467-473, 0167-6911.

Sage, H.; De Mathelin, M. & Ostertag, E. (1999). Robust control of robot manipulators: a survey. *International Journal of control*, Vol. 72, 1498-1522, 0020-7179.

Slotine J., & Li, W. (1987). On the adaptive control of robot manipulators. *International Journal of Robotics Research*, Vol. 6, No. 3, 49-59.

Spong M. & Vidyasagar, M. (1989). *Robot dynamics and control*, John Wiley & Sons, 0-4716-1243-X, New York.

Design and Implementation of Fuzzy Control for Industrial Robot

Muhammad Suzuri Hitam

1. Introduction

The dynamic equations of motion for a mechanical manipulator are highly non-linear and complex. It is therefore, very difficult to implement real-time control based on a detailed dynamic model of a robot, if not impossible (Luh et al., 1980; Lee et al., 1982). The control problem becomes more difficult if adaptive control is necessary to accommodate changing operational conditions. Such a requirement frequently exits in the manufacturing environment; therefore, an alternative design approach would be attractive to the industrial practitioner. A better solution to the complex control problem might result if human intelligence and judgement replaces the design approach of finding an approximation to the true process model. A practical alternative would be the use of fuzzy logic. It has been reported that fuzzy logic controllers performed better, or at least as good as, a conventional controller and can be employed where conventional control techniques are inappropriate (Li et al., 1989; Sugeno, 1985; Ying et al., 1990). In contrast to adaptive control, fuzzy logic algorithms do not require a detailed mathematical description of the process to be controlled and therefore the implementation of fuzzy logic should, theoretically, be less demanding computationally. Fuzzy logic algorithms can be designed for environments where the available source information is not accurate, subjective and of uncertain quality. Furthermore, these algorithms provide a direct means of translating qualitative and imprecise linguistic statements on control procedures into precise computer statements. In this chapter, a proposed fuzzy logic design to control an actual industrial robot arm is outlined. The description of fuzzy logic controller is described in Section 2. It includes the methodology for the design of a fuzzy logic controller for use in robotic application. Section 3 presents the robot control system architecture. In Section 4, the relevant issues that arise relating to the design techniques employed are discussed in detailed. These issues include choise of sampling time, fuzzy rules design strategy, and controller tuning strategy. To evaluate the effectiveness of the proposed design strategy, studies are made to

investigate which design strategy leads to the best control performance under various robot conditions. Section 5 concludes this chapter.

2. Description of Fuzzy Logic Controller Architecture

The basic structure of the fuzzy logic controller (FLC) most commonly found in the literature is presented in Fig. 1 (Lee, 1990a). The basic configuration of a fuzzy system is composed of a fuzzification interface, a knowledge base, a fuzzy inference machine and a defuzzification interface as illustrated in the upper section of Fig. 1. The measured values of the crisp input variables are mapped into the corresponding linguistic values or the fuzzy set universe of discourse at the fuzzification interface. The knowledge base comprises both the fuzzy data and fuzzy control rules. The fuzzy data base contains all the necessary definitions used in defining the fuzzy sets and linguistic control rules whereas, the fuzzy control rule base includes the necessary control goals and control policy, as defined by an experts, in the form of a set of linguistic rules. The fuzzy inference engine emulates human-decision making skills by employing fuzzy concepts and inferring fuzzy control actions from the rules of inference associated with fuzzy logic. In contrast to the fuzzification stage, the defuzzification interface converts the values of the fuzzy output variables into the corresponding universe of discourse, which yields a non-fuzzy control action from the inferred fuzzy control action.

In general, for a regulation control task, the fuzzy logic controller maps the significant and observable variables to the manipulated variable(s) through the chosen fuzzy relationships. The feedback from the process output is normally returned a crisp input into the fuzzification interface. The crisp or non-fuzzy input disturbance, illustrated in Fig. 1, would normally include both error and change in error, and these are mapped to their fuzzy counterparts at the fuzzification stage. These latter variables are the inputs to the compositional rules of inference from which the fuzzy manipulated variable is obtained. At the output from the defuzzification process, a crisp manipulated variable is available for input to the process. In conclusion, it can be stated that to design a fuzzy logic controller, six essential stages must be completed:

1. Input and output variables to be used must be identified.
2. Design the fuzzification process to receive the chosen input variables.
3. Establish the data and rule bases.
4. Select the compositional rule of inference for decision making.
5. Decide which defuzzification process is to be employed.
6. Develop the computational units to access the data and rule bases.
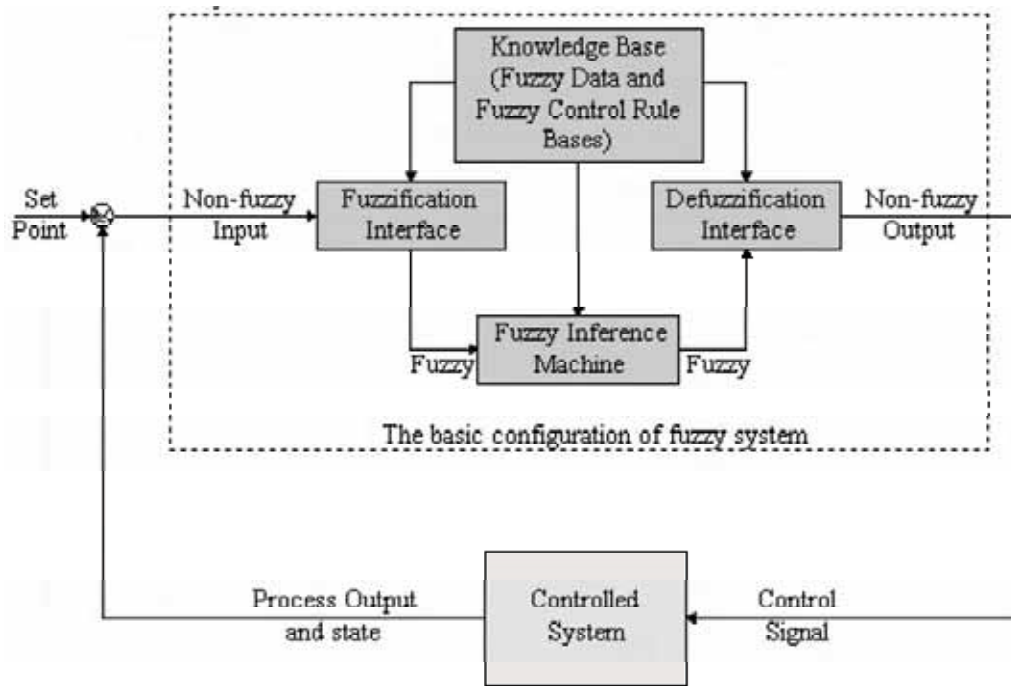
Figure 1. The general form of the fuzzy logic control architecture

2.1 Input and Output Variables

In any fuzzy logic control system, the observed input must be fuzzified before it is introduced to the control algorithm. The most commonly used antecedents at this fuzzification stage are the state variables, error and rate of change in error. For the case of positioning a joint within a robot arm, the first variable is the difference (error) between the desired and the current joint position. The value of the second state variable is the numerical difference between two successive values of error (change in error). These two state variables give a good indication of the instantaneous performance of the system and both variables are quantifiable by fuzzy sets. In this project, error (E) and change in error (CE) are defined as the input fuzzy sets and the controlled action (CU) as the output fuzzy set. The evaluation of the error and the change in error at sample interval, k, is calculated as follows :

$$\text{Error}(\,k\,) = \text{Demand}(\,k\,) - \text{Actual position}(\,k\,) \tag{1}$$

$$\text{Change in error}(\,k\,) = \text{Error}(\,k\,) - \text{Error}(\,k - 1) \tag{2}$$

2.2 Method of Representing Fuzzy Sets

According to Lee (1990a), there are two methods for defining a fuzzy set; nu-

merical and functional, depending on whether the universe of discourse is discrete or continuous. In the case of a discrete universe, a numerical definition is employed where the value of the membership function is represented by a vector; the order of the vector dependent on the degree of discretisation. The user has to specifically define the grade of membership of each cardinal in the fuzzy sets. For a continuous universe of discourse, a functional definition can be utilised to define the membership function of a fuzzy set. The triangle, trapezoidal and the bell shaped functions are the popular types found in many engineering applications. In this Chapter, this latter form of representation is adopted. The evaluation of the membership function is evaluated on-line during process operation. A combination of bisected trapezoidal, trapezoidal and triangular shaped fuzzy set templates are used to represent the input and output variables; template shapes that are readily evaluated and require the minimum of computer memory storage. At present, researchers are still looking for the best guidance to determine the best shape for a fuzzy set to provide an optimum solution to a specific control problem. In general, the use of simple shapes could provide satisfactory performance. The geometry of these templates can be defined by the base width and the side slope when mapped to the universe of discourse.

2.2.1 Mapping Fuzzy Sets to the Universe of Discourse

In any application, it is essential for a practitioner to identify the most appropriate parameters prior to the mapping of the fuzzy sets to the chosen universe of discourse; the determination of the size of both the measurement and control spaces; the choice of the discretisation levels for both the measurement and control spaces, the definition of the basic fuzzy sets within these discretised spaces and finally the sample interval to be used. The size of both the measurement and control spaces can be directly determined by estimating the probable operating range of the controlled system. However, the choice of the discretisation levels in both the measurement and control spaces, and the fuzzy set definitions can only be defined subjectively and are normally based on the experience and judgement of the design engineer. From a practical point of view, the number of quantisation levels should be large enough to provide an adequate resolution of the control rules without demanding excessive computer memory storage. Generally 5 to 15 level of discretisations are found to be adequate. It should be emphasised that the choice of these parameters has a significant influence on the quality of the control action that can be achieved in any application (Lee, 1990a). The use of higher resolution in the discretisation levels will result in an increase in the number of control rules and thereby make the formulation of these control rules more difficult. It should also be emphasised that the fuzzy sets selected should always completely cover the whole of the intended working range to ensure that proper

control action can be inferred for every state of the process. The union of the support sets on which the primary fuzzy sets are defined should cover the associated universe of discourse in relation to some value, ε. This property is referred to as the "ε-completeness" by Lee (1990a). To ensure a dominant rule always exists, the recommendation is that the value of ε at the crossover point of two overlapping fuzzy sets is 0.5. At this value of ε, two dominant rules will be fired. To define the input fuzzy sets, error (E) and change in error (CE), the following procedure is adopted. In the case of the former fuzzy sets, the maximum range of error for a particular joint actuator is calculated. For example, a robot waist joint with a counter resolution of 0.025 degree per count, and a maximum allowable rotation of 300.0 degree would result in a maximum positional error of 12000 counts. A typical schematic representation for the error fuzzy set universe of discourse would be as illustrated in Fig. 2. The linguistic terms used to describe the fuzzy sets in Fig. 2 are:

{ *NB, NM, NS, ZE, PS, PM, PB* }

where N is negative, P is positive, B is big, M is medium, S is small and ZE is zero; a notation that is used throughout this chapter. Combinations of these letters are adopted to represent the fuzzy variables chosen, for example PositiveBig, PositiveMedium and PositiveSmall. As a result, 7 discretisation levels are initially defined for each input and output domain. The size and shape of the fuzzy sets displayed in Fig. 2 are chosen subjectively and tuned during process operation to obtain the most appropriate response. The proposed tuning methodology of these fuzzy sets is detailed later in Figure 4.2.
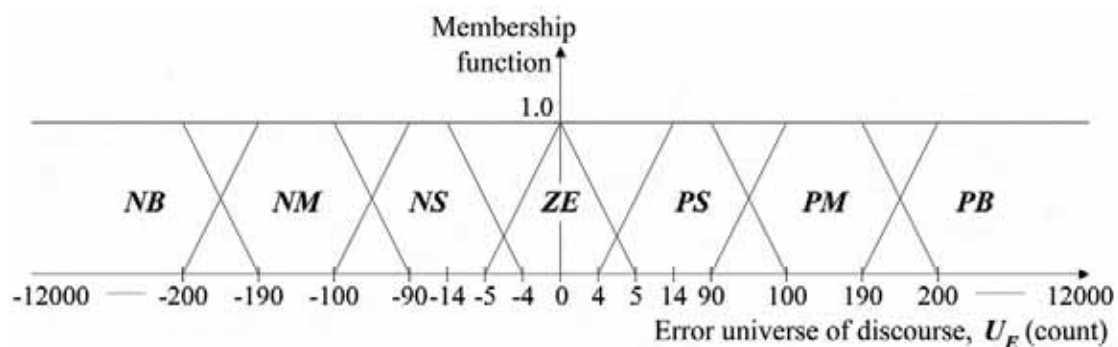
Figure 2. Universe of discourse: error fuzzy sets

To determine the domain size for the change in error variable in this project, an open loop test was conducted. In this test, a whole range of voltage (from the minimum to the maximum) was applied to each of the robot joint actuator and the respective change in angular motion error was recorded every sample interval. From this information, the fuzzy sets illustrated in Fig. 3 for the change in error were initially estimated. Although the open loop response of

the system will be different from the close loop response, it will give a good initial guide to the size of the domain appropriate for use with the fuzzy logic controller.
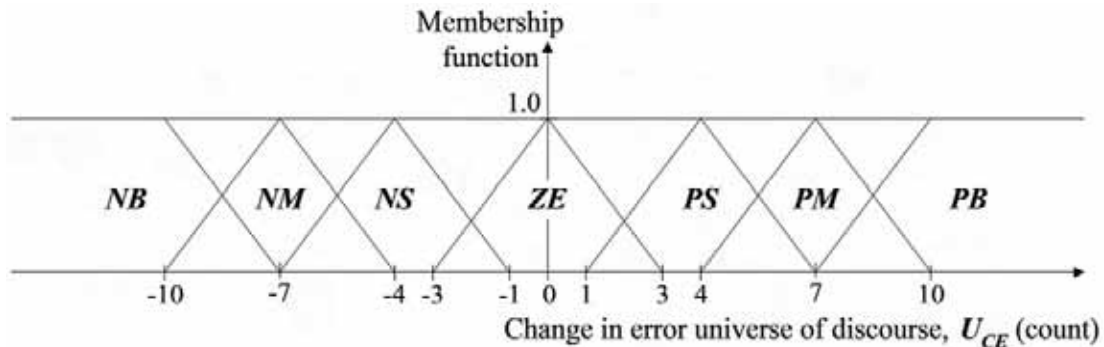
Figure 3. Change in error fuzzy sets domain of discourse

It should be noted that the choice of sampling interval is very important because it will affect the maximum change in error value recorded. It was found that the use of a very high sampling rate caused the recorded maximum change in angular motion error to be close to zero and this made it impossible to define the location of each fuzzy set in the domain of discourse. For example, a sampling period of 0.001 seconds will result in a maximum change in waist positional error of 2 counts; a value found experimentally. In a similar manner, the control variable output fuzzy sets were selected. However, in this particular case, the dimentionality of the space is determined by the resolution of the available D/A converters. The D/A converters adopted are of an 8-bit type which yield 256 resolution levels as indicated on the horizontal axis in Fig. 4(a). Again, the universe of discourse was partitioned into 7 fuzzy set zones as depicted in Fig. 4(b).
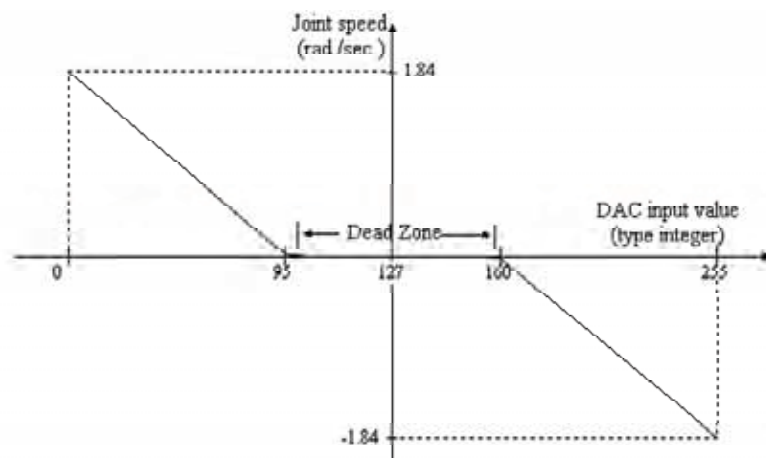
Figure 4(a). A typical characteristic for the waist joint actuator

It should be noted that the fuzzy set labelled Zero is defined across the dead zone of the dc-servo motor in order to compensate for the static characteristics of the motor in this region. The initial sizes and distribution of the fuzzy sets are tuned during operation to improve the closed loop performance of the system.
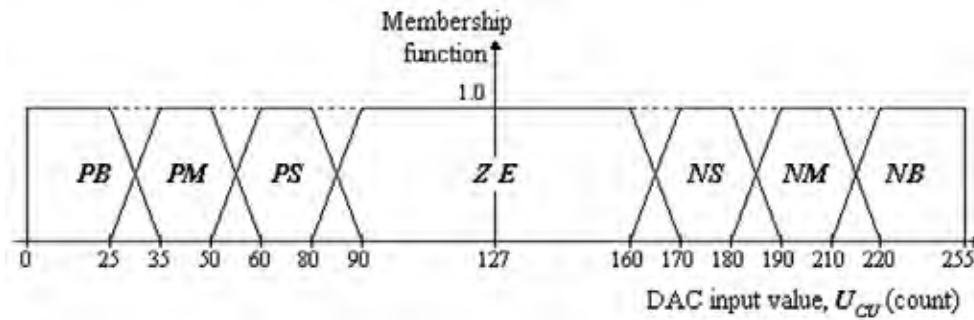
Figure 4(b). Control action domain of discourse

2.2.1.1 Transforming a Crisp Input to a Fuzzy Variable

Consider the trapezoidal representation of an error fuzzy set as illustrated in Fig. 5. Let an input error at sample interval k be $e(k) \in U_E$ and the corresponding membership grade of the fuzzy set $E_i \subset U_E$ be defined by the template [a, b, c, d]. Therefore, its membership ership function, μ_{E_i} can be directly evaluated using the expression:

$$
\left.
\begin{array}{l}
\textbf{IF } e(k) \notin (a, d) \textbf{ THEN } \mu_{E_i}(e(k)) = 0.0 \\[4pt]
\textbf{IF } e(k) \in [a, b) \textbf{ THEN } \mu_{E_i}(e(k)) = (e(k) - a)\, slope.ab \\[4pt]
\textbf{IF } e(k) \in [b, c] \textbf{ THEN } \mu_{E_i}(e(k)) = 1.0 \\[4pt]
\textbf{IF } e(k) \in (c, d] \textbf{ THEN } \mu_{E_i}(e(k)) = (d - e(k))\, slope.cd
\end{array}
\right\}
\tag{3}
$$

where the gradients *slope.ab* and *slope.cd* are calculated from the expressions;

$$
slope.ab = \frac{1.0}{b - a}
\tag{4}
$$

$$
slope.ab = \frac{1.0}{d - c}.
\tag{5}
$$

In a similar manner the properties of a triangular or bisected trapezoidal fuzzy set template can be defined.
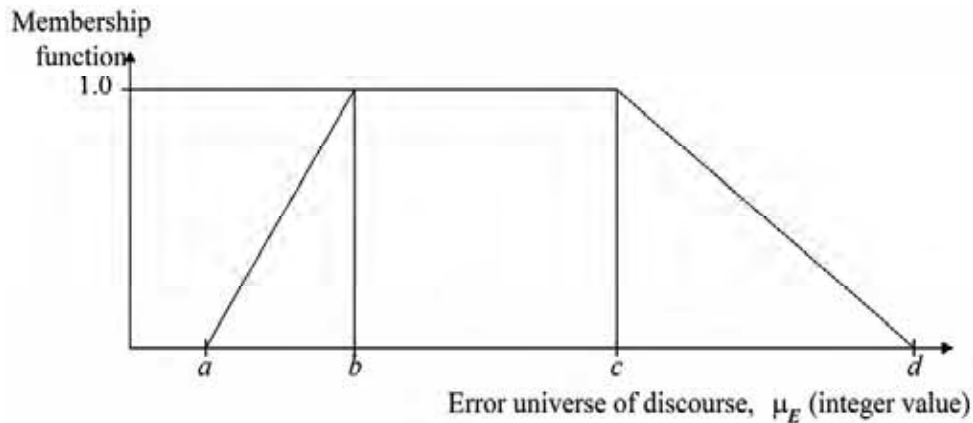
Figure 5. Trapezoidal representation of an error fuzzy set.

2.3 Defining the Fuzzy Rule Base

The fuzzy rule base employed in FLC contains fuzzy conditional statements which are currently chosen by the practitioner from a detailed knowledge of the operational characteristics of the process to be controlled. The fuzzy rule base can be derived by adopting a combination of four practical approaches which are mutually exclusive, but are the most likely to provide an effective rule base. These can be summarised as follows (Lee, 1990a): 1. Expert experience and control engineering knowledge. In nature, most human decision making are based on linguistic rather than numerical descriptions. From this point of view, fuzzy control rules provide a natural framework for the characterisation of human behaviour and decision making by the adoption of fuzzy conditional statements and the use of an inference mechanism. 2. Operational experience. The process performance that can be achieved by a human operator when controlling a complex process is remarkable because his reactions are mostly instinctive. An operator through the use of conscious or subconscious conditional statements derives an effective control strategy. These rules can be deduced from observations of the actions of the human controller in terms of the input and output operating data. 3. Fuzzy model of the process. The linguistic description of the dynamic characteristics of a controlled process may be viewed as a fuzzy model of a process. Based on this fuzzy model, a set of fuzzy control rules can be generated to attain an optimal performance from a dynamic system. 4. Learning. Emulation of human learning ability can be carried out through the automatic generation and modification of the fuzzy control rules from experience gained. The rule base strategy adopted in this work

is developed from operational and engineering knowledge. The initial control rule base adopted is displayed in the look-up table, Table 1. This table should be read as:

$$\textbf{IF} \ (\textit{error is } \textbf{NegativeMedium}) \ \textbf{AND} \ (\textit{change in error is } \textbf{Zero}) \ \textbf{THEN} \ (\textit{control action is } \textbf{PositiveBig}) \tag{6}$$

	U	CHANGE IN ERROR						
		NB	NM	NS	ZE	PS	PM	PB
ERROR	NB	PB						PM
	NM					PM		PS
	NS		PS					NM
	ZE	PM		ZE				
	PS						NS	
	PM	NS	NM			NB		
	PB	NM	NB					

Table 1. Initial rules selected for fuzzy logic controller

2.4 Fuzzy Inference Mechanism

One virtue of a fuzzy system is its inference mechanisms which is analogous to the human decision making process. The inference mechanism employs the fuzzy control rules to infer the fuzzy sets on the universe of possible control action. The mechanism acts as a rule processor and carries out the tasks of manoeuvring the primary fuzzy sets and their attendant operations, evaluating

the fuzzy conditional statements and searching for appropriate rules to form the output action. As mention earlier, the input and output variables of error, change in error and control action, U_E, U_{CE} and U_{cu}. respectively, are all chosen to be discrete and finite, and are in the form of;

$$E \subset U_E \,, \ CE \subset U_{CE} \ \text{ and } CU \subset U_{cu} \tag{7}$$

where indicates a fuzzy subset. As a result of selecting 7 discretisation levels for each fuzzy input and output variable, i.e. **PB**, **PM**, **PS**, etc., 49 fuzzy control rules result. These control rules are expressed in the form of fuzzy conditional statements;

$$\textbf{IF } (\textit{error is } \textbf{E}) \ \ \textbf{AND } (\textit{change in error is } \textbf{CE}) \\ \textbf{THEN } (\textit{control action is } \textbf{CU}) \tag{8}$$

At sample interval k, the jth fuzzy control rule, equation (8), can be expressed as;

$$\textbf{IF } (\textit{e(k) is } E_j) \ \ \textbf{AND } (\textit{ce(k) is } CE_j) \ \ \textbf{THEN } (\textit{cu(k) is } CU_j) \tag{9}$$

where $e(k)$, $ce(k)$ and $cu(k)$ denote the error, change in error and manipulated control variable respectively. The jth fuzzy subsets E_j, CE_j and CU_j are defined as;

$$E_j = \{ (e \,, \mu_E(e)) \}, \ CE_j = \{ (ce \,, \mu_{CE}(ce)) \}, \ CU_j = \{ (cu \,, \mu_{cu}(cu)) \} \tag{10}$$

Alternatively, Equation (9) can be evaluated through the use of the compositional rule of inference. If the minimum operator is utilised, the resulting membership function can be expressed as;

$$\mu_{\mathfrak{R}_j}(e(k), \, ce(k), \, cu(k)) \ = \ \left[\mu_{E_j}(e(k)) \ \textbf{AND} \ \mu_{CE_j}(ce(k)) \right] \ \Rightarrow \ \mu_{CU_j}(cu(k))$$
$$= \ min \left[\mu_{E_j}(e(k)) \,, \ \mu_{CE_j}(ce(k)), \ \mu_{CU_j}(cu(k)) \right] \tag{11}$$

where the symbol $\square$ indicates the fuzzy implication function and $\mathfrak{R}_j = E_j \times CE_j \times CU_j$ denotes the fuzzy relation matrix on

$$\mathfrak{R} \ = \ \bigcup_{j=1}^{49} \mathfrak{R}_j \ = \ \max_{j=1}^{49} \mathfrak{R}_j \tag{12}$$

In term of the membership functions, this can be expressed as;

$$\mu_{\mathfrak{R}_j}(e, \, ce, \, cu) \ = \ \max_{j=1}^{49} \left\{ min \left[\mu_{E_j}(e), \, \mu_{CE_j}(ce) \,, \, \mu_{CU_j}(cu) \right] \right\} \tag{13}$$

To use the result of Equation (13), a defuzzification process is necessary to produce a crisp output for the control action value.

2.5 Choosing Appropriate Defuzzification Method

Several approaches (Lee, 1990b) have been proposed to map the fuzzy control action to a crisp value for input to the process. Basically, all have the same aim that is, how best to represent the distribution of an inferred fuzzy control action as a crisp value. The defuzzification strategies most frequently found in the literature are the maximum method and centre of area method:

1. The maximum method. Generally, the maximum method relies on finding the domain value, z_0, that maximises the membership grade which can be represented by;

$$z_o = \max_{z \in W} \mu_{C_T}(z) \tag{14}$$

In the case when there is more than one maximum membership grade in W, the value of z_0 is determined by averaging all local maxima in W. This approach known as mean of maximum method (MOM) is expressed as;

$$z_o = \frac{1}{n} \sum_{i=1}^{n} z_i \tag{15}$$

where $z_i = \max_{z \in w} \mu_{C_T}(z) \; z_i = \max_{z \in w} \mu_{C_T}(z)$ and n is the number of times the membership function reaches the maximum support value.

2. The center of area method (COA). The center of area method sometimes called the centroid method produces the center of gravity of the possibility distribution of a control action. This technique finds the balance point in the output domain of the universe of discourse. In the case when a discrete universe of discourse with m quantisation levels in the output, the COA method produces;

$$z_o = \frac{\sum_{i=1}^{m} \mu_{C_T}(z_i) \, z_i}{\sum_{i=1}^{m} \mu_{C_T}(z_i)} \tag{16}$$

where zi is the ith domain value with membership grade of $\mu (zi)$.

3. Experimental Setup

The robot control system is composed of the host computer, the transputer network, and the interface system to a small industrial robot. The schematic representation of the control structure is presented in Fig. 6.
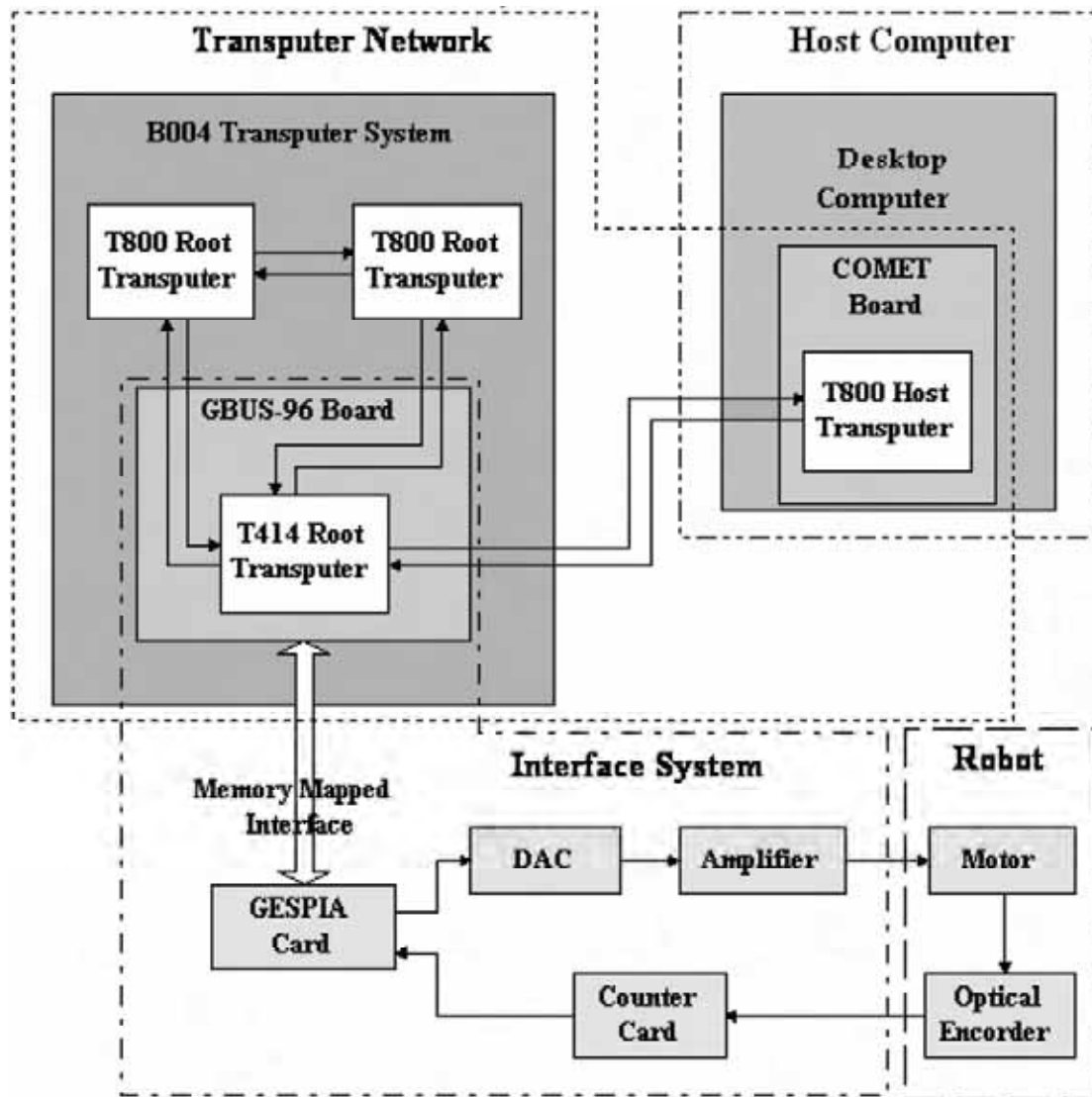
Figure 6. Schematic representation of robot control architecture.

The controller structure is hierarchically arranged. At the top level of the system hierarchy is a desktop computer which has a supervisory role for supporting the transputer network and providing the necessary user interface and disc storage facilities. The Transputer Development System acts as an operating system with Occam II as the programming language. At the lower level are the INMOS transputers; in this application one T800 host transputer is resident on the COMET board and mounted in one of the expansion slots of the desktop

computer with the remaining three transputers resident in a SENSION B004 system. The robot is the RM-501 Mitsubishi Move Master II, with the proprietary control unit removed to allow direct access to the joint actuators, optical encoders and joint boundary detection switches. The host transputer also provides an interface facilities to the user, for example, input and output operation from the keyboard to the screen. The three transputers resident in the SENSION B004 system are a T414 transputer which is resident on the GBUS-96 board and provides a memory mapped interface to the robot through a Peripheral Interface Adapter (PIA) Card. The remaining two T800 root transputers are used to execute the controller code to the robot. The PIA card allows a parallel input and output interface to the robot joint actuators and conforms to the interface protocol implemented on the GBUS-96 which is known as a GESPIA Card. The actual hardware arrangement together with the interfacing employed is shown in Fig. 7, with the Mitsubishi RM-501 robot shown in Fig. 8.
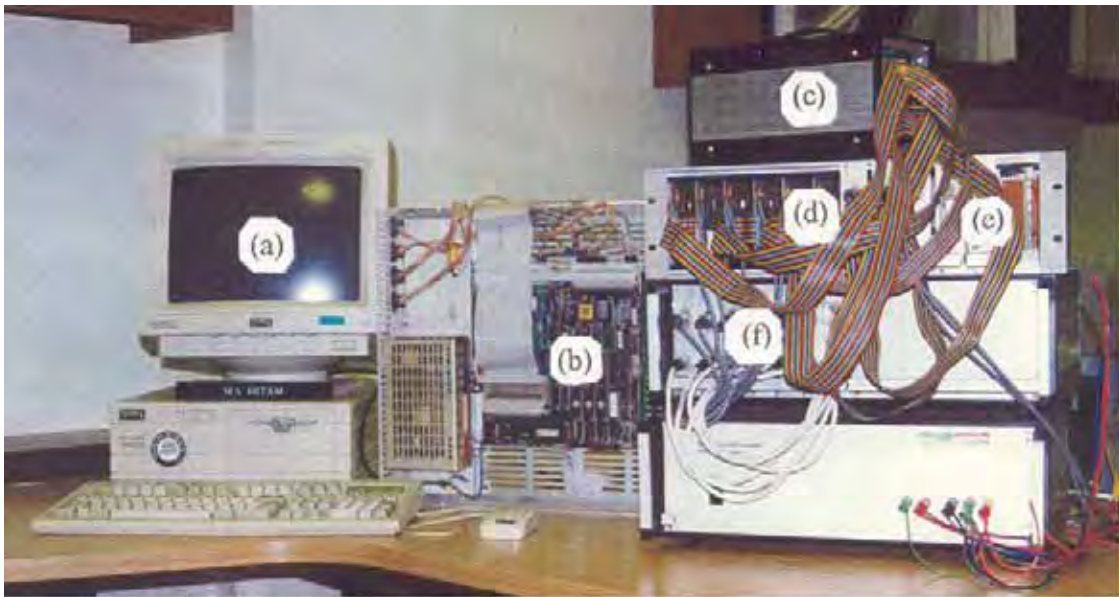
Figure 7. System hardware and interfacing. (a) host computer, (b) B004 transputer system, (c) GESPIA card, (d) DAC cards, (e) counter cards and (f) power amplifier.
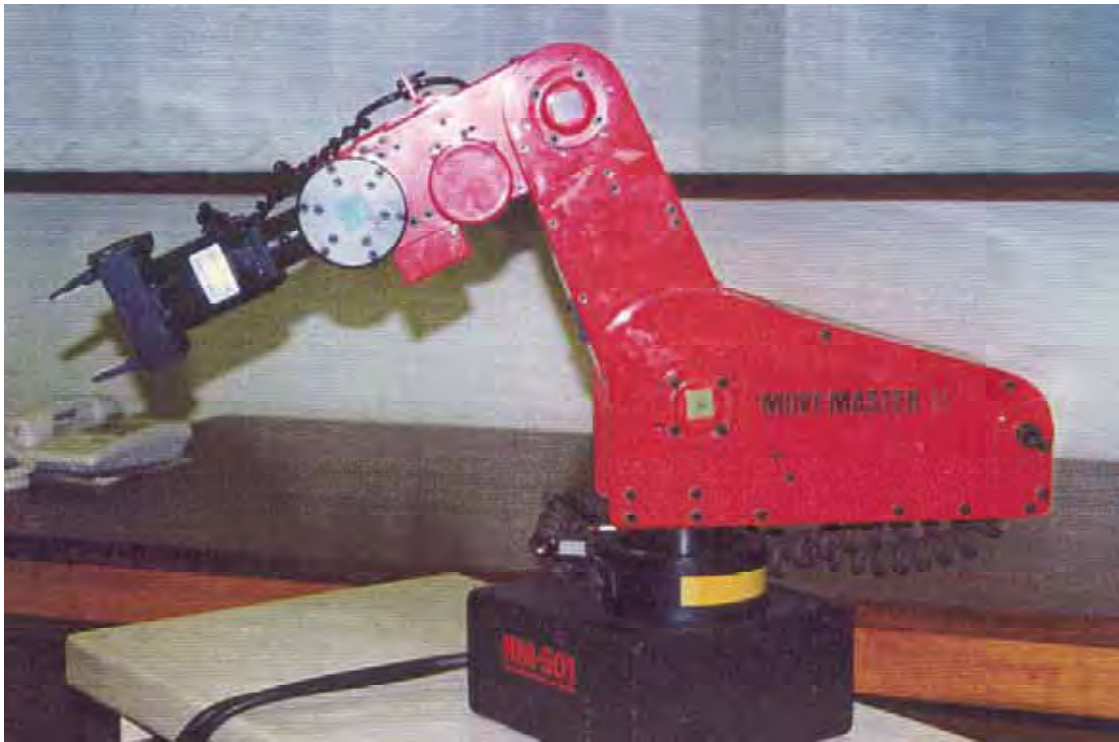
Figure 8. The Mitsubishi RM-501 Move Master II Industrial Robot.

3.1 The Mitsubishi RM-501 Move Master II Robot

This industrial robot is a five degree of freedom robot with a vertical multi-joint configuration. The robot actuators are all direct current servo motors, but of different powers. At the end of each joint, a sensor is provided to limit the angular movement. The length of link and its associated maximum angular motion is listed in Table 2. Fig. 9(a) and 9(b) illustrate the details of the robot dimensions1 and it's working envelop. The maximum permissible handling weight capacity is 1.2 kg including the weight of the end effector. Table 2 The Mitsubishi RM-501 Move Master II geometry.

Join	Link Length (mm)	Maximum Rotation (Degree)
Waist	250	300
Shoulder	220	130
Elbow	160	90
Wrist roll	65	+-90
Wrist pitch	65	+-180

Table 2. The Mitsubishi RM-501 Move Master II geometry
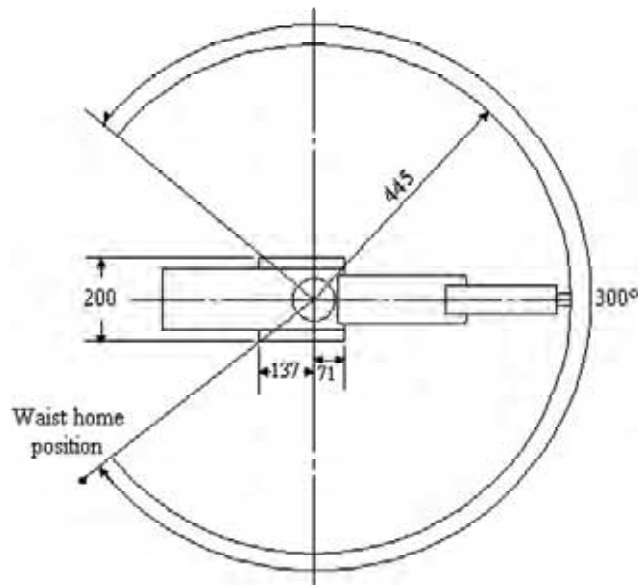
Figure 9(a). Range of movement of waist joint and robot dimensions (all dimensions are measured in millimeter).
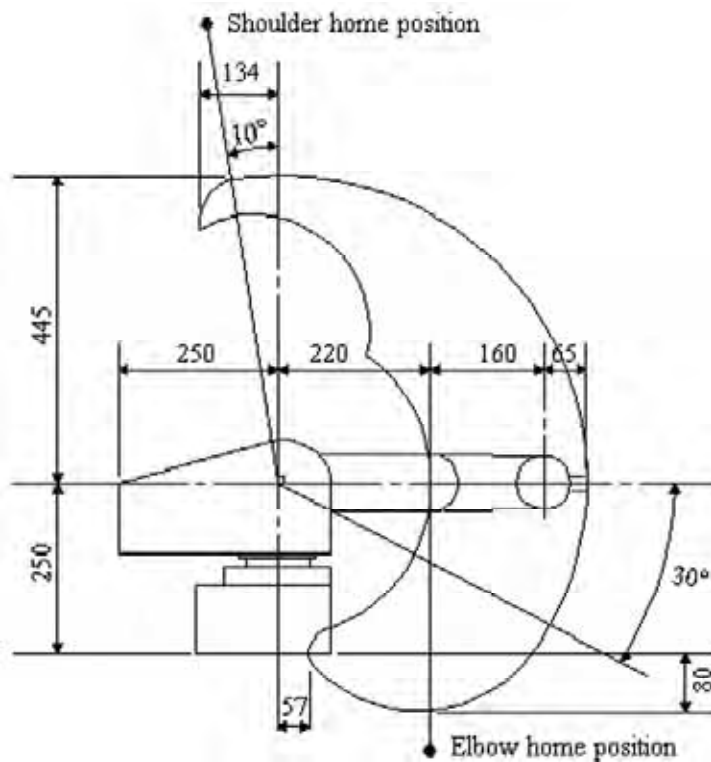
Figure 9(b). Robot dimension and range of movement when hand is not attached.

4. Experimental Studies

A program code for the development of a FLC was written in the Occam language and executed in a transputer environment. This approach would enable the evaluation of the robustness of the controller design proposed and applied to the first three joint of a RM-501 Mitsubishi industrial robot. A T800 transputer is assigned to position each joint of the robot independently. To determine the effect on controller performance of changing different controller parameters, one joint only is actuated and the other two are locked. In the first experiment the impact on overall robot performance of changes in sample interval was assessed. This was followed by an investigation into how best to tune a controller algorithm and whether guide-lines can be identified for future use. The problem is to overcome the effect of changing robot arm configuration together with a varying payload condition.

4.1 The Choice of Sampling Time

Inputs (error and change in error) to the fuzzy logic control algorithm that have zero membership grades will cause the membership grades of the output fuzzy sets to be zero. For each sample period, the on-line evaluation of the algorithm with 49 control rules has been found by experiment to be 0.4 milliseconds or less. Hence, to shorten the run time, only inputs with non-zero membership grades are evaluated. For times of this magnitude, real-time control is possible for the three major joint controllers proposed. It has been cited in the literature that it is appropriate to use a 0.016 seconds sampling period (60 Hertz) because of its general availability and because the mechanical resonant frequency of most manipulators is around 5 to 10 Hz (Fu et al., 1987). Experiments have been carried out to determine how much improvement can be achieved by shorten the sampling period from 0.02 seconds to 0.01 seconds. In the first experiment, the waist joint is subjected to a 60.0 degree (1.047 radian or 2400 counter count) step disturbance with all other joints in a temporary state of rest. The results shown in Fig. 10 suggest that very little improvement in transient behaviour will be achieved by employing the shorter sampling period. The only benefit gained is a reduction in the time to reach the steady state of 0.4 seconds. In a second test, the waist joint is commanded to start from its zero position and to reach a position of 5 degree (0.0087 radian or 20 counter count) in 2 seconds; it remains at this position for an interval of 1 second after which it is required to return to its home position in 2 seconds as showed in Fig. 11. Again the benefit is only very marginal and of no significance for most industrial applications. Despite these results, it was decided that the higher of the two sampling rates would generally ensure better transient behaviour, hence the 0.01 seconds sampling period is used throughout this project.
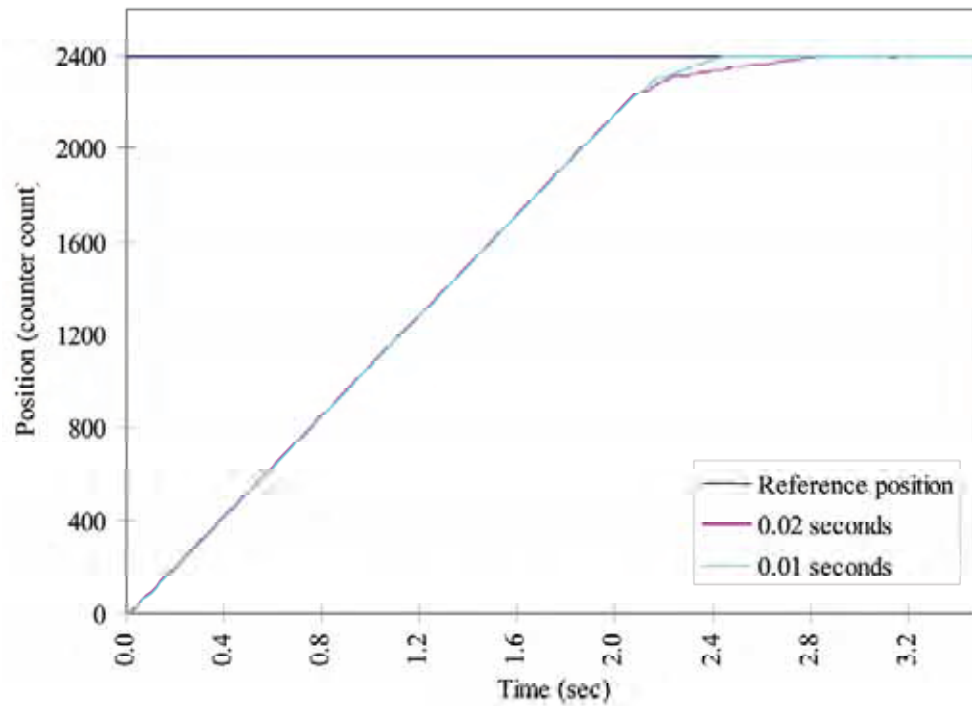
Figure 10. Waist response to a step input for different sampling periods.
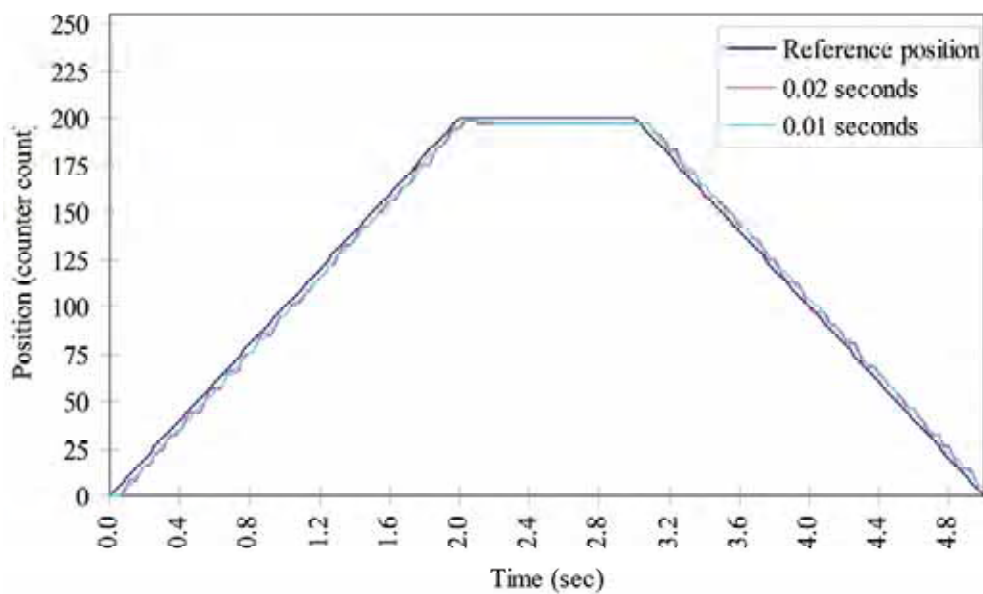
Figure 11. Waist trajectory tracking at different sampling periods.

4.2 Controller Tuning Strategies

Tuning of a FLC may be carried out in a number of ways; for instance modifying the control rules, adjusting the support of the fuzzy sets defining magnitude and changing the quantisation levels. The objective is always to minimise the difference between the desired and the actual response of the system. Because of the large number of possible combinations available and the different operational specifications that exist, no formal procedure exists for tuning the parameters of a FLC. In practice, trial and observation is the procedure most commonly employed. This can be tedious and time consuming and may not result in the selection of the most suitable parameters and in many practical situations does require safeguards to prevent excessive output variations and subsequent plant damage. To establish a rule base for a FLC, it is necessary to select an initial set of rules either intuitively or by the combination of methods described in Figure 2.3 for the process to be controlled. Rule modifications can be studied by monitoring the response of the close loop system to an aperiodic disturbance in a phase plane for error and change in error in this case. This trial and observation procedure is then repeated as often as required until an acceptable response is produced. In this project, three different ways for tuning the controller have been investigated. The initial control rules were initially selected by modifying the rule based initiated by Daley (1984). The modifications made to the rule base given in Daley (1984) were necessary to ensure a faster transient response with minimum overshoot and steady state error for the robot arm. The rule base listed in Table 1 was found to be the most appropriate for the robotic control process. A second procedure for tuning a FLC is investigate in this section and the final method will be presented in the next section. The use of two inputs and one output means that there is a three dimensional space in which to select the optimal solution. In most reported cases, scaling factors or gains were introduced to quantified these three universes of discourse. Hence, making it possible to tune a controller by varying the values of these gain terms. However, in this project, the functional forms of the fuzzy sets were utilised and these were mapped directly to the corresponding universe of discourse. Thereby, tuning is carried out by adjusting or redefining each fuzzy set location in the universe of discourse. The strategy developed in this project is to show the effect of changing each of the input and an output definitions to establish the impact on the overall performance of the robot. The initial estimate for the fuzzy sets employed in the three domains of discourse were made off-line as detailed in Figure 2.2.1. Fig. 12 shows the estimates for the error fuzzy set definitions. The corresponding estimates for the change in error and controlled action fuzzy set definitions are plotted in Fig. 13 and Fig. 14, respectively. Tuning of the error fuzzy sets is made by gradually moving the fuzzy set locations in the universe of discourse closer to the zero value of error. A similar procedure is adopted to tune the output fuzzy sets,

however, the initial selection positioned these fuzzy sets as close as possible to the equilibrium point. For these sets, tuning is executed by gradually moving the fuzzy sets away from the equilibrium point until an acceptable close loop response is found. To demonstrate the effect of changing the error fuzzy sets definition, three choices of error fuzzy sets definition were made and are plotted in Fig. 12(a) - 12(c) named Case 1, 2 and 3. The other two fuzzy set definitions remained unchanged and are shown in Fig. 13 and 14. A step disturbance of 30.0 degree (1200 counter counts or 0.523 radian) was introduced at the waist joint and the other joints, shoulder and elbow, were locked at 10 degrees and 90 degrees respectively2. This robot arm configuration was chosen to exclude variations in inertia, gravitational force and cross coupling resulting from the movement of these two joints. The impact from the use of an inappropriate definition for the output fuzzy sets is significant when variations in these forces are present and this is studied in latter experiments. The joint response to a 30.0 degree step disturbance is shown in Fig. 15(a) and the associated control signals are shown in Fig. 15(b). Notice that in Figure 15(a), at about 1.0 second the joint response for error fuzzy set definition Case 1 started to deviate from the other two cases because of the large spread in the fuzzy sets in comparison to the other two cases depicted in Fig. 12(a). There is no noticeable difference in transient response between Cases 2 and 3 because the fuzzy set definitions are close to the equilibrium point. However, differences start to emerge as the two responses approach the chosen set point, Case 3 response is more oscillatory. This corresponds to the excessive voltage variations that can be observed in Fig. 15(b). This phenomenon occurred because of the use of a very tight definition for error (close to zero) in Case 3 (Fig. 12(c)) which leads to a large overshoot and hunting around the set point. The use of the fuzzy set definition, Case 2 gives a good accuracy throughout the whole envelope of operation for the waist joint with this particular robot arm configuration. The maximum steady state error is found to be 0.0013 radian (3 counter count) as a result of the coarseness of the fuzzy sets used; fuzzy set labelled **ZERO** is defined between -5 to +5 counts.
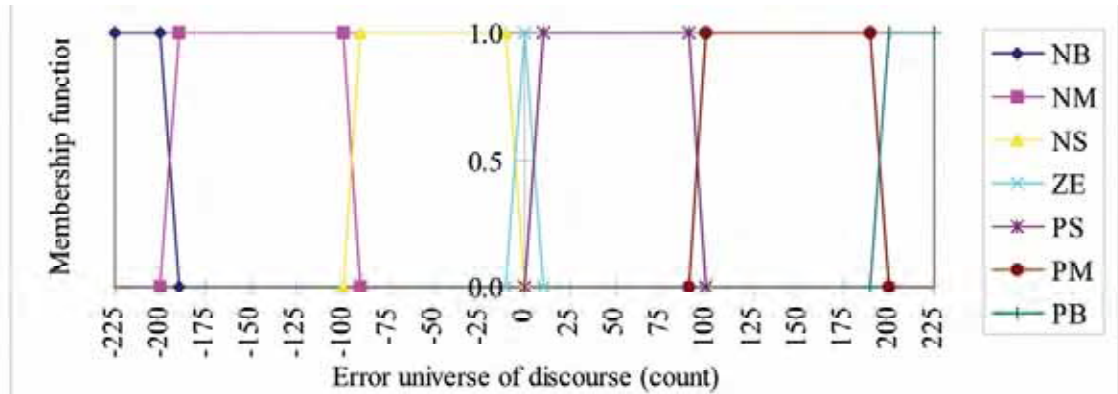
Figure 12(a). Case 1; error fuzzy set definition.
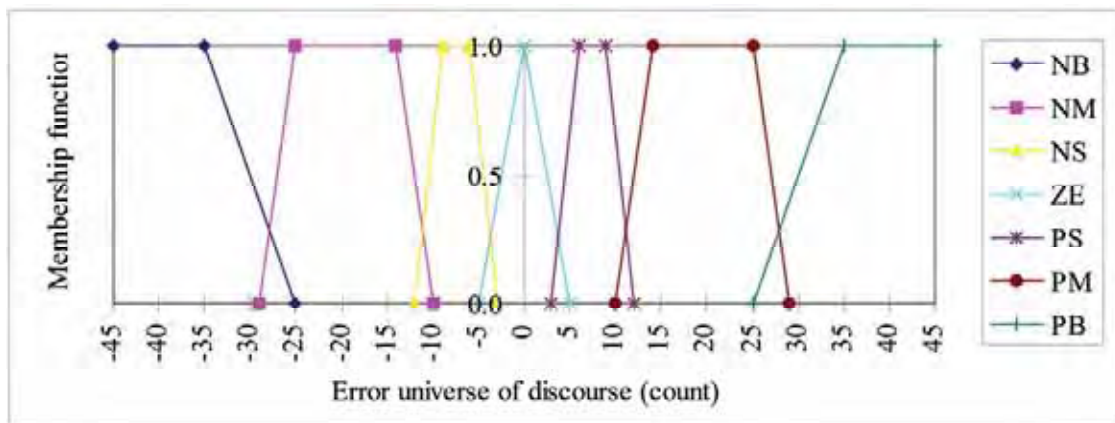
Figure 12(b). Case 2; error fuzzy set definition.
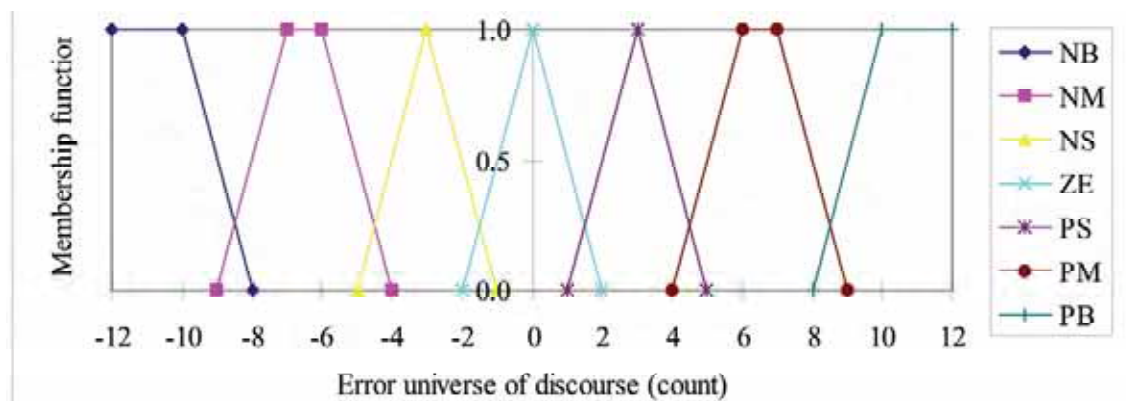
Figure 12(c). Case 3; error fuzzy set definition.
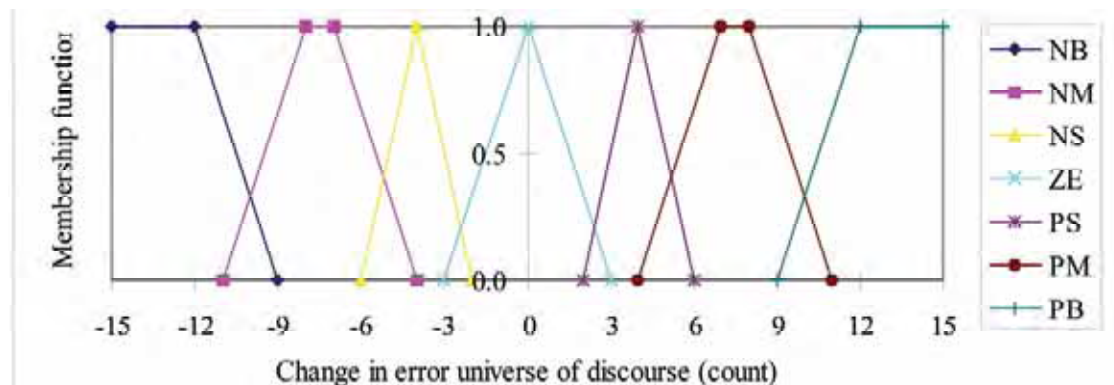
Figure 13. Change in error fuzzy set definition.
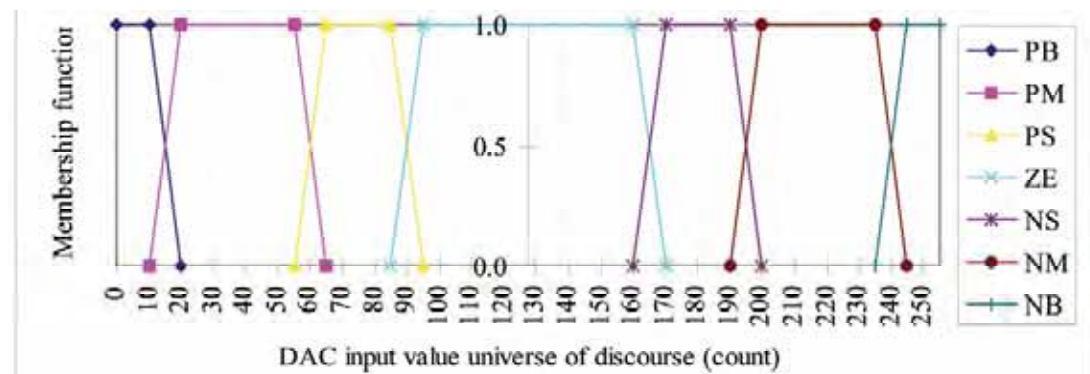
Figure 14. Control action fuzzy set definition.

This performance, however, is degraded when the robot arm configuration is changed. For example, when the shoulder and elbow joints are at full stretch and locked at 100.0 and 90.0 degree respectively, large overshoot cannot be avoided due to the increased inertia on the waist joint. Fig. 16 is provided as an illustration of the waist response to a small step disturbance of 1.0 degree (40 counter count or 0.017 radian). The blue line included in this figure is a waist response for the re-tune fuzzy sets definition which will be discussed later in this section. It can be seen that that despite exhaustive tuning of both the input fuzzy sets a large overshooting cannot be avoided for the second robot arm configuration. From these results, it can be concluded that to provide a better control a new combination of fuzzy sets have to be defined. A way of achieving this is to reduce the waist speed of operation. By redefining a smaller error, change in error and control action fuzzy sets in this region of operation, finer control around the equilibrium can be achieved but of a reduced operational speed. The smaller tuned fuzzy set combinations are plotted in Fig. 17, 18 and 19. The blue line in Fig. 16 shows the waist response and the corresponding controlled input signals for the smaller tuned fuzzy set combinations for com-

parison with the Case 2 combination previously defined.
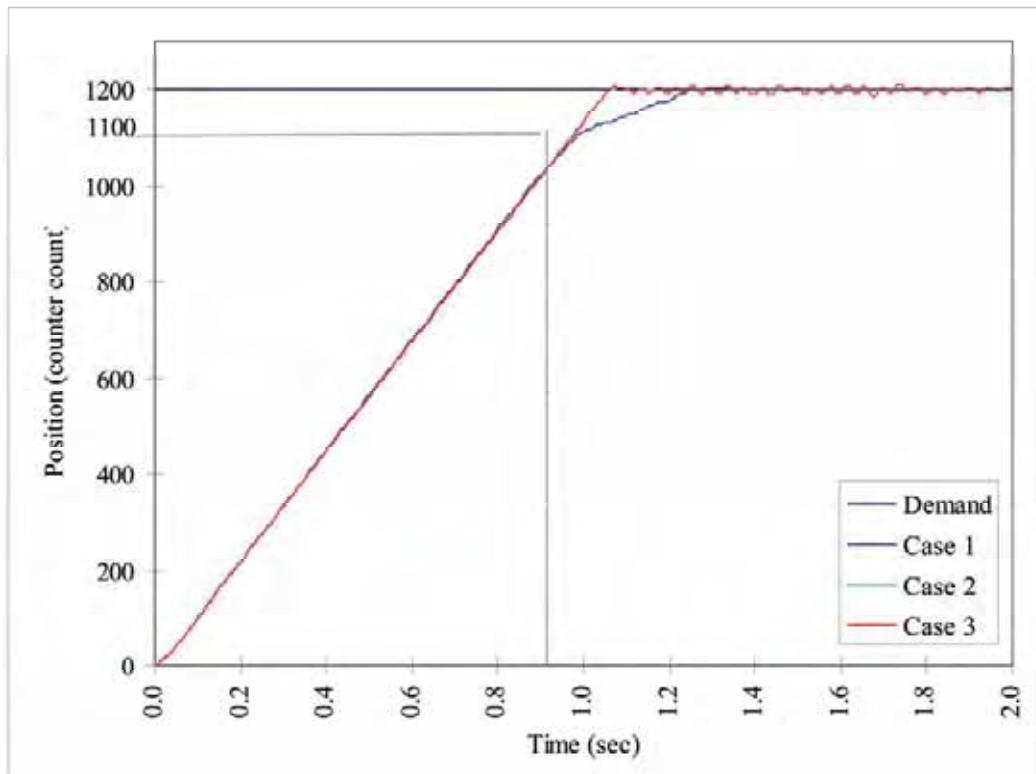
Figure 15(a). Waist response for different error fuzzy set definitions.
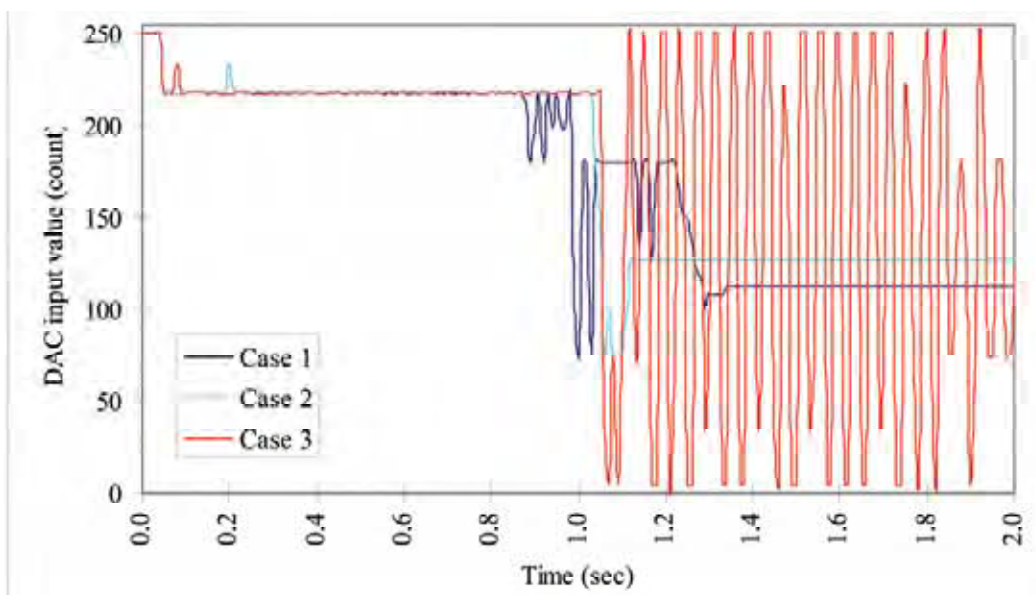
Figure 15(b). Control signals for different error fuzzy set definitions.
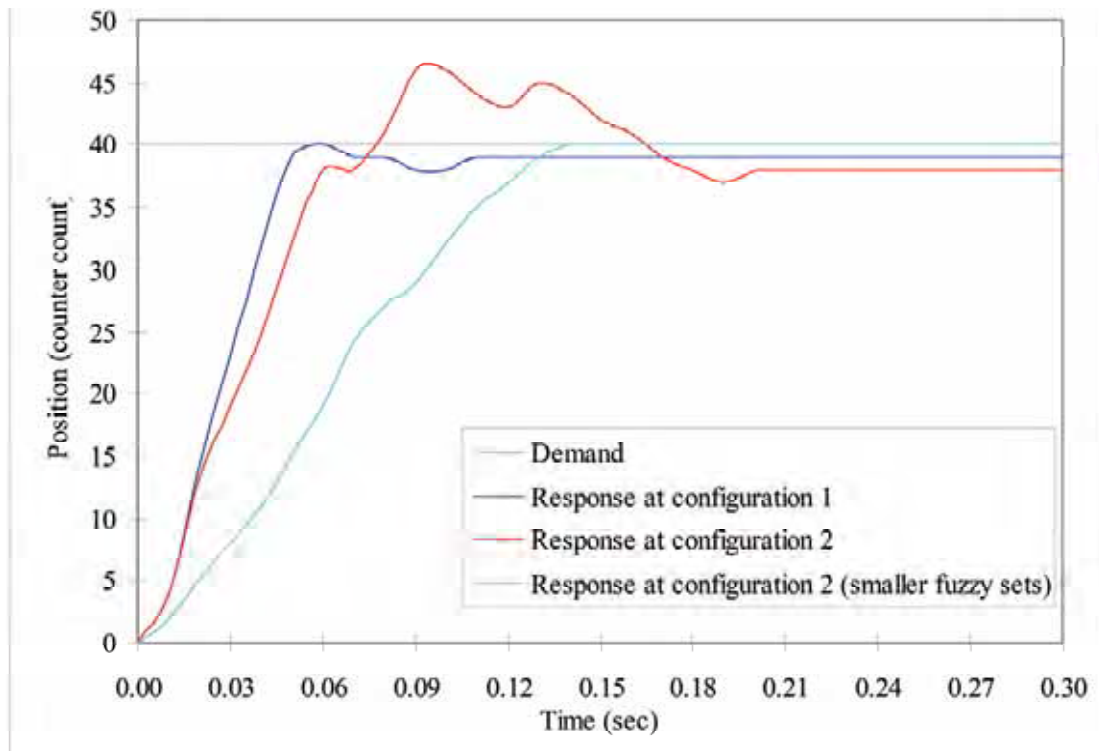
Figure 16. Waist response at different arm configurations.

The drawback of using the smaller fuzzy set combinations is obvious when the waist is subjected to a large step disturbance, for example at 30.0 degrees (1200 counter count or 0.523 radian) with the arm in the second configuration. The subsequent waist response is plotted in Fig. 20 together with the response for the second definition of the fuzzy set combinations. The waist response when using the smaller fuzzy set combinations of Fig. 16 and Fig. 20 show that the controller could position the waist with a smaller overshoot of 0.025 degree (1 counter count) and zero steady state error, however, the penalty to be paid is an increase in rise time. The zero steady state error is achieved because of the use of a fuzzy singleton definition in the error fuzzy set labelled ZERO, i.e. the error fuzzy set is defined between ± 1 in the universe of discourse as is de-picted in Fig. 17. Although the Case 2 fuzzy set combinations can provide a faster response (about 1.2 seconds quicker for a 30.0 degree step input), the overshoot (0.25 degree) and steady state error (0.075 degree) are both greater, Fig. 20. These results deteriorate when the controller operates under gravita-tional force and variable payload. A further comparison to evaluate the per-formance between the smaller fuzzy set combinations and the Case 2 fuzzy set combinations is conducted by subjecting the waist to a sinusoidal signal dis-turbance of 30.0 degree amplitude and 4.0 seconds time periods. Fig. 21(a) shows clearly that trajectory following with the Case 2 fuzzy set combinations

is by far the better result. Fig. 21(b) illustrates that the use of a smaller range of voltage in the output fuzzy sets definition could not generate an adequate controlled signal. These results suggest that tuning can never optimise simultaneously speed of response and accuracy. If fuzzy logic is to be used successfully in industrial process control, a method which can provides a fast transient response with minimum overshoot and steady state error must be found. One way to achieve this is to partition the problem into coarse and fine control, an approach suggested by Li and Liu (1989).
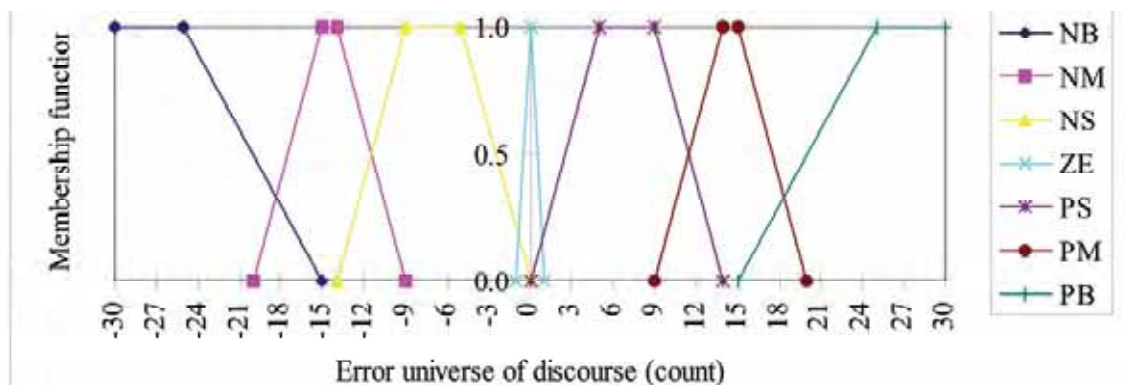
Figure 17. Smaller error fuzzy set definition.
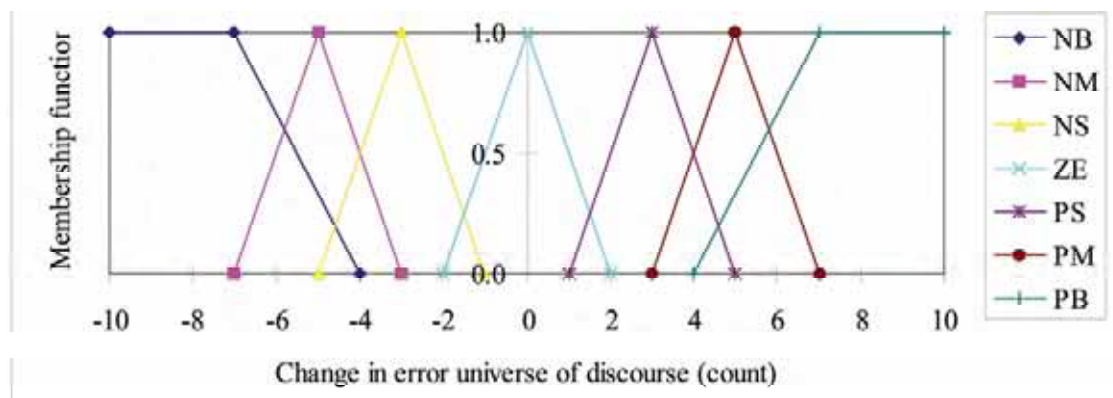
Figure 18. Smaller change in error fuzzy set definition.
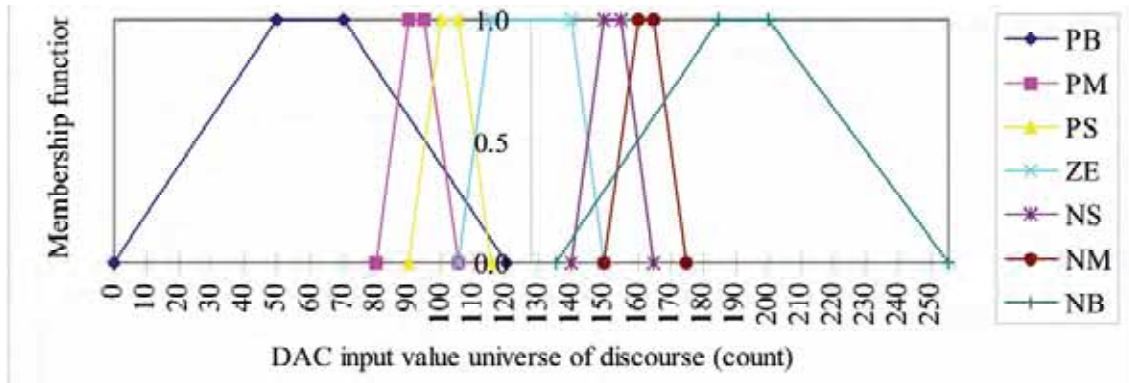
Figure 19. Smaller control action definition of fuzzy sets.
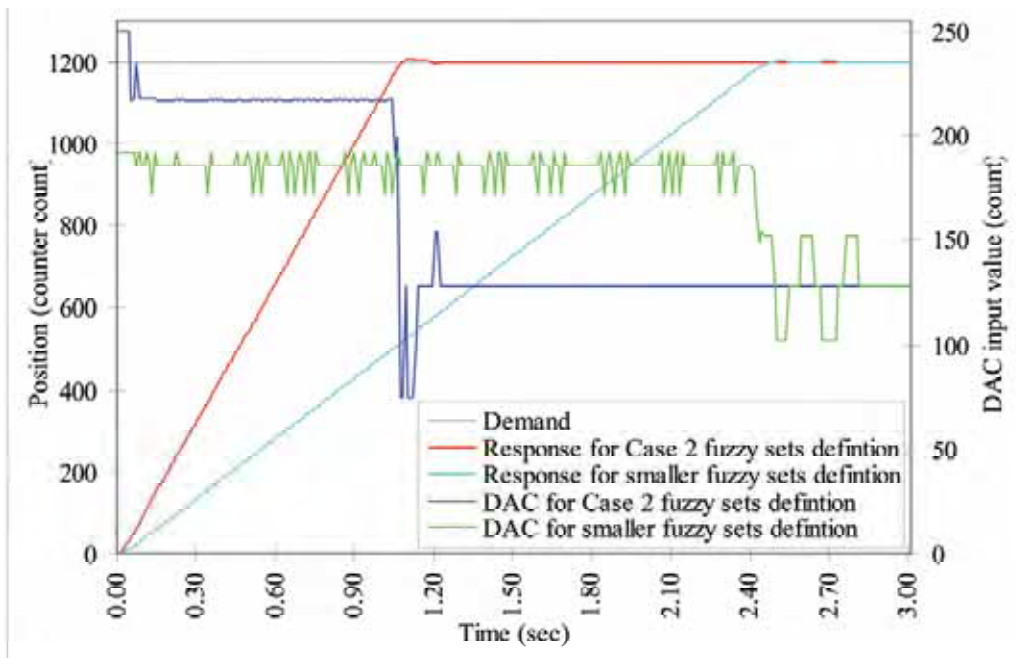
Figure 20. Waist response for different tuned fuzzy sets at second robot arm configuration.
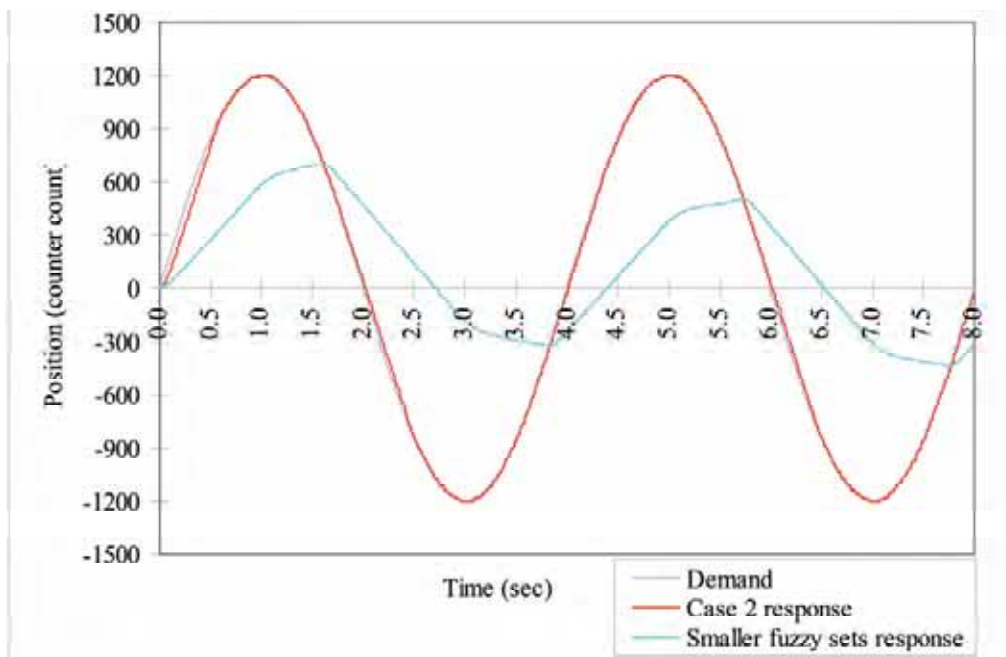
Figure 21(a). Response of the smaller and Case 2 fuzzy set combinations to sinusoidal tracking.
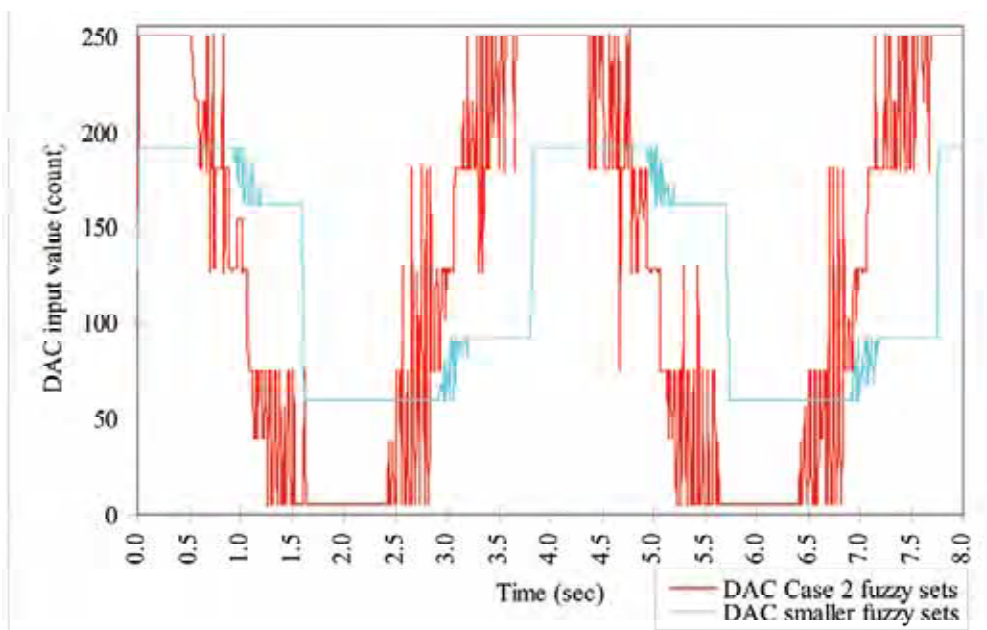
Figure 21(b). Control signal in Fig. 21(a).

Having investigated the problems associated with the control of the waist joint, the investigation was extended to the more difficult upper-arm link, the

shoulder joint. The control of this joint is difficult because of the gravitational force acting on it. For example, when the elbow is fully stretched and the shoulder is at 30.0 degree location to the working envelop, a load of 0.4 kg. is enough to drag the shoulder joint downwards with 0 voltage (127 DAC input value) applied to the actuator. The use of a single output fuzzy set definition was found only suitable for a limited range of operation and not applicable for the robot employed in this study. To illustrate this limitation, Fig. 22 shows the effect of using a single output definition in 4 different operational regions when the elbow is fully stretched. To compensate for the gravitational loading effect, 4 operational regions were identified, and each was assigned a different output fuzzy set. The switches 1, 2, 3 and 4 would control the choice of the output fuzzy set in the range of 0 to 10 degrees, 10 to 30 degrees, 30 to 90 degrees and 90 to 130 degrees of the shoulder joint working envelop, respectively. The four switched output fuzzy sets are presented in Fig. 23(a) - 23(d) and these have been tuned as previously discussed. In all four modes of operation, the input fuzzy set combinations of Case 2 were utilised. From Fig. 23(a) - 23(d), it is obvious that the fuzzy sets labelled ZERO is moving towards the right of the plot from the left as the region of operation moves from 1 to 4. This is to compensate for the gravitational load which forces the joint to overshoot when moving downwards as can be seen in Fig. 20. It should be noted that the use of the switches in selecting the output fuzzy set definition is just a coarse estimate, and as a result can give up to a maximum steady state error of 0.125 degrees (5 counter count) for the shoulder joint working envelop. If more accurate positional control is needed, it will be necessary to increase the number of switching regions or alternatively a different method will have to be found.
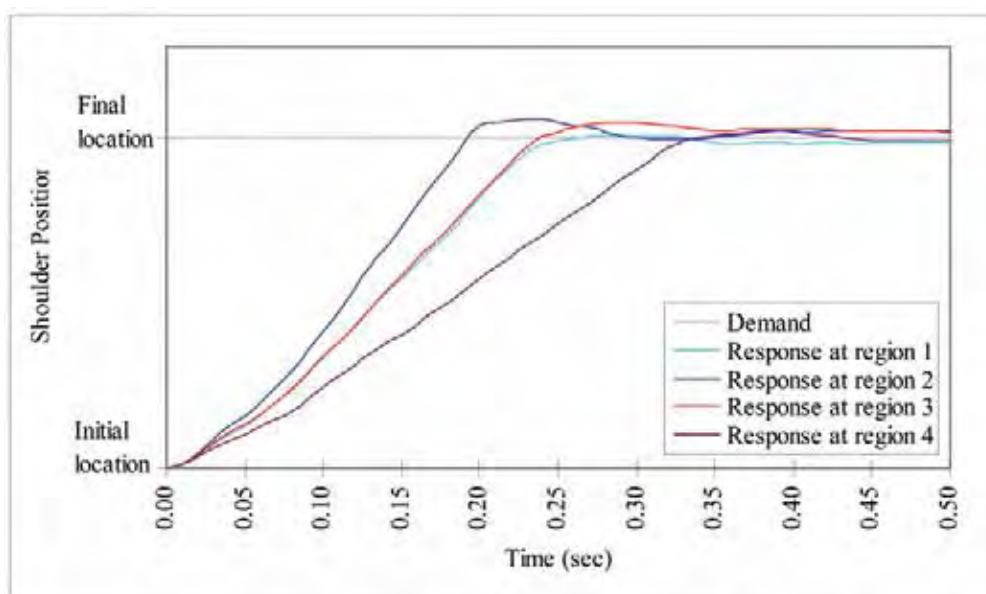
Figure 22. Shoulder response in different regions of operation.
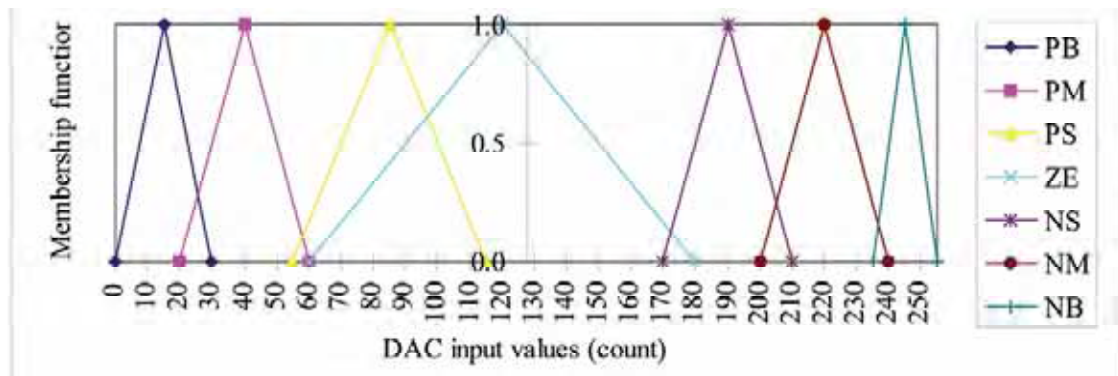
Figure 23(a). Switch 1; control action fuzzy set definitions for shoulder.
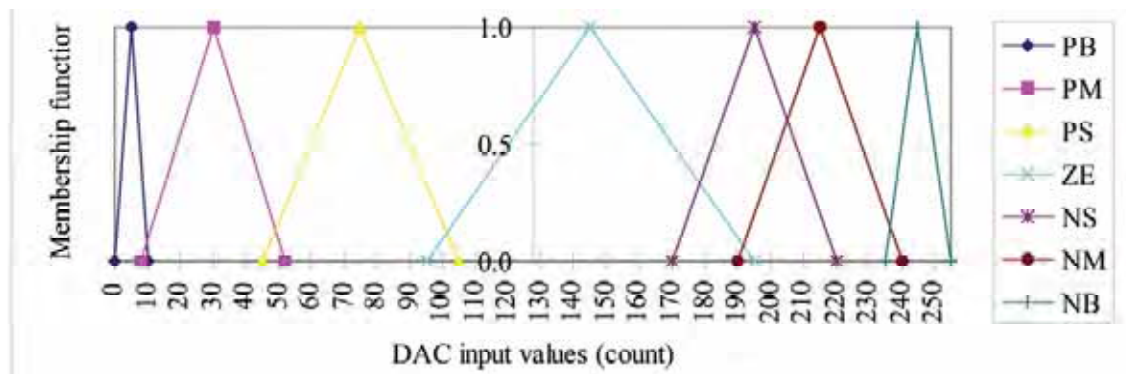
Figure 23(b). Switch 2; control action fuzzy set definitions for shoulder.
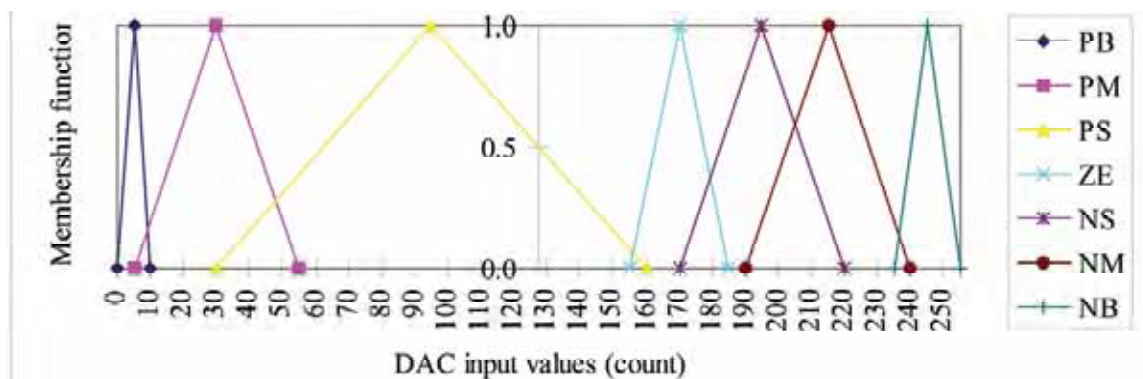
Figure 23(c). Switch 3; control action fuzzy set definitions for shoulder.
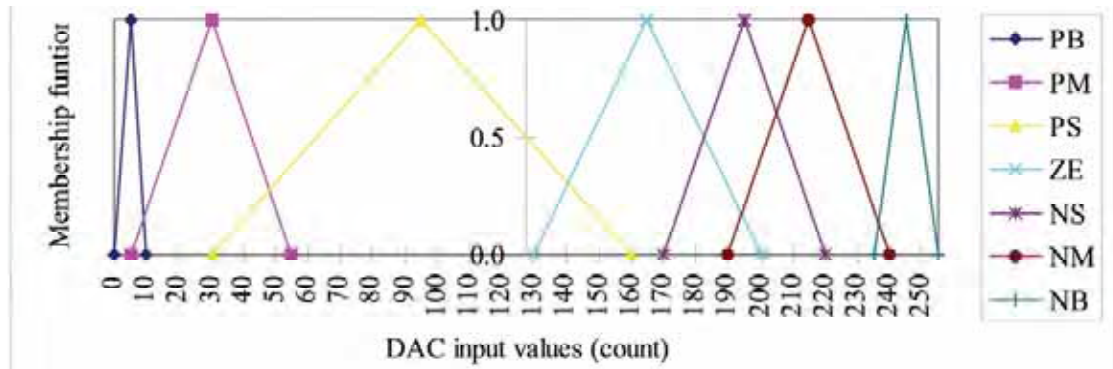
Figure 23(d). Switch 4; control action fuzzy set definitions for shoulder.

It should also be mentioned that the use of a trapezoidal function to represent the dead-zone area, mention in Section 2.2.1, is not suitable for implementation at this joint because of the unsymetrical nature of the actuator dead-zone in different regions of operation. Therefore, as an alternative the triangular function was used because it provides a more operational acceptable definition for the fuzzy sets. From the experiments gained in this section, it can be concluded that by using a trial and observation procedure, tuning of the FLC parameters can be successfully accomplished. To reduce the design time consumed adopting the trial and observation tuning method, a good initial estimate for the fuzzy set definitions is essential.

5. Conclusion

In this chapter, a methodology for the application of fuzzy logic theory to the development of a fuzzy logic controller had been presented and successfully implemented. The developed algorithm had been shown to be of simple design and implementable for real-time operation of a three joint industrial robot using joint space variables for control. The methodology to estimate the initial fuzzy sets has been presented. The use of the function form of template to represent the fuzzy sets provides a way to directly map these fuzzy sets into the corresponding universe of discourse. Unfortunately, this design could only be arrived at by the use of a trial and observation procedure and would suggest a more formal procedure must be developed for industrial applications. Furthermore, design by a trail and observation procedure cannot be guaranteed to yield the best result. In conclusion, it had been shown that a FLC can be designed to match specific process dynamics without the use of a process model within the control loop. Therefore, if automatic tuning can be introduced into the FLC design a very robust control approach will result and this could be directly applied to any poorly defined non-linear process.

6. References

Lee, C.S.G., chung, M.J., Turney, J.L. & Mudge, T.N. (1982). On the control of mechanical manipulators, Proceedings of the Sixth IFAC Conference in Estimation and Parameter Identification, pp.1454-1459, Washington DC, June, 1982.

Li, Y.F. & Lau, C.C. (1989). Development of fuzzy algorithms for servo systems, IEEE Control Systems Magazine, pp.65-71, April 1989.

Luh, J.Y.S., Walker, M.W. & Paul, R. (1980). Resolved acceleration control of mechanical manipulators, IEEE Transaction on Automatic Control, Vol. AC-25, No.3, 468-474.

Sugeno, M. (1985). An introductory survey of fuzzy control, Information Science, vol.36, pp.59-83, 1985.

Ying, H., siler, W. & Buckley, C. (1990). Fuzzy control theory : A nonlinear case, Automatica, Vol.26, No.3, pp.513-520, 1990.

Lee, C.C. (1990a). Fuzzy logic in control systems : Fuzzy logic controller – Part I, IEEE Transactions on Systems, Man & Cybernatics, Vol.20, No.2, pp.404-418, March/April 1990.

Lee, C.C. (1990b). Fuzzy logic in control systems : Fuzzy logic controller – Part II, IEEE Transactions on Systems, Man & Cybernatics, Vol.20, No.2, pp.419-453, March/April 1990.

Fu, K.S., Gonzalez, R.C. & Lee, C.S.G. (1987). ROBOTICS : Control, Sensing, Vision and Intelligence, McGraw-Hill International Edition, New York, 1987.

Daley, S. (1984). Analysis of fuzzy logic control algorithms and their application to engineering systems, Ph.D theses, University of Leeds, UK. 1984.

Li, Y.F. & Lau, C.C. (1989). Development of fuzzy algorithms for servo systems, IEEE Control System Magazine, pp.65-71, April 1989.

Modelling of Parameter and Bound Estimation Laws for Adaptive-Robust Control of Mechanical Manipulators Using Variable Function Approach

Recep Burkan

1. Introduction

Two different approaches have been actively studied to maintain performance in the presence of parametric uncertainties: adaptive control and robust control. The basic philosophy of adaptive controller is that incorporates some sort of parameter estimation and adaptive controller can learn from experiences in the sense that parameters are changed. Some of the adaptive control laws introduced by Craig et al.(1987), Middleton&Goodwin (1988), Spong&Ortega (1990) require the acceleration measurements and/or the computation of the inverse of the inertia matrix containing estimated parameters. Later, Slotine&Li (1987, 1988) Spong et.al (1990), Egeland&Godhavn (1994) have derived adaptive control algorithms without using the joint accelerations and the inverse of inertia matrix. Other adaptive control laws are proposed in references (Carelli et al 1995, Kelly et al 1989, Burkan&Uzmay 2005, Burkan 2005, Burkan, 2006). Comparative studies of adaptive control laws are given in references (Ortega&Spong 1989, Colbaugh at al 1996).

On the other hand, robust control has been successfully used to design controller with disturbance, unmodelled dynamics and other sources of uncertainty. The papers about application these techniques for the background of robotic application are given in survey papers (Abdullah at al 1991, Sage at al 1999).

Based on the approach of Corless-Leitmann (1981), Spong (1992) developed a new robust control law. In this approach (Spong 1992), the Leitmann (1981) or Corless-Leitmann (1981) approach was used to design a robust controller. Different extension of the scheme by Spong (1992) has been developed by Liu&Goldenberg (1996a, 1997), Yaz (1993), Candudas de Wit et al. (1996). An adaptive scheme of uncertainty bound is given in the papers (Koo&Kim 1994, Burkan and Uzmay 2003b, Burkan and Uzmay 2005). Similar algorithms have proposed by Dawson at. al. (1993) and Zenieh&Corless (1997). Comparative studies of robust controllers are given in the references (Liu & Goldenberg 1996b, Jaritz & Spong (1996).

In pure adaptive control laws, parameters are updated in time and there is no additional control input. However, parameters are not adaptive and fixed (or adaptive) uncertainty bound is used as an additional control input in robust control laws. In the studies (Burkan, 2002; Uzmay & Burkan 2002, Burkan & Uzmay 2003 a, Burkan & Uzmay 2006) adapts previous results on both robust and adaptive control techniques for robot manipulators in an unified scheme, so an adaptive-robust control law is proposed. As distinct from previous studies, variable functions are used in derivation, and parameter and bound estimation laws are updated using exponential and logarithmic functions depending on the robot link parameters and tracking error.

2. Adaptive Control Law

In the absence of friction or other disturbances, the dynamic model of an n-link manipulator can be written as (Spong &Vidyasagar, 1989)

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{1}$$

where q denotes generalised coordinates, τ is the n-dimensional vector of applied torques (or forces), $M(q)$ is the nxn symmetric positive definite inertia matrix, $C(q,\dot{q})\dot{q}$ is the n-dimensional vector of centripetal and Coriolis terms and $G(q)$ is the n-dimensional vector of gravitational terms. Equation (1) can also be expressed in the following form.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Y(q,\dot{q},\ddot{q})\pi \tag{2}$$

where π is a constant (px1) dimensional vector of inertia parameters and Y is an nxp matrix of known functions of the joint position, velocity and acceleration. For any specific trajectory consider known the desired position, velocity and acceleration vectors q_d, $\dot{q}_d$ and $\ddot{q}_d$ and measured the actual position and velocity errors $\tilde{q} = q_d - q$, and $\dot{\tilde{q}} = \dot{q}_d - \dot{q}$. Using the above information a corrected desired velocity and acceleration vectors for nonlinearities and decoupling effects are proposed as:

$$\dot{q}_r = \dot{q}_d + \Lambda\tilde{q} \qquad \ddot{q}_r = \ddot{q}_d + \Lambda\dot{\tilde{q}} \tag{3}$$

where Λ is a positive definite matrix. Then the following control law is considered.

$$\tau = M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + G(q) + K\sigma \tag{4}$$

where $\sigma = \dot{q}_r - \dot{q} = \dot{\tilde{q}} + \Lambda\tilde{q}$ is a corrected velocity error and $K\sigma$ is the vector of PD action.

Suppose that the computational model has the same structure as that of the manipulator dynamic model, but its parameters are not known exactly. The control law (4) is then modified into

$$\begin{aligned}
\tau &= \hat{M}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{G} + K\sigma \\
&= Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\pi} + K\sigma
\end{aligned}$$

(5)

where $\hat{\pi}$ represents the available estimate on the parameters, and accordingly, $\hat{M}$, $\hat{C}$, $\hat{G}$ denote the estimated terms in the dynamic model. Substituting (5) into (2) gives

$$M(q)\dot{\sigma} + C(q,\dot{q})\sigma + K\sigma = -\tilde{M}(q)\ddot{q}_r - \tilde{C}(q,\dot{q})\dot{q}_r - \tilde{G} = -Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\tilde{\pi}$$

(6)

where $\tilde{\pi} = \hat{\pi} - \pi$ is the property of linearity in the parameter error. Error quantities concerning system parameters are characterised by

$$\tilde{M} = \hat{M} - M, \ \tilde{C} = \hat{C} - C, \ \tilde{G} = \hat{G} - G$$

(7)

The Lyapunov function candidate is defined as

$$V(\sigma,\tilde{q},\tilde{\pi}) = \frac{1}{2}\sigma^T M(q)\sigma + \frac{1}{2}\tilde{q}^T B\tilde{q} + \frac{1}{2}\tilde{\pi}^T K_\pi \tilde{\pi} > 0$$

(8)

where π is a p dimensional vector containing the unknown manipulators and load parameters, $\hat{\pi}$ is its estimate and $\tilde{\pi} = \hat{\pi} - \pi$ denotes the parameter estimation error vector. B and K_π are positive definite, usually diagonal matrix. Using the property $\sigma^T[\dot{M}(q) - 2C(q,\dot{q})]\sigma = 0 \ \ \forall \sigma \in R^n$ and choosing $B = 2\Lambda K$, the time derivative of V along the trajectory of system (6) is

$$\dot{V} = -\dot{\tilde{q}}^T K\dot{\tilde{q}} - \tilde{q}^T \Lambda K \Lambda \tilde{q} + \tilde{\pi}^T(K_\pi \dot{\tilde{\pi}} - Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)\sigma)$$

(9)

If the estimate of the parameter vector is updated as the adaptive law

$$\dot{\hat{\pi}} = K_\pi^{-1} Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)\sigma$$

(10)

Equation (9) becomes

$$\dot{V} = -\dot{\tilde{q}}^T K\dot{\tilde{q}} - \tilde{q}^T \Lambda K \Lambda \tilde{q}$$

(11)

So, $\dot{V}$ is negative semidefinite and Equation (6) is stable. It should be noted that $\dot{\hat{\pi}} = \dot{\tilde{\pi}}$ (π is constant) (Sciavicco & Siciliano, 1996). The parameter estimation law (10) can also be written as

$$\hat{\pi} = \int K_\pi^{-1} Y^T (q, \dot{q}, \dot{q}_r, \ddot{q}_r) \sigma dt + \hat{\pi}(0) \tag{12}$$

where $\hat{\pi}(0)$ is the initial estimation of the parameters. The resulting block diagram of the adaptive control law is given in Fig. 1 (Sciavicco & Sciliano, 1996)
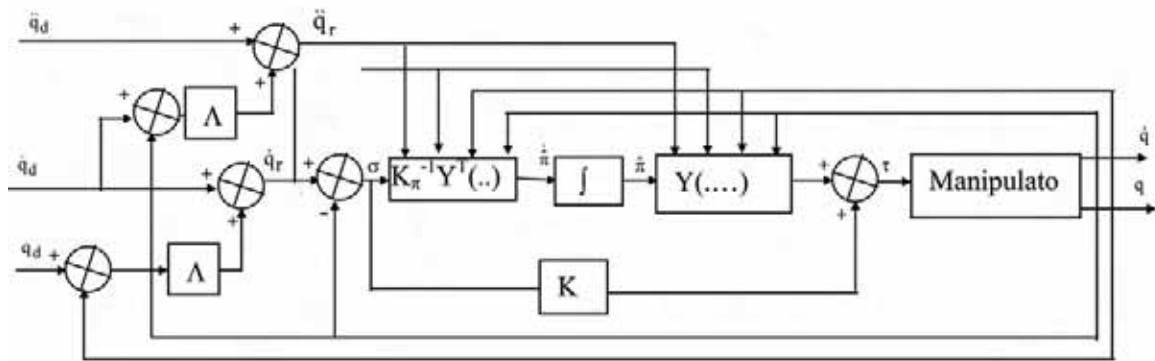
Figure 1. Implementation of the adaptive control law (10) (Sciavicco & Siciliano, 1996).

3. Robust Control Law

Consider the nominal control vector for the model system described by Equations (1) and (2).

$$\begin{aligned}
\tau_0 &= M_0(q)\ddot{q}_r + C_0(q, \dot{q})\dot{q}_r + G_0(q) - K\sigma \\
&= Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\pi_0 - K\sigma
\end{aligned} \tag{13}$$

The definition of the nominal control law τ_0 is based on the adaptive algorithm of Slotine and Li (1987). It is important to understand that the nominal control vector τ_0 in Equation (13) is defined in terms of fixed parameters which are not changed or updated in time as would be an adaptive control strategy. The control input τ can be defined in terms of the nominal control vector τ_0 and a compensation vector for parameter variations as:

$$\tau = \tau_0 + Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)u(t) = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)(\pi_0 + u(t)) - K\sigma \tag{14}$$

where

$$\tilde{q} = q - q_d \, ; \; \dot{q}_r = \dot{q}_d - \Lambda\tilde{q} \, ; \; \ddot{q}_r = \ddot{q}_d - \Lambda\dot{\tilde{q}} \tag{15}$$

It is supposed that the parameter estimation vector π is uncertain and it is assumed that both $\pi_0 \, \varepsilon R^p$ and $\rho \, \varepsilon R$ are known a priori, such that

$$\left\| \tilde{\pi} \right\| = \left\| \pi - \pi_0 \right\| \le \rho \tag{16}$$

Let $\varepsilon > 0$ and the additional control vector as defined by Spong (1992) as:

$$u(t) = \begin{cases} -\rho \dfrac{Y^T\sigma}{\left\| Y^T\sigma \right\|} & \text{if} \quad \left\| Y^T\sigma \right\| > \varepsilon \\[4mm] -\rho \dfrac{Y^T\sigma}{\varepsilon} & \text{if} \quad \left\| Y^T\sigma \right\| \le \varepsilon \end{cases} \tag{17}$$

Considering adaptive control law (Sciavicco & Siciliano, 1996), the block diagram of the pure robust controller is given in Fig. 2.
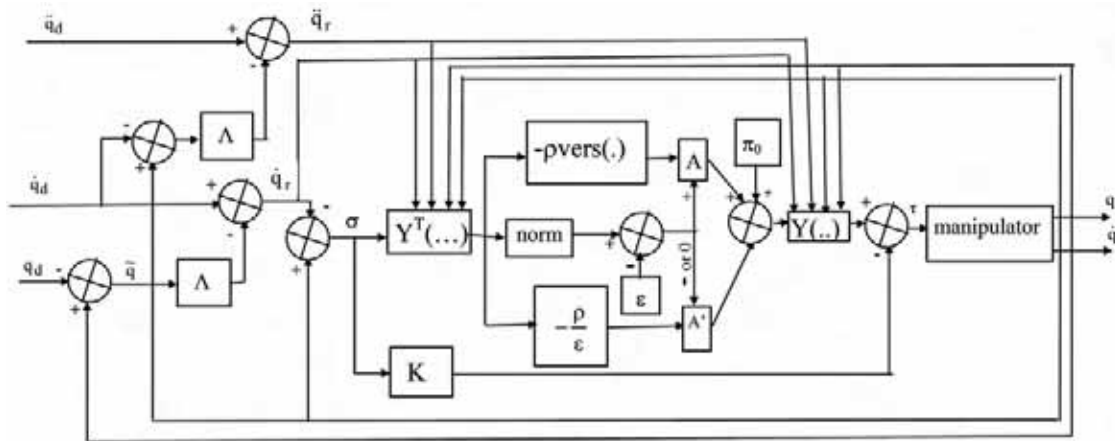
Figure 2 Block diagram of the robust control law. (Burkan &Uzmay, 2003 c)

Since the controller which is defined by Equation (17) consists of two different input depending on ε, the matrices A and A' are introduced to select appropriate control input. The A matrix is diagonal with ones and zeros on the diagonal. When $\left\| Y^T\sigma \right\| - \varepsilon > 0$, a one is present in A, a zero is present in A' and the first additional control input is in effect. When $\left\| Y^T\sigma \right\| - \varepsilon \le 0$ a zero is present in A, a one is present in A', and so the second additional control input is in ef-

fect. Hence, the matrices A and A′ are simple switches which set the mode of additional control input to be used (Burkan & Uzmay, 2003 c).

As a measure of parameter uncertainty on which the additional control input is based, ρ can be defined as

$$\rho = \left(\sum_{i=1}^{p} \rho_i^{\,2} \right)^{1/2} \tag{18}$$

Having a single number ρ to measure the parametric uncertainty may lead to overly conservative design, higher than necessary gains, ect. For this reason we may be interested in assigning different "weights" or gains to the components of τ. We can do this as follows. Suppose that we have a measure of uncertainty for each parameter $\tilde{\pi}_i$ separately as:

$$\lfloor \tilde{\pi}_i \rfloor \le \rho_i \qquad\qquad i=1,2,..,p \tag{19}$$

Let υ_i denote the ith component of the vector $Y^T \sigma$, $\varepsilon_i = i=1,2....p_i$ represent the ith component of ε, and define the ith component of the control input τ_i as (Spong, 1992), then

$$u(t)_i = \begin{cases} -\rho_i \upsilon_i / |\upsilon_i| & \text{if} \quad |\upsilon_i| > \varepsilon_i \\ -(\rho_i / \varepsilon_i)\upsilon_i & \text{if} \quad |\upsilon_i| \le \varepsilon_i \end{cases} \tag{20}$$

4. Adaptive-Robust Control Law

Considering the dynamic model of a n-link robot manipulator given by Equations (1) and (2), the control input vector that comprises the parameter estimation and the additional control input is defined as

$$\tau = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)(\hat{\pi} + \delta(t)) + K\sigma \tag{21}$$

Substituting (21) into (1) and some arrangements yield
$$M(q)\dot{\sigma} + C(q,\dot{q})\sigma + K\sigma = -Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\tilde{\pi} - Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\delta(t) \tag{22}$$

Adaptive robust parameters are identical as adaptive control law in the known parameter case such as σ, q_r, Λ and K. It is assumed that the parameter error is unknown such that

$$\tilde{\pi} = \hat{\pi} - \pi = \rho(t) \tag{23}$$

where $\hat{\pi}$ is the estimate of the available parameters and updated in time. The upper bounding function $\hat{\rho}(t)$ is assumed to be unknown, and should be determined using the estimation law to control the system properly. Finally the error $\tilde{\rho}(t)$ shows the difference between parameter error and upper bounding function as

$$\tilde{\rho}(t) = \rho(t) - \hat{\rho}(t) = \hat{\pi} - \pi - \hat{\rho}(t) \qquad (24)$$

Theorem (Burkan & Uzmay, 2003 a):

Let $\alpha > 0$ be a positive number, π be the unloaded and lower bound of parameter, and ρ be the upper uncertainty bound of Equation (16). The three of them are supposed to be known initially. If the estimate of parameter $\hat{\pi}$ and the additional control input $\delta(t)$ in control law (21) are defined, respectively as

$$\hat{\pi} = -\frac{2}{\alpha} e^{-\alpha t} Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r) + \pi; \quad \delta(t) = \rho e^{-\frac{\alpha}{2} t} \qquad (25)$$

and substitute them in the control input (21) for the trajectory control of the model manipulator, then the tracking errors $\tilde{q}$ and $\dot{\tilde{q}}$ will converge to zero.

Proof:

By taking into account above parameters and control algorithm, the Lyapunov function candidate is defined as

$$V(\sigma, \tilde{q}, \tilde{\rho}(t)) = \frac{1}{2} \sigma^T M(q) \sigma + \frac{1}{2} \tilde{q}^T B \tilde{q} + \frac{1}{2} \tilde{\rho}(t)^T \Gamma \tilde{\rho}(t) \qquad (26)$$

Apart from similar studies, Γ is the positive definite diagonal matrix and change in time. The time derivative of Equation (26) is written as

$$\dot{V} = \sigma^T M(q) \dot{\sigma} + \sigma^T \frac{1}{2} \dot{M}(q) \sigma + \tilde{q}^T B \dot{\tilde{q}} + \frac{1}{2} \tilde{\rho}(t)^T \dot{\Gamma} \tilde{\rho}(t) + \tilde{\rho}(t)^T \Gamma \dot{\tilde{q}}(t) \qquad (27)$$

where

$$\tilde{\rho}(t) = \rho(t) - \hat{\rho}(t) = \hat{\pi} - \pi - \hat{\rho}(t) \; ; \dot{\tilde{\rho}}(t) = \dot{\rho}(t) - \dot{\hat{\rho}}(t) = \dot{\hat{\pi}} - \dot{\hat{\rho}}(t) \qquad (28)$$

Let $B = 2\Lambda K$ and use the property $\sigma^T[\dot{M}(q) - 2C(q, \dot{q})]\sigma = 0$, $\forall \sigma \in R^n$, the time derivative of V along the system (22) is

$$\dot{V} = -\dot{\tilde{q}}^T K \dot{\tilde{q}} - \tilde{q}^T \Lambda K \Lambda \tilde{q} - \sigma^T Y \delta(t) - \sigma^T Y \rho(t) + \frac{1}{2} \tilde{\rho}(t)^T \dot{\Gamma} \tilde{\rho}(t) + \tilde{\rho}(t)^T \Gamma \dot{\tilde{q}}(t) \qquad (29)$$

Since K>0, and Λ>0 the first terms of Equation (29) are less or equal to zero that is:

$$-\dot{\tilde{q}}^T K \dot{\tilde{q}} - \tilde{q}^T \Lambda K \Lambda \tilde{q} \leq 0 \qquad (30)$$

So, in order to find conditions to make $\dot{V} \leq 0$ we concentrate on the remaining terms of the equation. If the rest of Equation (29) is equal to or less than zero, the system will be stable. Substituting Equation (24) into the remaining terms of Equation (29) the following equation is obtained:

$$-\sigma^T Y \delta(t) - \sigma^T Y \rho(t) + \frac{1}{2}[\rho(t) - \hat{\rho}(t)]^T \dot{\Gamma}[\rho(t) - \hat{\rho}(t)] + [\rho(t) - \hat{\rho}(t)]^T \Gamma[\dot{\rho}(t) - \dot{\hat{\rho}}(t)] = 0 \quad (31)$$

Now, in considering $\delta(t)$ as an estimated term of uncertainty bound, that is, $\delta(t) = -\hat{\rho}(t)$ then Equation (31) is written as:

$$\sigma^T Y \hat{\rho}(t) - \sigma^T Y \rho(t) + \frac{1}{2}[\rho(t) - \hat{\rho}(t)]^T \dot{\Gamma}[\rho(t) - \hat{\rho}(t)] + [\rho(t) - \hat{\rho}(t)]^T \Gamma[\dot{\rho}(t) - \dot{\hat{\rho}}(t)] = 0 \quad (32)$$

Taken $[\rho(t) - \hat{\rho}(t)]$ as a common multiplier, Equation (32) is written as:

$$[(\rho(t) - \hat{\rho}(t)]^T[-Y^T\sigma + \frac{1}{2}\dot{\Gamma}[\rho(t) - \hat{\rho}(t)) + \Gamma[\dot{\rho}(t) - \dot{\hat{\rho}}(t)] = 0 \qquad (33)$$

Hence, we look for the conditions for which the equation $-Y^T\sigma + \frac{1}{2}\dot{\Gamma}[\rho(t) - \hat{\rho}(t)) + \Gamma[\dot{\rho}(t) - \dot{\hat{\rho}}(t)] = 0$ is satisfied. The terms constituting the above equation are expressed as

$$\tilde{\rho}(t) = \rho(t) - \hat{\rho}(t); \quad \tilde{\pi} = \rho(t) = \hat{\pi} - \pi; \quad \dot{\rho}(t) = \dot{\hat{\pi}}; \quad \dot{\tilde{\rho}}(t) = \dot{\rho}(t) - \dot{\hat{\rho}}(t) = \dot{\hat{\pi}} - \dot{\hat{\rho}}(t) \qquad (34)$$

Substituting the parameters in Equation (34) into Equation (33) yields

$$-Y^T\sigma + \frac{1}{2}\dot{\Gamma}[\hat{\pi} - \pi - \hat{\rho}(t)] + \Gamma[\dot{\hat{\pi}} - \dot{\hat{\rho}}(t)] = 0 \qquad (35)$$

Then

$$-Y^T\sigma + \frac{1}{2}\dot{\Gamma}(\hat{\pi} - \pi) + \Gamma\dot{\hat{\pi}} - [\frac{1}{2}\dot{\Gamma}\hat{\rho}(t) + \Gamma\dot{\hat{\rho}}(t)] = 0 \qquad (36)$$

A solution for Equation (36) can be derived if it is divided into two equations as:

$$-Y^T \sigma + \frac{1}{2}\dot{\Gamma}(\hat{\pi} - \pi) + \Gamma \dot{\hat{\pi}} = 0 \tag{37}$$

$$-(\frac{1}{2}\dot{\Gamma}\hat{\rho}(t) + \Gamma \dot{\hat{\rho}}(t)) = 0 \tag{38}$$

Equation (37) can also be written as;

$$\frac{1}{2}\dot{\Gamma}(\hat{\pi} - \pi) + \Gamma \dot{\hat{\pi}} = Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\sigma \tag{39}$$

For the proposed approach, Γ and its time derivative are chosen as a positive definite diagonal matrix of the form

$$\Gamma = e^{\alpha t}I, \quad \dot{\Gamma} = \alpha e^{\alpha t}I \tag{40}$$

where I is a pxp dimensional matrix. Substitution of Equation (40) into Equation (39) yields;

$$e^{\alpha t}\dot{\hat{\pi}} + \frac{1}{2}\alpha e^{\alpha t}(\hat{\pi} - \pi) = Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\sigma \tag{41}$$

Dividing Equation (41) by the factor $e^{\frac{\alpha}{2}t}$ result in the following expression.

$$e^{\frac{\alpha}{2}t}\dot{\hat{\pi}} + \frac{1}{2}\alpha e^{\frac{\alpha}{2}t}\hat{\pi} = e^{-\frac{\alpha}{2}t}Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\sigma + \frac{1}{2}\alpha e^{\frac{\alpha}{2}t}\pi \tag{42}$$

Equation (42) can be arranged as

$$\frac{d}{dt}(e^{\frac{\alpha}{2}t}\hat{\pi}) = e^{-\frac{\alpha}{2}t}Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\sigma + \frac{1}{2}\alpha e^{\frac{\alpha}{2}t}\pi \tag{43}$$

For a given instant, Y^T and σ can be assumed to be constant. Integrating both sides of Equation (43) yields;

$$(e^{\frac{\alpha}{2}t}\hat{\pi}) = \int (e^{-\frac{\alpha}{2}t}Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\sigma + \frac{1}{2}\alpha e^{\frac{\alpha}{2}t}\pi)dt = -\frac{2}{\alpha}e^{-\frac{\alpha}{2}t}Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\sigma + e^{\frac{\alpha}{2}t}\pi + C \tag{44}$$

If Equation (44) is divided by $e^{\frac{\alpha}{2}t}$, the result is

$$\hat{\pi} = -\frac{2}{\alpha}e^{-\alpha t}Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)\sigma + \pi + Ce^{\frac{1}{2}\alpha t} \tag{45}$$

If the initial condition is $\hat{\pi}(0)=\pi$, the constant C becomes zero. So, the parameter adaptation algorithm is derived as

$$\hat{\pi} = -\frac{2}{\alpha}e^{-\alpha t}Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)\sigma + \pi \tag{46}$$

Adaptive parameter estimation law is obtained as a solution of Equation (37). As a result of Equation (38), robust parameter estimation law $\hat{\rho}(t)$ can be also obtained. Substitution of Equation (40) into Equation (38) yields;

$$-(e^{\alpha t}\dot{\hat{\rho}}(t) + \frac{1}{2}\alpha e^{\alpha t}\hat{\rho}(t)) = 0 \tag{47}$$

By dividing Equation (47) by the factor $e^{\frac{\alpha}{2}t}$, the following expression is found.

$$-(e^{\frac{\alpha}{2}t}\dot{\hat{\rho}}(t) + \frac{1}{2}\alpha e^{\frac{\alpha}{2}t}\hat{\rho}(t)) = 0 \tag{48}$$

If Equation (48) is arranged according to $\hat{\rho}(t)$

$$-\frac{d}{dt}((e^{\frac{\alpha}{2}t}\hat{\rho}(t))) = 0 \tag{49}$$

Integrating both sides of Equation (49) yields

$$-(e^{\frac{\alpha}{2}t}\hat{\rho}(t)) = C \Rightarrow -(\hat{\rho}(t)) = Ce^{-\frac{\alpha}{2}t} \tag{50}$$

If $\hat{\rho}(0) = \rho$ is taken as an initial condition, the constant C is equivalent to ρ. So, the robust parameter estimation algorithm is derived as

$$\hat{\rho}(t) = -\rho e^{-\frac{\alpha}{2}t} \tag{51}$$

Since $\delta(t) = -\hat{\rho}(t)$, the control vector can be written as

$$\tau = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)[-\frac{2}{\alpha}e^{-\alpha t}Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)\sigma + \pi + \rho e^{-\frac{\alpha}{2}t})] + K\sigma \qquad (52)$$

The block diagram of adaptive-robust control law is shown in Fig. 3.
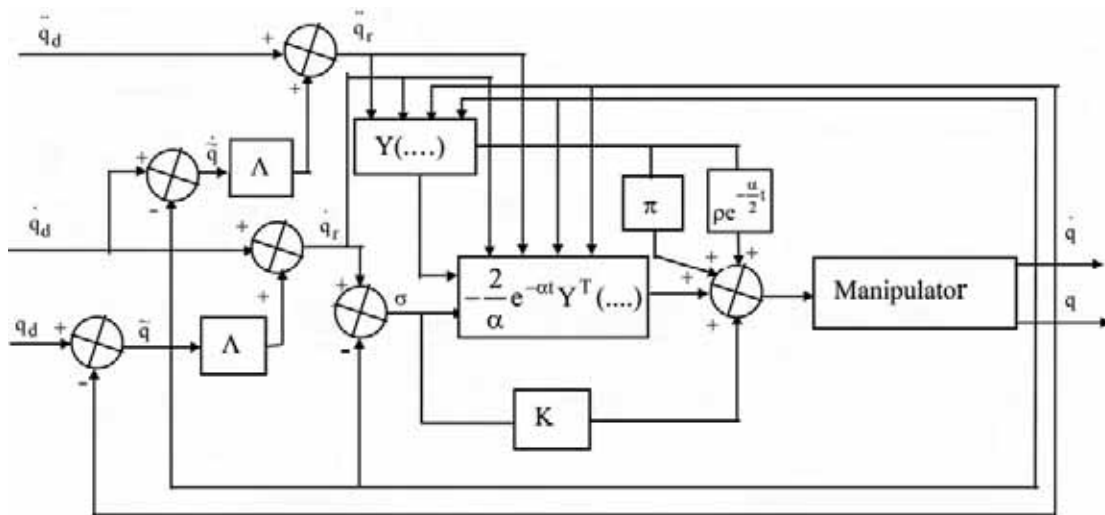
Figure 3. Block diagram of the adaptive-robust control law (52) (Burkan & Uzmay, 2003a)

If Equation (46) and (51) are substituted in Equation (29) it will become a negative semidefinite function of the form of Equation (30). So, the system (22) will be stable under the conditions assumed in the theorem.

At this point, it is very important to choose the variable function Γ in order to solve the Equations (38) and (39), and there is no a certain rule for selection of Γ for this systems. We use system state parameters and mathematical insight to search for appropriate function of Γ as a solution of the first order differential in Equations (38) and (39). For the second derivation, we choose variable function Γ and its derivative such that (Uzmay & Burkan, 2002).

$$\Gamma = e^{\int Y^T\sigma dt} \; ; \; \dot{\Gamma} = Y^T\sigma e^{\int Y^T\sigma dt} \qquad (53)$$

where Γ is a pxp dimensional identity matrix. Substitution of (53) into (39) yields

$$e^{\int Y^T\sigma dt}\dot{\hat{\pi}} + \frac{1}{2}Y^T\sigma e^{\int Y^T\sigma dt}(\hat{\pi} - \pi) = Y^T\sigma \qquad (54)$$

Remembering that $\dot{\tilde{\pi}} = \dot{\hat{\pi}}$ (π is constant). Dividing Equation (54) by $e^{\int Y^T\sigma dt}$ yields

$$\dot{\hat{\pi}} + \frac{1}{2} Y^T \sigma \hat{\pi} = e^{-\int Y^T \sigma dt} Y^T \sigma + \frac{1}{2} Y^T \sigma \pi \tag{55}$$

Multiplying Equation (55) by the factor $e^{\frac{1}{2}\int Y^T \sigma dt}$ results

$$e^{\frac{1}{2}\int Y^T \sigma dt} \dot{\hat{\pi}} + \frac{1}{2} Y^T \sigma e^{\frac{1}{2}\int Y^T \sigma dt} \hat{\pi} = e^{\frac{1}{2}\int Y^T \sigma dt} e^{-\int Y^T \sigma dt} Y^T \sigma + \frac{1}{2} Y^T \sigma e^{\frac{1}{2}\int Y^T \sigma dt} \pi \tag{56}$$

Equation (56) can be arranged as

$$\frac{d}{dt}(e^{\frac{1}{2}\int Y^T \sigma dt} \hat{\pi}) = e^{-\frac{1}{2}\int Y^T \sigma dt} Y^T \sigma + \frac{1}{2} Y^T \sigma e^{\frac{1}{2}\int Y^T \sigma dt} \pi \tag{57}$$

Integrating both sides of Equation (57) yields

$$e^{\frac{1}{2}\int Y^T \sigma dt} \hat{\pi} = \int e^{-\frac{1}{2}\int Y^T \sigma dt} Y^T \sigma dt + \frac{1}{2} \int \pi Y^T \sigma e^{\frac{1}{2}\int Y^T \sigma dt} dt \tag{58}$$

$$e^{\frac{1}{2}\int Y^T \sigma dt} \hat{\pi} = -2e^{-\frac{1}{2}\int Y^T \sigma dt} + \pi e^{\frac{1}{2}\int Y^T \sigma dt} + C \tag{59}$$

By dividing both sides of Equation (59) by $e^{\frac{1}{2}\int Y^T \sigma dt}$, the following result is obtained.

$$\hat{\pi} = -2e^{-\int Y^T \sigma dt} + \pi + Ce^{-\frac{1}{2}\int Y^T \sigma dt} \tag{60}$$

If the condition of $\hat{\pi}(0) = \pi$ is taken as an initial condition, the constant C is equivalent to 2. Hence, the parameter adaptation law is derived as

$$\hat{\pi} = -2e^{-\int Y^T \sigma dt} + \pi + 2e^{-\frac{1}{2}\int Y^T \sigma dt} = 2(e^{-\frac{1}{2}\int Y^T \sigma dt} - e^{-\int Y^T \sigma dt}) + \pi \tag{61}$$

In order to drive $\hat{\rho}(t)$, Equation (53) is substituted into (38) yields

$$-e^{\frac{1}{2}\int Y^T \sigma dt} \dot{\hat{\rho}}(t) + \frac{1}{2} Y^T \sigma e^{-\int Y^T \sigma dt} \hat{\rho}(t) = 0 \tag{62}$$

By dividing $e^{\frac{1}{2}\int Y^T \sigma dt}$ Equation (62), the following expression is found

$$-(e^{\frac{1}{2}\int Y^T \sigma dt} \dot{\hat{\rho}}(t) + \frac{1}{2} Y^T \sigma e^{\frac{1}{2}\int Y^T \sigma dt} \hat{\rho}(t)) = 0 \tag{63}$$

Equation (63) is arranged according to

$$-\frac{d}{dt}(e^{\frac{1}{2}\int Y^T \sigma dt} \hat{\rho}(t)) = 0 \tag{64}$$

Integrate both side of Equation (64) yields

$$(e^{\frac{1}{2}\int Y^T \sigma dt} \hat{\rho}(t)) = -C \tag{65}$$

Then

$$\hat{\rho}(t) = -Ce^{-\frac{1}{2}\int Y^T \sigma dt} \tag{66}$$

If $\hat{\rho}(0) = \rho$ is taken as an initial condition, the constant C will be equivalent to ρ. Hence the bound estimation law is derived as

$$\hat{\rho}(t) = -\rho e^{-\frac{1}{2}\int Y^T \sigma dt} \tag{67}$$

As a result, the adaptive-robust control law is obtained as (Uzmay & Burkan, 2002).

$$\tau = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)[2(e^{-\frac{1}{2}\int Y^T \sigma dt} - e^{-\int Y^T \sigma dt}) + \pi + \rho e^{-\frac{1}{2}\int Y^T \sigma dt})] + K\sigma \tag{68}$$

The block diagram of adaptive-robust control law is shown in Fig. 4.
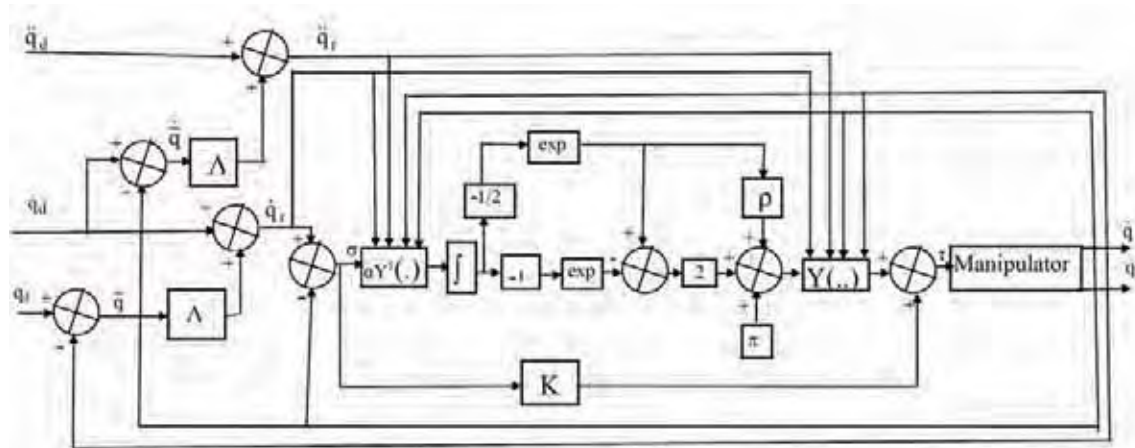
Figure 4. Block diagram of the adaptive-robust control law (68)

Theorem 2: (Burkan & Uzmay, 2006):

Let $\alpha \in R^+$, $(\alpha \int Y^T \sigma dt)_i \geq 0$, i=1,2……p, ρ_i i=1,2……p be the initial estimation of the upper bounding function $\hat{\rho}(t)$ and it is assumed to be known initially. If the estimation of parameter $\hat{\pi}$, estimation of the upper bounding function $\hat{\rho}(t)$ and the additional control input $\delta(t)$ are defined respectively as

$$\hat{\pi} = (1/\alpha) \begin{bmatrix} \dfrac{\ln((\alpha \int Y^T \sigma dt)_1 + 1)}{\int (\alpha Y^T \sigma dt)_1 + 1} \\ \dfrac{\ln((\alpha \int Y^T \sigma dt)_2 + 1)}{\int (\alpha Y^T dt)_2 + 1} \\ ..... \\ \dfrac{\ln((\alpha \int Y^T \sigma dt)_p + 1)}{\int (\alpha Y^T \sigma dt)_p + 1} \end{bmatrix} + \begin{bmatrix} \pi_1 \\ \pi_2 \\ ..... \\ \pi_p \end{bmatrix} ; \ \hat{\rho}(t) = - \begin{bmatrix} \dfrac{\rho_1}{\int (\alpha Y^T \sigma)_1 dt + 1} \\ \dfrac{\rho_2}{\int (\alpha Y^T \sigma)_2 dt + 1} \\ ...... \\ \dfrac{\rho_p}{\int (\alpha Y^T \sigma)_p dt + 1} \end{bmatrix} ; \ \delta(t) = \begin{bmatrix} \dfrac{\rho_1}{\int (\alpha Y^T \sigma)_1 dt + 1} \\ \dfrac{\rho_2}{\int (\alpha Y^T \sigma)_2 dt + 1} \\ ...... \\ \dfrac{\rho_p}{\int (\alpha Y^T \sigma)_p dt + 1} \end{bmatrix} \quad (69)$$

where $\delta(t) = -\hat{\rho}(t)$. Substitute $\hat{\pi}$ and $\delta(t)$ into the control input (21) for the trajectory control of the model manipulator, then the tracking errors $\tilde{q}$ and $\dot{\tilde{q}}$ will converge to zero.

Proof:

In the previous approaches, it is difficult to derive another parameter and bound estimation law because selection of appropriate variable function Γ and solution of the differential equation are not simple. However, the selection of the Γ and solution of the differential equation are simplified in the studies (Burkan 2005, Burkan & Uzmay, 2006) In order to simplify selection of the variable function Γ and simplify the solution of the differential equation, the following Lyapunov function is developed (Burkan & Uzmay, 2006).

$$V(\sigma, \tilde{q}, \tilde{\rho}(t)) = \frac{1}{2} \sigma^T M(q) \sigma + \frac{1}{2} \tilde{q}^T B \tilde{q} + \frac{1}{2} \tilde{\rho}(t)^T \Gamma^2 \tilde{\rho}(t) \quad (70)$$

where Γ is a pxp dimensional diagonal matrix and change in time. The time derivative of Equation (70) is written as

$$\dot{V} = \sigma^T M(q) \dot{\sigma} + \sigma^T \frac{1}{2} \dot{M}(q) \sigma + \tilde{q}^T B \dot{\tilde{q}} + \tilde{\rho}(t)^T \Gamma \dot{\Gamma} \tilde{\rho}(t) + \tilde{\rho}(t)^T \Gamma^2 \dot{\tilde{\rho}}(t) \quad (71)$$

Let $B = 2\Lambda K$ and use the property $\sigma^T [\dot{M}(q) - 2C(q, \dot{q})] \sigma = 0$, $\forall \sigma \in R^n$, the time derivative of V along the system (22) is determined as

$$\dot{V} = -\tilde{q}^{T}K\dot{\tilde{q}} - \tilde{q}^{T}\Lambda K\Lambda\tilde{q} - \sigma^{T}Y\delta(t) - \sigma^{T}Y\rho(t) + \tilde{\rho}(t)^{T}\Gamma\dot{\Gamma}\tilde{\rho}(t) + \tilde{\rho}(t)^{T}\Gamma^{2}\dot{\tilde{p}}(t) \tag{72}$$

Substituting Equation (24) into Equation (72) yields the following equation.

$$-\sigma^{T}Y\delta(t) - \sigma^{T}Y\rho(t) + [\rho(t) - \hat{\rho}(t)]^{T}\Gamma\dot{\Gamma}[\rho(t) - \hat{\rho}(t)] + [\rho(t) - \hat{\rho}(t)]^{T}\Gamma^{2}[\dot{\rho}(t) - \dot{\hat{\rho}}(t)] = 0 \tag{73}$$

Now, let's consider $\delta(t) = -\hat{\rho}(t)$, then Equation (73) is written as:

$$\sigma^{T}Y\hat{\rho}(t) - \sigma^{T}Y\rho(t) + [\rho(t) - \hat{\rho}(t)]^{T}\Gamma\dot{\Gamma}[\rho(t) - \hat{\rho}(t)] + [\rho(t) - \hat{\rho}(t)]^{T}\Gamma^{2}[\dot{\rho}(t) - \dot{\hat{\rho}}(t)] = 0 \tag{74}$$

Taking $[\rho(t) - \hat{\rho}(t)]$ as a common multiplier, Equation (74) is arranged as:

$$[(\rho(t) - \hat{\rho}(t))^{T}[-Y^{T}\sigma + \Gamma\dot{\Gamma}(\rho(t) - \hat{\rho}(t)) + \Gamma^{2}(\dot{\rho}(t) - \dot{\hat{\rho}}(t))] = 0 \tag{75}$$

Substituting the parameters in Equation (34) into (75) yields

$$-Y^{T}\sigma + \Gamma\dot{\Gamma}[\hat{\pi} - \pi - \hat{\rho}(t)] + \Gamma^{2}[\dot{\hat{\pi}} - \dot{\hat{\rho}}(t)] = 0 \tag{76}$$

Then
$$-Y^{T}\sigma + \Gamma\dot{\Gamma}(\hat{\pi} - \pi) + \Gamma^{2}\dot{\hat{\pi}} - [\Gamma\dot{\Gamma}\hat{\rho}(t) + \Gamma^{2}\dot{\hat{\rho}}] = 0 \tag{77}$$

A a result, two different equations can be obtained from Equation (77) as follows.

$$-Y^{T}\sigma + \Gamma\dot{\Gamma}(\hat{\pi} - \pi) + \Gamma^{2}\dot{\hat{\pi}} = 0 \tag{78}$$
$$-(\Gamma\dot{\Gamma}\hat{\rho}(t) + \Gamma^{2}\dot{\hat{\rho}}(t)) = 0 \tag{79}$$

Equation (79) can also be written as
$$\Gamma\dot{\hat{\pi}} + \dot{\Gamma}\hat{\pi} = \Gamma^{-1}Y^{T}\sigma + \dot{\Gamma}\pi \tag{80}$$

since $\dot{\tilde{\pi}} = \dot{\hat{\pi}}$ (π is a constant). Equation (80) is arranged as

$$\frac{d}{dt}(\Gamma\hat{\pi}) = \Gamma^{-1}Y^{T}\sigma + \dot{\Gamma}\pi \tag{81}$$

Integration both sides of Equation (81) yields

$$\Gamma\hat{\pi} = \int\Gamma^{-1}Y^{T}\sigma dt + \int\dot{\Gamma}\pi dt \tag{82}$$

Then, Equation (82) is arranged as

$$\Gamma\hat{\pi} = \int \Gamma^{-1} Y^T \sigma dt + \Gamma\pi + C \tag{83}$$

In Equation (83), $\hat{\pi}$ and Γ are unknown and in order to derive $\hat{\pi}$, Γ must be defined. There is no a certain rule for definition of Γ for this systems. We use system state parameters and mathematical insight to search for appropriate function of Γ as a derivation of the $\hat{\pi}$. For the third derivation, we choose Γ and Γ^{-1}, such that (Burkan & Uzmay, 2006).

$$\Gamma = \begin{bmatrix} (\int \alpha Y^T \sigma dt)_1 + 1 & 0 & \dots & 0 \\ 0 & (\int \alpha Y^T \sigma dt)_2 + 1 & \dots & 0 \\ \dots & & \dots & \dots \\ 0 & & \dots & (\int \alpha Y^T \sigma dt)_p + 1 \end{bmatrix};$$

$$\Gamma^{-1} = \begin{bmatrix} \dfrac{1}{(\int \alpha Y^T \sigma dt)_1 + 1} & 0 & \dots & 0 \\ 0 & \dfrac{1}{(\int \alpha Y^T \sigma dt)_2 + 1} & \dots & 0 \\ \dots & & \dots & \dots \\ 0 & & \dots & \dfrac{1}{(\int \alpha Y^T \sigma dt)_p + 1} \end{bmatrix} \tag{84}$$

where Γ and Γ^{-1} are pxp dimensional diagonal matrices. Substitution of Equation (84) into Equation (83) yields

$$\begin{bmatrix} (\int \alpha Y^T \sigma dt)_1 + 1 & 0 & \dots & 0 \\ 0 & (\int \alpha Y^T \sigma dt)_2 + 1 & \dots & 0 \\ \dots & & \dots & \dots \\ 0 & & \dots & (\int \alpha Y^T \sigma dt)_p + 1 \end{bmatrix} x \begin{bmatrix} \hat{\pi}_1 \\ \hat{\pi}_2 \\ \dots \\ \hat{\pi}_p \end{bmatrix} = \int \begin{bmatrix} \dfrac{(Y^T \sigma)_1}{(\int \alpha Y^T \sigma dt)_1 + 1} \\ \dfrac{(Y^T \sigma)_2}{(\int \alpha Y^T \sigma dt)_2 + 1} \\ \dots \\ \dfrac{(Y^T \sigma)_p}{(\int \alpha Y^T \sigma dt)_p + 1} \end{bmatrix} dt$$

$$+ \begin{bmatrix} (\int \alpha Y^T \sigma dt)_1 + 1 & 0 & \dots & 0 \\ 0 & (\int \alpha Y^T \sigma dt)_2 + 1 & \dots & 0 \\ \dots & & \dots & \dots \\ 0 & & \dots & (\int \alpha Y^T \sigma dt)_p + 1 \end{bmatrix} x \begin{bmatrix} \pi_1 \\ \pi_2 \\ \dots \\ \pi_p \end{bmatrix} + C \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix} \tag{85}$$

After integration, the result is

$$
\begin{bmatrix} (\int \alpha Y^T \sigma dt)_1 +1 & 0 & \dots & 0 \\ 0 & (\int \alpha Y^T \sigma dt)_2 +1 & \dots & 0 \\ \dots & & \dots & \dots \\ 0 & & \dots & (\int \alpha Y^T \sigma dt)_p +1 \end{bmatrix} x \begin{bmatrix} \hat{\pi}_1 \\ \hat{\pi}_2 \\ \dots \\ \hat{\pi}_p \end{bmatrix} = (1/\alpha) \begin{bmatrix} \ln((\int \alpha Y^T \sigma dt)_1 +1) \\ \ln((\int \alpha Y^T \sigma dt)_2 +1) \\ \dots \\ \ln((\int \alpha Y^T \sigma dt)_p +1) \end{bmatrix}
$$

$$
+ \begin{bmatrix} (\int \alpha Y^T \sigma dt)_1 +1 & 0 & \dots & 0 \\ 0 & (\int \alpha Y^T \sigma dt)_2 +1 & \dots & 0 \\ \dots & & \dots & \dots \\ 0 & & \dots & (\int \alpha Y^T \sigma dt)_p +1 \end{bmatrix} x \begin{bmatrix} \pi_1 \\ \pi_2 \\ \dots \\ \pi_p \end{bmatrix} + C \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix} \qquad (86)
$$

Multiplying both sides of Equation (86) by Γ^{-1} and taken initial condition as $\hat{\pi}(0)=\pi$, the constant C will be equivalent to zero. Hence, the parameter adaptation law is derived as

$$
\hat{\pi} = (1/\alpha) \begin{bmatrix} \dfrac{\ln((\alpha \int Y^T \sigma dt)_1 +1)}{\int (\alpha Y^T \sigma dt)_1 +1} \\[4mm] \dfrac{\ln((\alpha \int Y^T \sigma dt)_2 +1)}{\int (\alpha Y^T dt)_2 +1} \\[2mm] \dots \\ \dfrac{\ln((\alpha \int Y^T \sigma dt)_p +1)}{\int (\alpha Y^T \sigma dt)_p +1} \end{bmatrix} + \begin{bmatrix} \pi_1 \\ \pi_2 \\ \dots \\ \pi_p \end{bmatrix} \qquad (87)
$$

Adaptive parameter estimation law is obtained as a solution of Equation (83). As a result of Equation (78), robust parameter estimation law $\hat{\rho}(t)$ can be also obtained. Equation (78) is arranged as

$$
-(\dot{\Gamma}\hat{\rho}(t) + \Gamma\dot{\hat{\rho}}(t)) = 0 \qquad (88)
$$

If Equation (88) is arranged according to $\hat{\rho}(t)$

$$
-(\frac{d}{dt}\Gamma\hat{\rho}(t)) = 0 \qquad (89)
$$

Integrating both sides of Equation (89) yields

$$
-(\Gamma\hat{\rho}(t)) = C \Rightarrow \hat{\rho}(t) = -\Gamma^{-1}C \qquad (90)
$$

If $\hat{\rho}(0) = \rho$ is taken as an initial condition as would be defined in Equation (90), the constant C will be equivalent to ρ. So, the robust parameter estimation algorithm is derived as

$$
\hat{\rho}(t) = -\Gamma^{-1}\rho = -\begin{bmatrix} \dfrac{\rho_1}{\int(\alpha Y^T\sigma)_1\,dt+1} \\[2mm] \dfrac{\rho_2}{\int(\alpha Y^T\sigma)_2\,dt+1} \\[2mm] ...... \\[2mm] \dfrac{\rho_p}{\int(\alpha Y^T\sigma)_p\,dt+1} \end{bmatrix}
\tag{91}
$$

Since $\delta(t) = -\hat{\rho}(t)$, the control vector in Equation (21) can be written as

$$
\tau = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\,[(1/\alpha)\begin{bmatrix} \dfrac{\ln((\alpha\int Y^T\sigma dt)_1+1)}{\int(\alpha Y^T\sigma dt)_1+1} \\[2mm] \dfrac{\ln((\alpha\int Y^T\sigma dt)_2+1)}{\int(\alpha Y^T dt)_2+1} \\[2mm] ..... \\[2mm] \dfrac{\ln((\alpha\int Y^T\sigma dt)_p+1)}{\int(\alpha Y^T\sigma dt)_p+1} \end{bmatrix} + \begin{bmatrix} \pi_1 \\ \pi_2 \\ ...... \\ \pi_p \end{bmatrix} + \begin{bmatrix} \dfrac{\rho_1}{\int(\alpha Y^T\sigma)_1\,dt+1} \\[2mm] \dfrac{\rho_2}{\int(\alpha Y^T\sigma)_2\,dt+1} \\[2mm] ...... \\[2mm] \dfrac{\rho_p}{\int(\alpha Y^T\sigma)_p\,dt+1} \end{bmatrix}] + K\sigma
\tag{92}
$$

The resulting block diagram of the proposed adaptive-robust control law is given in Fig. 5.
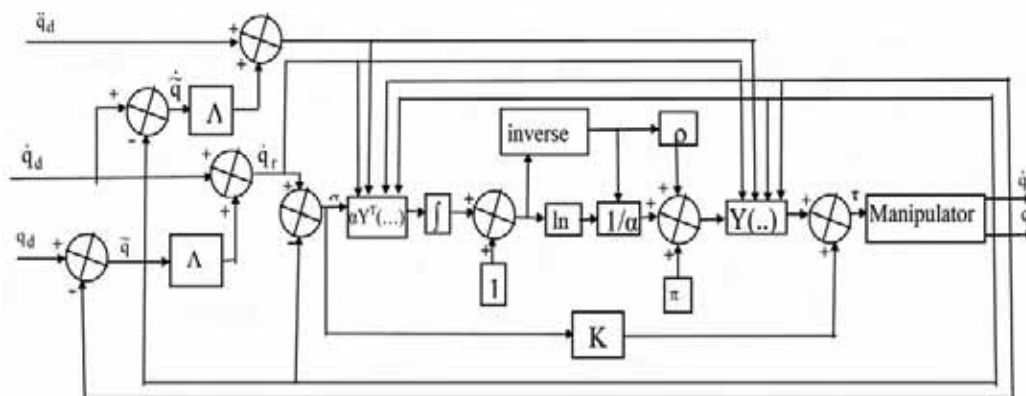
Figure 5. Implementation of the adaptive-robust control law (92) (Burkan & Uzmay, 2006).

For the fourth derivation, Γ and Γ^{-1} are chosen such that

$$\Gamma = \begin{bmatrix} \dfrac{1}{(\alpha_1 Y^T \sigma)_1 + \beta_1} & 0 & \dots & 0 \\[2mm] 0 & \dfrac{1}{(\alpha_2 Y^T \sigma)_2 + \beta_2} & \dots & 0 \\[2mm] \dots & & \dots & \dots \\[2mm] 0 & & \dots & \dfrac{1}{(\alpha_p Y^T \sigma)_p + \beta_p} \end{bmatrix}$$

(93)

$$\Gamma^{-1} = \begin{bmatrix} (\alpha_1 Y^T \sigma)_1 + \beta_1 & 0 & \dots & 0 \\ 0 & (\alpha_2 Y^T \sigma)_2 + \beta_2 & \dots & 0 \\ \dots & & \dots & \dots \\ 0 & & \dots & (\alpha_p Y^T \sigma)_p + \beta_p \end{bmatrix}$$

Substitution of Equation (93) into Equation (83) yields

$$\begin{bmatrix} \dfrac{1}{(a_1 Y^T \sigma)_1 + \beta_1} & 0 & \dots & 0 \\[2mm] 0 & \dfrac{1}{(a_2 Y^T \sigma)_2 + \beta_2} & \dots & 0 \\[2mm] \dots & & \dots & \dots \\[2mm] 0 & & \dots & \dfrac{1}{(a_p Y^T \sigma)_p + \beta_p} \end{bmatrix} x \begin{bmatrix} \hat{\pi}_1 \\ \hat{\pi}_2 \\ \dots \\ \hat{\pi}_p \end{bmatrix} = \int \begin{bmatrix} ((a_1 Y^T \sigma)_1 + \beta_1)(Y^T \sigma) \\ ((a_2 Y^T \sigma)_2 + \beta_2)(Y^T \sigma)_2 \\ \dots \dots \dots \\ ((a_p Y^T \sigma)_p + \beta_p)(Y^T \sigma)_p \end{bmatrix} dt$$

(94)

$$+ \begin{bmatrix} \dfrac{1}{(a_1 Y^T \sigma)_1 + \beta_1} & 0 & \dots & 0 \\[2mm] 0 & \dfrac{1}{(a_2 Y^T \sigma)_2 + \beta_2} & \dots & 0 \\[2mm] \dots & & \dots & \dots \\[2mm] 0 & & \dots & \dfrac{1}{(a_p Y^T \sigma)_p + \beta_p} \end{bmatrix} x \begin{bmatrix} \pi_1 \\ \pi_2 \\ \dots \\ \pi_p \end{bmatrix} + C \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}$$

After integration, the result is

$$
\begin{bmatrix}
\dfrac{1}{(a_1 Y^T \sigma)_1 + \beta_1} & 0 & \dots & 0 \\[2mm]
0 & \dfrac{1}{(a_2 Y^T \sigma)_2 + \beta_2} & \dots & 0 \\[2mm]
\dots & & \dots & \dots \\[2mm]
0 & & \dots & \dfrac{1}{(a_p Y^T \sigma)_p + \beta_p}
\end{bmatrix}
x
\begin{bmatrix}
\hat{\pi}_1 \\ \hat{\pi}_2 \\ \dots \\ \hat{\pi}_p
\end{bmatrix}
=
\begin{bmatrix}
0.5a_1 (\int Y^T \sigma dt)_1^2 + \beta_1 \int (Y^T \sigma)_1 \\
0.5a_2 (\int Y^T \sigma dt)_2^2 + \beta_2 \int (Y^T \sigma)_2 \\
\dots\dots\dots \\
0.5a_p (\int Y^T \sigma dt)_p^2 + \beta_p \int (Y^T \sigma)_p
\end{bmatrix}
$$

(95)

$$
\begin{bmatrix}
\dfrac{1}{(a_1 Y^T \sigma)_1 + \beta_1} & 0 & \dots & 0 \\[2mm]
0 & \dfrac{1}{(a_2 Y^T \sigma)_2 + \beta_2} & \dots & 0 \\[2mm]
\dots & & \dots & \dots \\[2mm]
0 & & \dots & \dfrac{1}{(a_p Y^T \sigma)_p + \beta_p}
\end{bmatrix}
x
\begin{bmatrix}
\pi_1 \\ \pi_2 \\ \dots \\ \pi_p
\end{bmatrix}
+ C
\begin{bmatrix}
1 \\ 1 \\ \dots \\ 1
\end{bmatrix}
$$

Multiplying both sides of Equation (95) by Γ^{-1} and taken initial condition as $\hat{\pi}(0) = \pi$, the constant C will be equivalent to zero. Hence, the parameter adaptation law is derived as

$$
\hat{\pi} =
\begin{bmatrix}
(\alpha_1 Y^T \sigma)_1 + \beta_1)(0.5\alpha_1 (\int Y^T \sigma dt)_1^2 + \beta_1 \int (Y^T \sigma)_1 dt) \\
(\alpha_2 Y^T \sigma)_2 + \beta_2)(0.5\alpha_2 (\int Y^T \sigma dt)_2^2 + \beta_1 \int (Y^T \sigma)_2 dt) \\
\dots\dots\dots\dots\dots\dots \\
.(\alpha_p Y^T \sigma)_p + \beta_p)(0.5\alpha_p (\int Y^T \sigma dt)_p^2 + \beta_1 \int (Y^T \sigma)_p dt)
\end{bmatrix}
+
\begin{bmatrix}
\pi_1 \\ \pi_2 \\ \dots \\ \pi_p
\end{bmatrix}
$$

(96)

If $\hat{\rho}(0) = \rho$ is taken as an initial condition as would be defined in Equation (90), the constant C will be equivalent to ρ. So, the upper bounding function is derived as

$$
\hat{\rho}(t) = -
\begin{bmatrix}
(\alpha_1 Y^T \sigma)_1 + \beta_1)\rho_1 \\
(\alpha_2 Y^T \sigma)_2 + \beta_2)\rho_2 \\
\dots\dots\dots\dots\dots\dots \\
.(\alpha_p Y^T \sigma)_p + \beta_p)\rho_p
\end{bmatrix}
$$

(97)

As a result, the fourth adaptive-robust control law is derived as

$$
\tau = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)[\begin{bmatrix} (\alpha_1 Y^T\sigma)_1 + \beta_1)(0.5\alpha_1(\int Y^T\sigma dt)_1^2 + \beta_1\int(Y^T\sigma)_1 dt) \\ (\alpha_2 Y^T\sigma)_2 + \beta_2)(0.5\alpha_2(\int Y^T\sigma dt)_2^2 + \beta_1\int(Y^T\sigma)_2 dt) \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ .(\alpha_p Y^T\sigma)_p + \beta_p)(0.5\alpha_p(\int Y^T\sigma dt)_p^2 + \beta_1\int(Y^T\sigma)_p dt) \end{bmatrix} + \begin{bmatrix} \pi_1 \\ \pi_2 \\ \dots \\ \pi_p \end{bmatrix}
$$

(98)

$$
+ \begin{bmatrix} (\alpha_1 Y^T\sigma)_1 + \beta_1)\rho_1 \\ (\alpha_2 Y^T\sigma)_2 + \beta_2)\rho_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ .(\alpha_p Y^T\sigma)_p + \beta_p)\rho_p \end{bmatrix}] + K\sigma
$$

The resulting block diagram of the proposed adaptive-robust control law is given in Fig. 6.
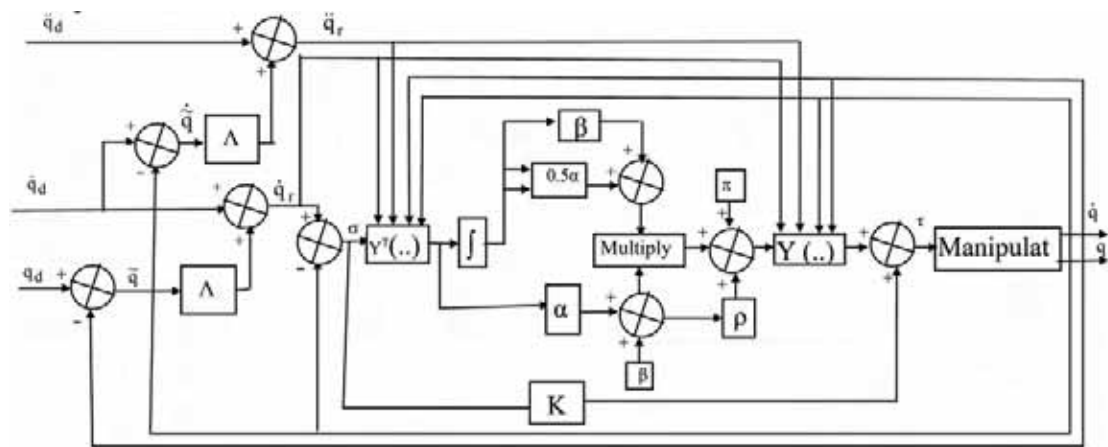
Figure 6. Implementation of the adaptive-robust control law (98)

5. Dynamic Model and Parametric Uncertainties

As an illustrations, a two-link robot arm manipulators shown in Fig. 7. The robot link parameters are

$$\pi_1 = m_1 l_{c1}^2 + m_2 l_1^2 + I_1 \quad \pi_2 = m_2 l_{c2}^2 + I_2 \quad \pi_3 = m_2 l_1 l_{c2}$$

$$\pi_4 = m_1 l_{c1} \quad \pi_5 = m_2 l_1 \quad \pi_6 = m_2 l_{c2}$$

(99)
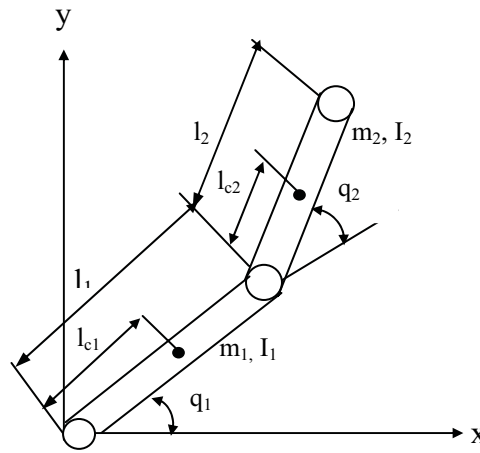
Figure 7. Two-link planar robot (Spong, 1992)

With this parameterization, the dynamic model in Equation (1) can be written as

$$Y(q, \dot{q}, \ddot{q})\pi = \tau \tag{100}$$

The component y_{ij} of $Y(q, \dot{q}, \ddot{q})$ are given as

$$y_{11} = \ddot{q}_1 ; \quad y_{12} = \ddot{q}_1 + \ddot{q}_2 ;$$
$$y_{13} = \cos(q_2)(2\ddot{q}_1 + \ddot{q}_2) - \sin(q_2)(\dot{q}_2^2 + 2\dot{q}_1\dot{q}_2) ;$$
$$y_{14} = g_c\cos(q_1);$$
$$y_{15} = g_c\cos(q_1);$$
$$y_{16} = g_c\cos(q_1+q_2) ;$$

(101)

$$y_{21} = 0;$$
$$y_{22} = \ddot{q}_1 + \ddot{q}_2 ;$$
$$y_{23} = \cos(q_2)\ddot{q}_1 + \sin(q_2)(\dot{q}_1^2) ;$$
$$y_{24} = 0 ;$$

y$_{25}$=0 ;
y$_{26}$= g$_c$cos(q$_1$+q$_2$).

$Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)$ has the component

$y_{11} = \ddot{q}_{r1}$;
$y_{12} = \ddot{q}_{r1} + \ddot{q}_{r2}$;
$y_{13} = \cos(q_2)(2\ddot{q}_{r1} + \ddot{q}_{r2}) - \sin(q_2)(\dot{q}_1\dot{q}_{r2} + \dot{q}_1\dot{q}_{r2} + \dot{q}_2\dot{q}_{r2})$;
y$_{14}$=g$_c$cos(q$_1$);
y$_{15}$= g$_c$cos(q$_1$) ;
y$_{16}$= g$_c$cos(q$_1$+q$_2$)

$$\tag{102}$$

y$_{21}$=0;
$y_{22} = \ddot{q}_{r1} + \ddot{q}_{r2}$;
$y_{23} = \cos(q_2)\ddot{q}_{r1} + \sin(q_2)(\dot{q}_1\dot{q}_{r1})$;
y$_{24}$=0 ;
y$_{25}$=0 ;
y$_{26}$= g$_c$cos(q$_1$+q$_2$).

For illustrated purposes, let us assume that the parameters of the unloaded manipulator are known and the chosen values of the link parameters are given by Table 1. Using these values in Table 1, the ith component of π obtained by means of Equation (99) are given in Table 2. These parametric values also show lower and unloaded robot parameters.

m$_1$	m$_2$	l$_1$	l$_2$	l$_{c1}$	l$_{c2}$	I$_1$	I$_2$
10	5	1	1	0.5	0.5	10/12	5/12

Table 1. Parameters of the unloaded arm (Spong, 1992)

π_1	π_2	π_3	π_4	π_5	π_6
8.33	1.67	2.5	5	5	2.5

Table 2. πi for the unloaded arm (Spong, 1992)

If an unknown load carried by the robot is regarded as part of the second link, then the parameters m$_2$, l$_{c2}$, and I$_2$ will change m$_2$+Δm$_2$, l$_{c2}$+Δl$_{c2}$ and I$_2$+ΔI$_2$, respectively. A controller will be designed that provides robustness in the intervals

$$0 \leq \Delta m_2 \leq 10 \; ; \quad 0 \leq \Delta l_{c2} \leq 0.5 \; ; \quad 0 \leq I_2 \leq \frac{15}{12} \qquad (103)$$

π_0 is chosen as a vector of nominal parameters and it also has the loaded arm parameters and their upper bounds. The computed values for ith component of π_0 are given in Table 3.

π_{01}	π_{02}	π_{03}	π_{04}	π_{05}	π_{06}
13.33	8.96	8.75	5	10	8.75

Table 3. Nominal parameter vector π0 (Spong, 1992)

With this choice of nominal parameter vector π_0 and uncertainty range given by (103), it is an easy matter to calculate the uncertainty bound ρ as follows:

$$\left\| \tilde{\pi} \right\|^2 = \sum_{i=1}^{6} (\pi_{i0} - \pi_i)^2 \leq 181.26 \qquad (104)$$

and thus $\rho = \sqrt{181.26} = 13.46$. Since extended algorithm (20) is used, the uncertainty bounds for each parameter separately are shown in Table 4. The uncertainty bounds ρ_i in Table 4 are simply the difference between values given in Table 3 and Table 2 and that the value of ρ is the Euclidean norm of the vector with components ρ_i (Spong, 1992).

ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
5	7.29	6. 25	0	5	6.25

Table 4. Uncertainty bound (Spong, 1992)

7. Conclusion

In the studies (Burkan, 2002; Uzmay & Burkan 2002, Burkan & Uzmay 2003 a), it is very difficult to use different variable functions for other derivation, and derivation of parameter and bound estimation laws are also not simple. However, in the recent studies (Burkan, 2005; Burkan & Uzmay 2006), first of all, a new method is developed in order to derive new parameter and bound estimation laws based on the Lyapunov function that guarantees stability of the uncertain system and the studies (Burkan, 2002; Uzmay&Burkan, 2002; Burkan&Uzmay, 2003a) provides basis of this study. In this new method, deriva-

tion of the parameter and bound estimation laws are simplified and it is not only possible to derive a single parameter and bound estimation laws, but also it is possible to derive various parameters and bound estimation laws using variable functions.

Parameters and bound estimation laws can be derived depending the variable function Γ, and if another appropriate variable function Γ is chosen, it will be possible to derive other adaptive-robust control laws. In derivation, other integration techniques are also possible to use in derivation for the new parameter and bound estimation laws.

$\hat{\pi}$ and $\hat{\rho}(t)$ are error-derived estimation rules act as a compensators, that is, estimates the most appropriate parameters and upper bounding function to reduce tracking error. The aim of this approach is to solve for finding a control law that ensures limited tracking error, and not to determine the actual parameters and upper bounding function. $\hat{\pi}$ is considered as an adaptive compensator, $\hat{\rho}(t)$ is implemented as a robust controller and both of them are employed during the control process. This has the advantages that the employed adaptive controller increases the learning, while the employed robust controller offers the ability to reject disturbance and ensures desired transient behaviour. This improvement is achieved by computation of the upper bounding function.

8. References

Abdullah, C.; Dawson, D.; Dorato, P & Jamshidi, M. (1991) Survey of robot control for rigid robots, *IEEE Control System Magazine*, Vol 11, No. 2, 24-30, ISSN: 1066-033X

Burkan, R. (2002). New approaches in controlling robot manipulators with parametric uncertainty, Ph.D. Thesis, Erciyes University, Institute of Science, Turkey.

Burkan R, Uzmay İ. (2005). A model of parameter adaptive law with time varying function for robot control, *Applied Mathematical Modelling,* Vol. 29, 361-371, ISSN: 0307-904X

Burkan, R. (2005). Design of an adaptive control law using trigonometric functions for robot manipulators, *Robotica*, Vol.23, 93-99, ISSN:0263-5747.

Burkan, R. & Uzmay, İ. (2003 a). Variable upper bounding approach for adaptive- robust control in robot control, *Journal of Intelligent & Robotic Systems*, Vol.37, No.4, 427-442, ISSN:0921-0296.

Burkan, R. & Uzmay, İ. (2003 b). Upper bounding estimation for robustness to the parameter uncertainty in trajectory control of robot arm, *Robotics and Autonomous Systems*, Vol.45, 99-110, ISSN: 0921-8890

Burkan, R. & Uzmay, İ. (2003 c). A Comparison of Different Control Laws in Trajectory Control for a Revolute-Jointed Manipulator, *Turkish Journal of*

Engineering and Environmental Sciences, Vol.27, No.5, 315-331, ISSN:1300-0160.

Burkan, R & Uzmay, İ. (2005). Logarithmic Based Robust Approach To Parametric Uncertainty For Control of Robot Manipulators, *International Journal of Robust and Nonlinear Control*, Vol.15, 427-436, ISSN: 1049-8923.

Burkan, R. (2006). Modelling of a logarithmic parameter adaptation law for adaptive control of mechanical manipulators, *Robotica*, Vol. 24, No.4, 523-525, ISSN: 0263-5747

Burkan, R. & Uzmay, İ. (2006), Application of logarithmic–based parameter and upper bounding estimation rules to adaptive- robust control of robot manipulators, *European Journal of Control*, Vol. 12, No.2, 156-170, ISSN: 0947-3580.

Canudas De Wit, C.; Siciliano, B. & Bastin, G. (1996). *Theory of Robot Control*, ISBN:3-540-76054-7, Springer, London

Carelli, C.; Camacho, E. F. & Patino, D. (1995). A neural-network-based feedforward adaptive controller for robots, *IEEE Transactions on Systems and Cybernetics*, Vol.2, 1281-1288, ISSN: 1083-4427.

Colbaugh, R.; Glass, K. & Seraji, H. (1996). Adaptive tracking control of manipulators: Theory and experiments, *Robotics & Computer-Integrated Manufacturing*, Vol.12, No.3, 209-216, ISSN: 0736-5845

Corless, M. & Leitmann, G. (1981). Continuous feedback guaranteeing uniform ultimate boundedness for uncertain dynamic systems, *IEEE Transactions Automatic Control*, Vol.26, 1139-1144, ISSN: 0018-9286.

Craig, J. J.; Hsu, P. & Sastry, S. S. (1987). Adaptive control of robot manipulator, *The International Journal of Robotics Research*, Vol.6, 16-28, ISSN: 0278-3649

Dawson, D. M.; Qu, Z. & Duffie, J. (1993). Robust tracking control of robot manipulators: Theory, simulation and implementation, *Robotica*, Vol.11, 201-208, ISSN:0263-5747

Egeland, O. & Godhavn, J. M. (1994). A note on Lyapunov stability for adaptive robot control, *IEEE Transactions on Automatic Control*, Vol.39, No.8, 1671-1673, ISSN: 0018-9286

Jaritz, A. & Spong, M. W. (1996). An experimental comparison of robust control algorithms on a direct drive manipulators, *IEEE Transactions on Control Systems Technology*, Vol.14, No.6, 627-640, ISSN: 1063-6536

Koo, K. M. & Kim, J. H. (1994). Robust control of robot manipulators with parametric uncertainty, *IEEE Transactions Automatic Control*, Vol. 39, No.(6, 1230-1233. ISSN: 0018-9286

Kelly, R.; Carelli, R. & Ortega, R. (1989). Adaptive motion control design of robot manipulators: an input output approach, *International Journal of Control*, Vol.50, No.6, 2563-2581. ISSN: 0020-7179

Leitmann, G. (1981). On the efficiency of nonlinear control in uncertain linear system, *Journal of Dynamic Systems Measurement and Control*, Vol.102, 95-102, ISSN:0022-0434.

Liu, G. & Goldenberg, A. A. (1996a). Uncertainty decomposition-based robust control of robot manipulators, *IEEE Transactions on Control Systems Technology*, Vol.4, 384-393, ISSN: 1063-6536.

Liu, G. & Goldenberg, A. A. (1996b). Comparative study of robust saturation-based control of robot manipulators: Analysis and experiments, *International journal of Robotics Research,* Vol.15, 474-491, ISSN: 0278-3649.

Liu, G. & Goldenberg, A. A. (1997). Robust control of robot manipulators based on dynamics decomposition, *IEEE Transactions on Robotics and Automation*, Vol.13, 783-789, ISSN: 1552-3098.

Middleton, R. H. & Goodwin, G. C. (1988). Adaptive computed torque control for rigid link manipulators, *System Control Letters,* Vol.10, 9-16, ISSN:0167-6911.

Ortega, R. & Spong, M. W. (1989). Adaptive motion control of rigid robots: A tutorial. *Automatica*, Vol.23, No.6, 877-888, ISSN: 0005-1098.

Sage, H. G.; De Mathelin, M. F. & Ostretag, E. (1999). Robust control of robot manipulators: A survey, *International Journal of Control*, Vol.72, No.16, 1498-1522, ISSN: 0020-7179

Sciavicco, L. & Siciliano, B. (1996). *Modelling and Control of Robot Manipulators*, The McGraw-Hill Companies, ISBN:0-07-057217-8, New York.

Spong, M. W. & Ortega, R. (1990). On adaptive inverse dynamics control of rigid robots, *IEEE Transactions on Automatic Control,* Vol.35, No.1, 92-95, ISSN: 0018-9286.

Spong, M. W. & Vidyasagar, M. (1989). Robot *Dynamics and Control*, Willey, New York, 1989, ISBN:0-471-50352-5.

Slotine, J. J. & Li, W.(1987). On the adaptive control of robotic manipulator, *The International Journal of Robotics Research,* Vol.6, No.3, 49-59,ISSN:0278-3649.

Slotine, J. J. & Li, W. (1988). Adaptive manipulator control: A case study, *IEEE Transactions on Automatic Control,* Vol.33, No.11, 994-1003, ISSN: 0018-9286.

Spong, M. W; Ortega, R & Kelley, R. (1990). Comment on adaptive manipulator control: A case study, *IEEE Transactions on Automatic Control,* Vol.35, No.6, 761-762, ISSN: 0018-9286.

Spong, M. W. (1992). On the robust control of robot manipulators, *IEEE Transactions on Automatic Control,* Vol.37, 1782-1786, ISSN: 0018-9286.

Uzmay, İ. & Burkan, R. (2002). Parameter estimation and upper bounding adaptation in adaptive-robust control approaches for trajectory control of robots, *Robotica,* Vol.20, 653-660, ISSN: 0263-5747

Yaz, E.(1993). Comments on the robust control of robot manipulators, *IEEE Transactions on Automatic Control,* Vol.38, No.38, 511-512, ISSN: 0018-9286.

Zenieh, M. & Corless, M. A. (1997). A simple robust r-α tracking controllers for uncertain fully-actuated mechanical systems, *Journal of Dynamics Systems, Measurement and Control*, Vol.119, 821-825, ISSN:0022-0434.

Soft Computing Based Mobile Manipulator Controller Design

Abdessemed Foudil and Benmahammed Khier

1. Introduction

During the last decades, numerous papers have been written on how to apply neuronal networks, fuzzy (multi-valued) logic, genetic algorithms and related ideas of learning from data and embedding structured human knowledge. These concepts and associated algorithms form the field of soft computing. They have been recognized as attractive alternatives to the standard, well established hard computing (conventional) paradigms. Traditional hard computing methods are often too cumbersome for today's problems. They always require a precisely stated analytical model and often a lot of computation time. Soft computing techniques which emphasize gains in understanding system behaviour in exchange for unnecessary accuracy have proved to be important practical tools for many real world problems. Because they are universal approximators of any multivariate function, the neuronal networks and fuzzy logic are of particular interest for modelling highly nonlinear, unknown or partial known complex systems. Due to their strong learning and cognitive ability and good tolerance to uncertainties and imprecision, soft computing techniques have found wide applications in robotic systems control. According to Zadeh (Zadeh, 1994), the basic premises of soft computing are

- The real world is pervasively imprecise and uncertain.
- Precision and certainty carry a cost.

And the guiding principle of soft computing, which follows from these premises, is exploit tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, and low solution costs.

Both the premises and the guiding principle differ strongly from those in classical hard computing, which require precision, certainty, and rigor. However, since precision and certainty carry a cost, the soft computing approach to computation, reasoning, and decision making should exploit the tolerance for imprecision (inherent in human reasoning) when necessary. A long standing tradition in science gives more respect to theories that are quantitative, formal, and precise than those that are qualitative, informal, and approximate. Many

contemporary problems do not lend themselves to precise solutions such as the mobile robot coordination.

Usually, learning implies acquiring knowledge about a previously unknown or partially known system. Learning from experimental data (statistical learning) and fuzzy logic are the most important constituents of soft computing. In nowadays, the soft computing is considered as a discipline that includes an emerging and more or less established family of problem stating and solving methods that attempt to imitate the intelligence found in nature. Very often, the devices and algorithms that can learn from data are characterized as intelligent. With the increasing complexity of industrial processes, the link among ambiguity, robustness and performance of these systems has become increasingly evident. This may explain the dominant role of emerging intelligent systems. The human mental abilities of learning, generalizing, memorizing and predicting should be the foundations of any intelligent system. The intelligent system is supposed to possess human like expertise within specific domain, adapts itself and learns to do better in changing environments and explains how it makes decisions and takes actions. It should be capable to deal with the large amount of data coming from different sensors, to plan under large uncertainties, to set the hierarchy of priorities, and to coordinate many different tasks simultaneously.

The behaviour coordination architectures can be divided into two categories: arbitration and command fusion schemes. In arbitration, the selected dominant behaviour controls the robot until the next decision cycle, whereas the motor commands of the suppressed behaviours are completely ignored. The command fusion approaches aggregate the control actions of multiple concurrently active behaviors into a consensual decision. Fuzzy rule based hierarchical architectures offer an alternative approach to robotic behaviour coordination. A set of primitive, self contained behaviours is encoded by fuzzy rule bases that map perceptions to motor commands. Reactive behaviours in isolation are incapable of performing autonomous navigation in complex environments. However, more complex tasks can be accomplished through combination and cooperation among primitive behaviours. A composite behaviour is implemented as a supervisory controller that activates and deactivates the underlying primitive behaviours according to the current robot context and goals. A fuzzy coordination offers the advantage that behaviours are active to a certain degree, rather than being either switched on or off. The weight with which a behavior contributes to the overall decision depends on its current applicability and desirability.

The goal of this chapter is to present the main role of the soft computing and the contribution it can bring in the control of the complicated systems such as robotic systems. By this, we meant only a brief overview of the subject.

2. Problems and Principle of Robot Control

Industrial Robotics includes mechanical systems that are highly non-linear, ill defined and subject to a variety of unknown disturbances. The control of such systems is facing challenges in order to meet the requirements that can be of different natures. A lot of effort has been devoted to capitalizing on the advances in mathematical control theory resulting in several techniques appeared to tackle this kind of mechanical systems. The navigational planning for mobile robot is a search problem, where the robot has to plan a path from a given initial position to goal position. The robot must move without

hitting an obstacle in its environment. So, the obstacles in the robot workspace act as constraints to the navigational planning problem. A genetic algorithm can solve the problem, by choosing an appropriate fitness function that takes into account the distance of the planned path segments from the obstacles, the length of the planned path and the linearity of the path as practicable. Furthermore, the learning process is constrained by the three mutually compromising constraints complexity of the task, number of training examples and prior knowledge. Optimisation of one or two of these objectives often results in a sacrifice of the third. Learning a complex behaviour in an unstructured environment without prior knowledge requires a long exploration and training phase and therefore creates a serious problem to robotic applications. Today's robots are faced with imprecise, uncertain, and randomly changing environments. The desire to deal with these environments leads to the basic premises and the guiding principles of soft computing.

Robot control is predominately motion control using classical servomechanism control theory. Due to the nonlinearity of the manipulator motion, a wide variety of control schemes have been derived. Classical schemes include computed torque, resolved motion, PID decoupled model control, reference adaptive and resolved motion adaptive control (Whitney, 1969), (Begczy, 1974), (Dubowsky & DesForges, 1979). These schemes can be very complicated and require intensive computer resources. For instance, the computer torque technique uses the Lagrange–Euler or Newton–Euler equations of motion of the manipulator to determine the required torque to servo each joint in real time to track the desired trajectory as closely as possible. However, since there are always uncertainties in the robot dynamic model, the ideal error response cannot be achieved and the performance could be well degraded. This problem led people to using adaptive control approaches to solve these problems and relatively good results were obtained (Craig et al, 1987), (Spong & Ortega, 1988). The problem is complicated if we think to enlarge the workspace of the manipulator by mounting over it a mobile platform, resulting on a new system called a mobile manipulator. Researches to investigate the capabilities of mobile platforms with onboard manipulators are devoting considerable effort to come up with solutions to this complicated system (Yamamoto & Yun, 1994).

Now, since the first control a plication of Mamdani (Mamdani & Assilian 1974) and his team, a lot of efforts have been devoted to capitalizing on the advances of fuzzy logic theory. Many fuzzy control approaches appeared. In fact, fuzzy logic provides tools that are of potential interest to control systems. Fuzzy controllers are a convenient choice when an analytical model of the system to be controlled cannot be obtained. They have shown a good degree of robustness in face of large variability and uncertainty in the parameters, and they lend themselves to efficient implementations, including hardware solutions. These characteristics fit well the needs to precision motion control of mobile manipulators. However, the main difficulty in designing a fuzzy logic controller is the efficient formul tion of the fuzzy **If-Then** rules. It is well known that it is easy to produce the antecedent parts of a fuzzy control rules, but it is very difficult to produce the consequent parts without expert knowledge. The derivation of such rules is often based on the e perience of skilled operators, or using heuristic thinking (Zadeh, L.A.1973), (Mamdani, E.H. 1974). In recent years and due to the availability of powerful computer platform, the theory of evolutionary algorithms starts to become popular to the problem of parameter optimization. Genetic algorithm as one approach to the implementation of evolutionary algorithms was used by Karr, (Karr, C.L. (1991) to generating the rules of the cart-pole balancing fuzzy logic controller. In this work, we investigate the problem of the motion control of a mobile manipulator using fuzzy control schemes. The m chanical system is split into two subsystems where the mobile platform and the manipulator constitute the parts. Appropriate fuzzy controllers are used to control each of these two subsystems. A genetic algorithm generates the rules of the fuzzy controllers letting the system turning around an optimal solution. The motion of the platform and that of the manipulator are coordinated by a Neural like network, wich is a sort of adaptive graph of operations, designed from the kinematics model of the system. A learning paradigm is used to produce the required reference variables for each of the mobile platform and the robot manipulator for an overall coordinate behaviour.

3. Robot Model

3.1 A mobile manipulator overview architecture

A mobile manipulator system is a robotic manipulator mounted on mobile platform.This combination allows manipulation tasks over unlimited working space. However, since the platform and the manipulator have independent movement, a particular point in the workspace may be reached in multiple configurations, resulting in a system with redundancy (Lee, J. K., & Cho, H. S.

1997). This can be helpful when it is desirable to perform tasks in a cluttered environment, or to optimally configure the system (Brock, O., Khatib, O. & Viji, S. 2002). Our o jective in this work, is to devise a controller for each of the mobile base and the manipulator separately, then we implement a sort of adaptive graph of operations to generate trajectory in the joint space. The net-work provides reference output values of the desired motion to the mobile manipulator system. The mechanical system is made up of the non-holonomic platform upon which is mounted a robot manipulator with 3 rotational de-grees of freedom as it is shown in Figure 1. The accomplishment of the task is the result of the perm nent movement of the two structures for which the suc-cess is based on the satisfaction of the tracking error. If we consider Figure 1 where the four principal coordinate frames are shown: World frame O_W, plat-form frame O_P, manipulator base frame O_B, and the end effector frame O_E. Then, the manipulator's end effector position/orientation with respect to O_W is given by:

$$T_E^W = T_P^W T_B^P T_E^B$$

Such that the matrix T_P^W is determined by a certain $A(q)$ matrix, T_B^W is a fixed matrix and T_E^B is determined by the joint variable vector $\theta = \left[\theta_1, \theta_2, ..., \theta_{n_m}\right]^T$, n_m represents the degree of freedom of the arm manipulator.
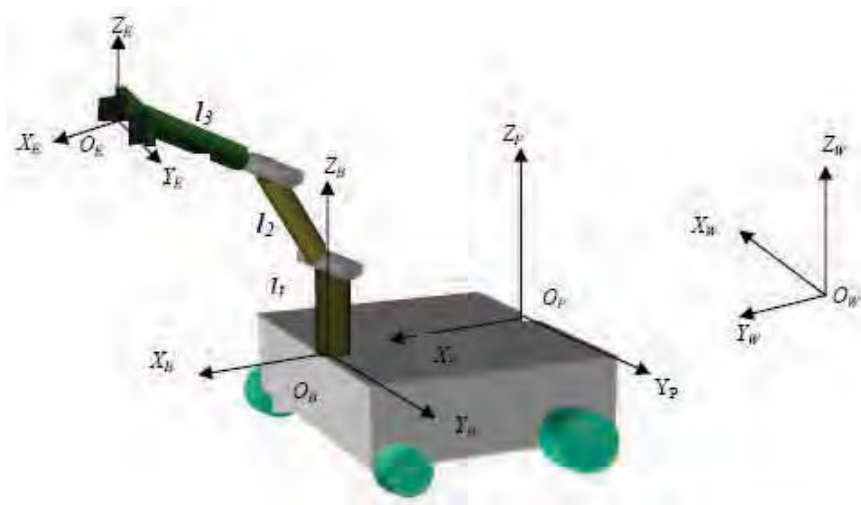
Figure 1. Mobile manipulator configuration

The vector of position of the end effector x_E^W is a non-linear function of the configuration vector, $q = \left[p^T, \theta^T \right]^T \in \Re^n, (n = 3 + n_m)$. The joint coordinates of the manipulator are $\theta = \left[\theta_1, \theta_2, \theta_3 \right]^T$ (thus $(n_m = 3)$. Therefore the generalized coordinates of the mechanical system are:

$$q = (q_1, q_2, ..., q_6)^T = (x_B, y_B, z_B, \theta_1, \theta_2, \theta_3)^T$$

Hence, the generalized space dimension of the mechanical system is equal to $\lambda = 6$. Now, for a given mechanical configuration system q, its structure imposes to its end effector η position and orientation constraints. In our case, only the end effector position is considered. Therefore, the number of constraints is r duced to $\sigma = 3$. On the other hand, we can observe that the system is non holonomic, and taking into account the constraint of the non-holonomy of the mobile platform, we can deduce the order of redundancy, which is equal to $(\lambda\ \sigma - 1) = 2$. This redundancy helps increasing the manipulator dexterity, prevents the arm from singular configurations, and the let the system away from obstacles while completing a given task. On the other hand, the control of such mechanisms becomes much harder.
If we refer to Figure 1 and following the *D-H* parameterization, the outputs of the neural like network are given by equations (1), which designates the Cartesian coordinates of the task variable *E*, with respect to the world frame {*W*}. In a closed form this can be written as $X_E(t) = F(q(t))$; where *F* represents the direct kinematic mapping from the joint space to the task space and
$X_E(t) = (x_E^W, x_E^W, x_E^W)^T$.

$$
\begin{aligned}
x_E^W &= x_B^W + \cos(\theta). \left[l_2 \cos(\theta_2) + l_3 \cos(\theta_2 + \theta_3) \right] \\
y_E^W &= y_B^W + \sin(\theta) . \left[l_2 \cos(\theta_2) + l_3 \cos(\theta_2 + \theta_3) \right] \\
z_E^W &= z_B^W + l_1 - l_2 \sin(\theta_2) - l_3 \sin(\theta_2 + \theta_3)
\end{aligned}
\tag{1}
$$

Such that,

$$\theta = \theta_1 + \varphi \tag{2}$$

Where φ is the heading angle of the mobile platform, and l_1, l_2 and l_3 are the lengths of the three links composing the manipulator arm. x_B^W, y_B^W, z_B^W, are the coordinates of the point *B* located in the front of the mobile platform with respect to the world frame $\{W\}$. In the sequel, we consider z_B^W equals zero for

simplicity. The goal is to find the generalized trajectory $q(t)$ for a given task space trajectory $X_E(t)$ such that $F(q(t)) = X_E(t)$ is satisfied. Since the system is redundant, the number of solutions is expected to be infinite. To realize a generalized task of the mechanical system, one has to derive the set of the λ generalized coordinates. In this context, an approach is suggested to investigate and solve this problem when we make a complete motion of the end effector resulting from a combined operation of the two subsystems that work in a coordinate manner.

3.2 The dynamic model of the manipulator

Two main approaches are used by most researchers to systematically derive the dynamic model of a manipulator, the Lagrange-Euler and the Newton-Euler formulations. The dynamic equations of motion are highly nonlinear and consist of inertia loading, coupling reaction forces between joints and gravity loading effects. For an n-link rigid manipulator the vector dynamic equation is given by:

$$\tau = M(\theta)\ddot{\theta} + B(\theta,\dot{\theta})\dot{\theta} + F(\theta,\dot{\theta}) + G(\theta) \tag{3}$$

where $\theta \in R^n$ is a vector of joint displacements, $\tau \in R^n$ is a vector of applied joint torques, $M(\theta): R^n \to R^{n \times n}$ is a symmetric positive definite manipulator inertia matrix, $B(\theta,\dot{\theta}): R^n \times R^n \times R^n \to R^n$ is a vector of centrifugal and Coriolis terms, $F(\theta,\dot{\theta}): R^n \times R^n \to R^n$ is a vector of frictional torques, and $G(\theta): R^n \to R^n$ is a vector of gravitational torques. The control of the robot manipulator is especially challenging due to the generic high nonlinearity existing in its dynamic model. Although the equations of motion (3) are complex and nonlinear for all but simple robots, they have several fundamental properties, which can be exploited to facilitate control system design.

1. **Property 1**. The inertia matrix $M(\theta)$ is symmetric, positive definite and both $M(\theta)$ and $M^{-1}(\theta)$ are uniformly bounded as a function θ of R^n.

2. **Property 2**. There is an independent control input for each degree of freedom

3. **Property 3**. The Lagrange-Euler equations for the robot are linear in the parameters.

Most feedback control laws, where a PD or PID controllers are used, are based on simplified dynamic equations. However the approach works well only at slow speeds of movement. At high speeds of movement the Coriolis and cen-

trifugal forces are major components of the dynamic equations and conse-
quently the error can not corrected. A lot of effort has been devoted to capital-
izing on the advances in mathematical control theory resulting in several tech-
niques appeared to tackle this kind of mechanical systems. May be the most
famous and which is considered, as the basic approach becoming very popular
is the model based computed torque method. However, since there are always
uncertainties in the robot dynamic model and the disturbances possibly aris-
ing from the actual running of the actuator or some other causes, the ideal er-
ror response cannot be achieved and the performance could be well degraded.
Now, since the reliability of the PID controller has been field proven besides
the application of fuzzy logic theory to process control, we propose in the next
section a combination of the two to make a robust controller for robot manipu-
lators.

3.3 The Kinematic model of the mobile platform

In this section, a kinematic description of a mobile robot is given. The vehicle
has two driving wheels at the rear corners and two passive supporting wheels
at the front corners. Two DC motors independently drive the two rear wheels.
The vehicle presents however two constraints: It is non-holonomic, which
means that it must move in the direction of the axis of symmetry, i.e.

$$\dot{y}_A - \dot{x}_A \tan\phi = 0 \tag{4}$$

ϕ is the heading angle of the vehicle from the X-axis of the world coordinates
as it is depicted in Figure 2.
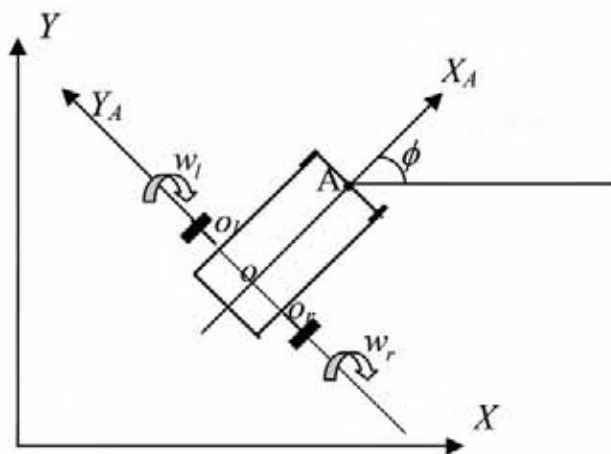
Figure 2. Mobile robot schematic

Writing the classical relationships between the velocity point O, and those of points O_l and O_r, we can easily determine the linear velocity $\vec{v}_o$, and the instantaneous angular velocity $\vec{\omega}$ of the mobile robot:

$$\vec{v}_o = \vec{v}_{or} + o\vec{o}_r \wedge \vec{\omega} \tag{5}$$

$$\vec{v}_o = \vec{v}_{ol} + o\vec{o}_l \wedge \vec{\omega} \tag{6}$$

$$\vec{\omega} = \dot{\phi}.\hat{k} \tag{7}$$

$\hat{k}$ is the unit vector along the Z_A axis; and $\vec{v}_{ol}$ and $\vec{v}_{or}$ are the linear velocities of the mobile robot points O_l and O_r respectively. When projecting expressions (5) and (6) on the X-axis and the Z-axis, we get the expressions of v_o and $\dot{\phi}$ as follows:

$$v_o = \frac{r}{2}(\omega_r + \omega_l) \tag{8}$$

$$\dot{\phi} = \frac{r}{2R}(\omega_r - \omega_l) \tag{9}$$

r and R are respectively the radius of the wheels and the width of the vehicle as it is shown in Figure 3. It has been proven by Samsung and Abderrahim (Samsung, C. & Abderrahim, K.A. 1990), that the vehicle converges better to its reference when controlling a point located in front of the rear wheel axis.
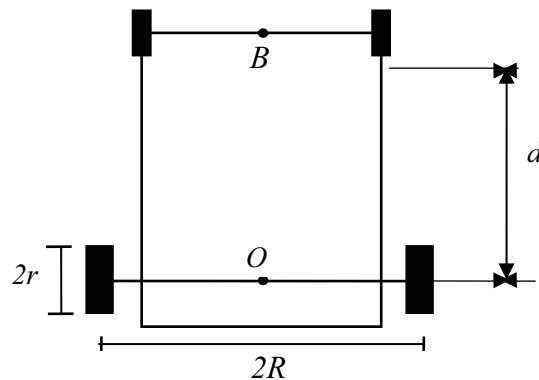
Figure 3. Geometric characteristic of the mobile robot

In this paper point B, as it is obvious from Figure 3, which is located at a distance d from point O, has been chosen to be the position control of the vehicle such that:

$$x_B = x_o + d.\cos\phi \tag{10}$$

$$y_B = y_o + d.\sin\phi \tag{11}$$

where:

$$x_o(t+1) = x_o(t) + \Delta D.\cos(\phi + \frac{\Delta\phi}{2}) \tag{12}$$

$$y_o(t+1) = y_o(t) + \Delta D.\sin(\phi + \frac{\Delta\phi}{2}) \tag{13}$$

Such that:

$$\Delta D = \frac{r}{2}(\Delta q_r + \Delta q_l) \tag{14}$$

$$\Delta\phi = \frac{r}{2R}(\Delta q_r - \Delta q_l) \tag{15}$$

Where, (x_o, y_o) and (x_B, y_B) denote the coordinates of points O and A respectively, whereas Δq_r and Δq_l are the angular steps of the right and left wheels respectively.

4. Robot Control

The control strategy combines the mobile base behaviour and the manipulator behaviour to produce an integrated system that performs a coordinated motion and manipulation. We propose in this section the two layer robot controller and the genetic algorithm to determine the solution that gives the optimum rule base for a precompensator which is associated with the PID controller (Abdessemed, F. & Benmahammed, K. 2001).

4.1 The two layer robot controller design

Control inputs to the joints are composed of both feedback PID control and precompensator sub-systems components, (Fig. 4). The output of the precom-

pensator is considered as a new reference input to the PID-plant system. The introduction of the precompensator is justified by the fact that when the system evolves toward an abnormal mode, it is necessary to anticipate this evolution rather than to wait to arrive to this mode in order to avoid its consequences especially if it is dangerous. The dynamics of the precompensator-PID controller is explained as follows: The two inputs to the PID controller are $e_i^{'}(k)$ and $\dot{e}(k)$;

Where:

$$e_i^{'}(k) = \theta_i^c(k) - \theta_i(k) \tag{16}$$

$$\dot{e}_i(k) = \dot{\theta}_i^d(k) - \dot{\theta}_i(k); \qquad \text{i=1,2 ,3 refer to the ith link.} \tag{17}$$

Note that the desired angular position is not directly compared to the measured one, but passes first through the precompensator to be transformed to a new reference angular value for the PID-plant system. Thus, one writes:

where:

$$\theta_i^c(k) = \theta_i^d(k) + m_i(k); \quad \text{i=1,2 and 3} \tag{18}$$

$$m_i = F_i (e_i , \Delta e_i); \quad \text{i=1,2 and 3.} \tag{19}$$

$e(k)$ and $\Delta e(k)$ are inputs to the map F, and $m_i(k)$ is the output of the i-th joint; such that:

$$e_i(k) = \theta_i^d(k) - \theta_i(k); \quad \text{i=1, 2 and 3} \tag{20}$$

$$\Delta e_i(k) = e_i(k) - e_i(k-1); \quad \text{i=1,2 and 3} \tag{21}$$
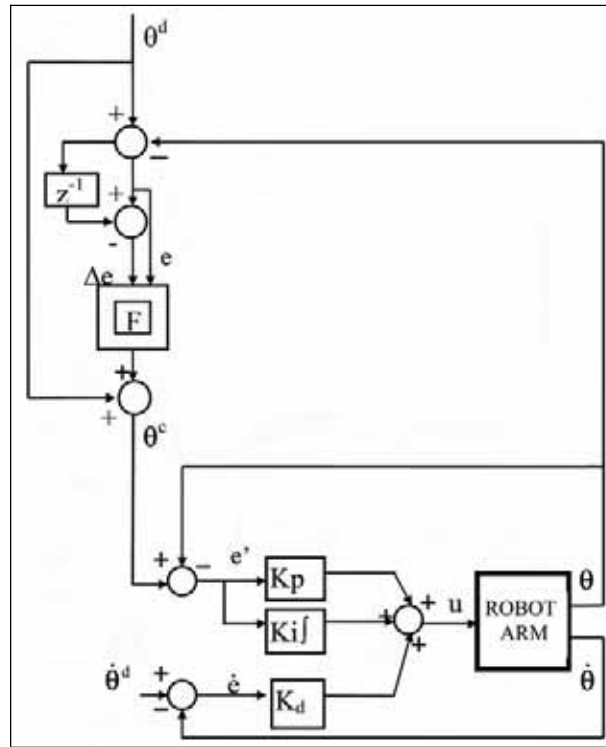
Figure 4. Diagram simulation of one-link robot control

4.1.1 The PID controller parameters determination

As a first attempt to regulate the robot manipulator, we consider the proportional-integral-derivative (PID) control law given by

$$\tau = K_P e(t) + K_D \dot{e}(t) + K_I \int e(t) dt \tag{22}$$

Where $e(t) = \theta^d - \theta$ and θ^d and θ are the desired reference and the measured trajectories respectively. The required controller parameters are found based on simplified dynamic equations; i.e, the manipulator is supposed evolving at slow speeds of movement. Consequently, the Coriolis and centrifugal forces are of the dynamic equations are neglected, thus

$$\tau = M(\theta)\ddot{\theta} \tag{23}$$

Considering equations (22) and (23), the following transfer function is obtained:

$$\frac{\theta_i}{\theta_i^d} = \frac{k_{Di}s^2 + k_{Pi}s + k_{Ii}}{J_i s^3 + k_{Di}s^2 + k_{Pi}s + k_{Ii}} \tag{24}$$

Thus, the characteristic equation is written as:

$$\Delta(s) = s^3 + \frac{k_{Di}}{J_i} s^2 + \frac{k_{P\,i}}{J_i} s + \frac{K_{Ii}}{J_i} = 0 \tag{25}$$

J_i is taken to be the fixed term of the J_{ii} element of the inertial matrix J. In robotic control, the gains are typically chosen so that the two poles are double and real negative. This gives a damping coefficient of one and by consequence a fast response without oscillations. The remaining pole is then placed on the real axis far away from the first two. Thus:

$$\Delta(s) = (s+\beta)2(s+n\beta). \tag{26}$$

The desired gains are given by the following relationships:

$$\begin{aligned}
k_{Pi} &= m_i(2n+1)\beta^2, \\
k_{Di} &= m_i(2+n)\beta, \\
k_{Ii} &= m_i n\beta^3
\end{aligned} \tag{27}$$

with: $\beta>0,\ n>1$. The choice of β depends on the sampling frequency as well as to possible saturation effects.

4.1.2 The precompensator design

The precompensator is a fuzzy controller. Its main goal is to enter into action whenever it is needed to reinforce the conventional controller to provide the necessary commands that allow the end effector to track the desired trajectory with minimum error. A fuzzy controller is a system, which use a rule-based expert in the form of **If Then** statements. The input state with respect to a certain universe of discourse constitutes the premise of the rule, whereas the output state constitutes the consequence of the rule. We can distinguish among the steps of the rule treatment, the following procedures: *Fuzzification*, *Fuzzy inference* and *Defuzzification*. The main difficulty in designing a fuzzy logic controller is the efficient formulation of the fuzzy **If-Then** rules. It is well known that it is easy to produce the antecedent parts of a fuzzy control rules, but it is very difficult to produce the consequent parts without expert knowledge. In this work, the fuzzy rule base of the precompensator designed is found by using a genetic algorithm that search for the solution that gives the optimum rule base for the precompensator. If we assume that the error and its derivative are partitioned into K and L subsets then, the the number of possible combinations is LxK, which represent the number of rules per output. A single rule is defined as:

Rule R_{ij} :   *if e is A_i^e and Δe is $A_j^{\Delta e}$ then m_{ij} is A_p^m* $\quad$ $i=1, 2, ...\ 7; j=1, 2, ..., 7; p=1, 2, ..., 7.$

R_{ij} is the label of he fuzzy if-then rule, A_i^e and $A_j^{\Delta e}$ are fuzzy subsets on the interval *[-1, 1]*, and A_p^m is the consequent fuzzy subset. The aim is to sort out the appropriate optimised rules adequate for the system by employing an evolving genetic algorithm. The system being evolved is encoded into a long valued string called a chromosome. Initially a random population of these strings is generated. Each string is then evaluated according to a given performance criterion and assigned a fitness score. The strings with the best score are used in the reproduction phase to give the next generation. Here, the genetic algorithm is presented as a seven-tuple entity and is abstracted in the following encapsulated form:

GA={M(t), l, D, $\boldsymbol{\Phi_F}$, sel, pcross, pmut } $\hspace{6cm}$ (28)

where:

- $\quad$ **M(t) = {m_0, ... ,m_6}^{l}** $\hspace{2.5cm}$ encoding chromosome ($-1 \le m_i \in \mathfrak{R} \le 1$).
- $\quad$ **$l \in \aleph$** $\hspace{5.3cm}$ length of chromosome.
- $\quad$ **$D \in \aleph$** $\hspace{5.2cm}$ population size.
- $\quad$ **$\boldsymbol{\Phi_F} : M \rightarrow \mathfrak{R}$** $\hspace{4.2cm}$ fitness function.
- $\quad$ **sel** : $Crom^D \rightarrow Crom$ $\hspace{2.6cm}$ parent selection operation
- $\quad$ **pcross**: $Crom^2 \rightarrow Crom^2$ crossover operation.
- $\quad$ **pmut** : $Crom \rightarrow Crom$ $\hspace{2.5cm}$ mutation operation.

Each individual chromosome represents a complete rule base solution formulated as a set *M* of the generated *KxL* fuzzy if-then rules such that:

M^i={ m_i^j | i=1, ..., D; j=1, ..., l} $\hspace{5cm}$ (29)

The set of all the individuals represents a population. If we denote by *P(t)* a population at a time *t,* then we can write:

$$P(t) = \left\{ M^1(t),\ M^2(t),\ ... , M^D(t) \right\} \hspace{4cm} (30)$$

Where $m_i^j \in S_m \in \mathfrak{R}$, are the *j-th* consequent parts of the fuzzy rules of the *i-th* individual. It takes its value from the set S_m=*{-1, -0.66, -0.33, 0.0, 0.33, 0.66, 1}*. These values correspond to the projection of the peaks of the membership functions on the normalized universe of discourse of the action.

4.2 Mobile platform fuzzy control design

If we consider the vehicle moving in a free obstacle environment, then the optimal trajectory from its current position to its end configuration is naturally a line joining these two extreme points as it is shown for instance by Figure 5, where θ is the angle between the symmetric axis of robot and the line that joins the control point of the robot to its final point.
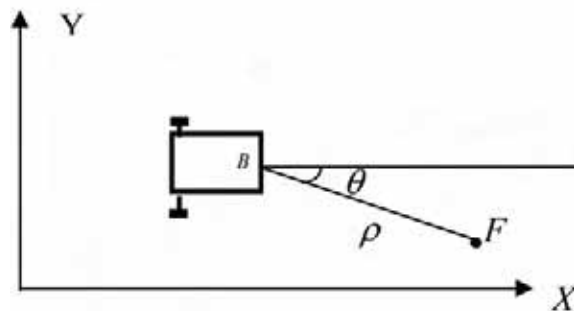
Figure 5. Example of situation " Reaching a point "

If we link the points with segments, then the goal is to control the driving point A of the autonomous robot with respect to these segments and to come the closest possible to the end point. The distance ρ becomes zero when the vehicle stabilizes at its final configuration. Figure 6 gives a schematic block diagram of this architecture. From this figure one can notice that the inputs to the fuzzy controller are ρ and θ, and its output is the steering angle γ. (Abdessemed, F. & Benmahammed, K. 2004)
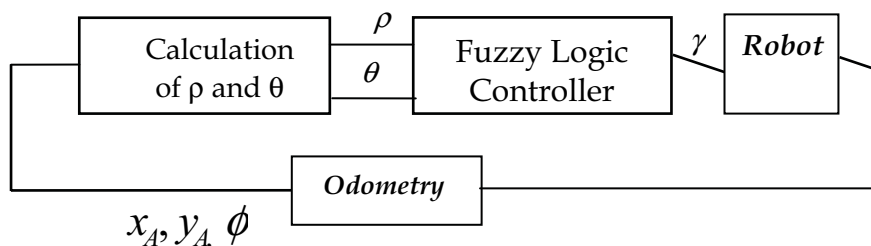
Figure 6. Block diagram of the controlled system

The best fuzzy system is implemented with five and eight triangular membership functions for the controller input variable ρ and θ respectively. The second step in designing an *FLC* is the fuzzy inference mechanism. For instance, the knowledge base of the system consists of rules in the form:

Rule R_{ij} :   IF ρ is S AND θ is NM THEN γ is n_p $i=1, 2, ... N; j=1, 2, ..., K; p=1, 2, ..., L.$

Such that:

$$\gamma(k)=F[\rho(k),\theta(k)] \tag{31}$$

$\rho(k)$ and $\theta(k)$ are inputs to the map F, and the output $\gamma(k)$ denotes a numerical value within the interval $[-90°, +90°]$, characterizing the relative variation in the direction that should be taken by the vehicle to reach the final point. The rules could be defined by using the human-like description of the vehicle's movement behaviour. But, this approximate human reasoning may lead to certain unsatisfactory rules. Furthermore, the global behaviour of the vehicle may result from the combination of several basic behaviours for instance, the trajectory tracking, the vehicle speed, and the obstacle avoidance. As a matter of fact, it is not obvious to define the respective rules. Therefore, and following the same approach described in the last section, we propose an evolutionary algorithm as an efficient solution for the extraction of the consequent part of the rules. A $(\mu+\lambda)$-evolutionary programming is described by an octuple entity defined by the following format:

$$EP=\{I(t), L, \mu, \lambda, sel, pmut, f, g\} \tag{32}$$

For which the components are defined as follows:

- I= $[a_1, a_2, \dots , a_{2L}]$ Encoding chromosome
- $2L \in \aleph$ Length of chromosome
- $\mu \in \aleph$ Population size
- $\lambda \in \aleph$ Number of offspring = μ
- pmut: $I \rightarrow I$ mutation operator
- f: $\Re L \rightarrow \Re$ fitness function
- g: $\Re L \rightarrow \Re$ set of constraints

The design of the *EP* is based mainly on three mechanisms:

- The representation of individuals,
- Implication of the variation operators,
- The generation procedure.

Each individual chromosome represents a complete rule base solution. The components: $(a_1 = m_1,\ a_2 = m_2,\ \ldots,\ a_L = m_L)$ determine the consequent part of the fuzzy rules and the remaining components, $(a_{L+1} = \sigma_1,\ a_{L+2} = \sigma_2,\ \ldots,\ a_{2L} = \sigma_L)$ contain the standard deviation, which controls the mutation process. A complete string of chromosome could be written in the following way: $a_1 a_2 \ldots a_L a_{L+1} a_{L+2} \ldots a_{2L}$, representing one individual. The set of all the individuals represent a population. If we denote by *P(k)* a population at a time *k*, then we can write:

$$P(k) = \bigcup_{i=1\ldots\mu} I^i(k) \ \text{ with } I^i = \left\{ a^i_j,\ {}_{i=1,\ldots,\mu,\ j=1,\ldots,2L} \right\} \tag{33}$$

Where I^i designates the *i-th* individual in which the components a_j describe the consequent parts of the rules and the standard deviations. In this application, we have rather chosen the floating point encoding instead of the binary code. The algorithm seeks many local optima and increases the likelihood of finding the global optimum, representing the problem goal.

5. Motion Control of a Mobile Manipulator

The control strategy combines the mobile base behavior and the manipulator behavior to produce an integrated system that performs a coordinated motion and manipulation. If we refer to the arm manipulator by *agent1* and the mobile platform by *agent2*, then the architecture shown by Figure 7 illustrates the actions on the environment by the two defined agents.
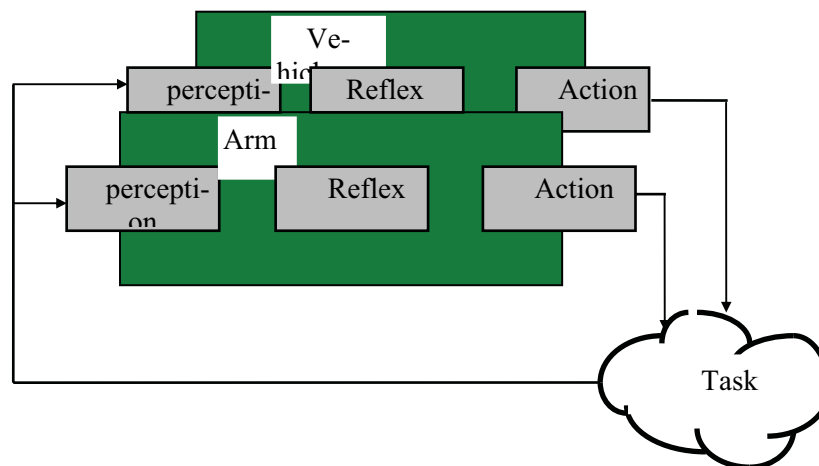
Figure 7. Configuration of the coordinated motion and manipulation of the robotic system architecture

To provide a solution to the mobile manipulation motion, we have arranged the direct geometric model equations (1) into a sort of adaptive graph of operation made up of three layers, and which we have called a neural-like-network (Fig. 8). Each layer has a number of transfer functions each of which is defined by the expressions given by equations (34). This neural-like-network is the kernel of our proposal; it is very interesting and uncommon in robot trajectory generation (F. Abdessemed, *et al*. 2006). In this case, the two mechanical stru tures are considered as a unique entity. This arrangement will facilitate the implementation of the back propagation algorithm as a learning rule to adapt the weights so that the output values of the neural-like-network come close to the desired reference values describing the task space trajectory. The accomplishment of the task is the result of the permanent movement of the two structures for which the success is based on the satisfaction of the tracking error. Figure.8 illustrates the model architecture of this combined structure. For convenience we define *x31, x32, x33* as the outputs of the network, and which designates the Cartesian coordinates of the task variable $E : x_E^W, y_E^W$ and z_E^W respectively. Where:

$$f_{11}(\theta,\ w_{11}) = x_{11} = \cos(w_{11}\theta)$$
$$f_{12}(\theta_2,\ w_{22}) = x_{12} = \cos(w_{22}\theta_2)$$
$$f_{13}(x_{12}) = x_{13} = \cos^{-1}(x_{12})$$
$$f_{14}(x_{13},\theta_3,\ w_{33}) = x_{14} = (x_{13} + w_{33}\theta_3)$$
$$f_{21}(x_{11}) = x_{21} = \sin(\cos^{-1}(x_{11}))$$
$$f_{22}(x_{12},\ x_{14}) = x_{22} = l_3\cos(x_{14}) + l_2 x_{12} \qquad (34)$$
$$f_{23}(x_{13}) = x_{23} = l_2\sin(x_{13})$$
$$f_{24}(x_{14}) = x_{24} = l_3\sin(x_{14})$$
$$f_{31}(x_{11},\ x_{22}) = x_{31} = x_{11}.x_{22} + b_1 x_B$$
$$f_{32}(x_{21},\ x_{22}) = x_{32} = x_{21}.x_{22} + b_2 y_B$$
$$f_{33}(x_{23},\ x_{24}) = x_{33} = l_1 - x_{23} - x_{24}$$
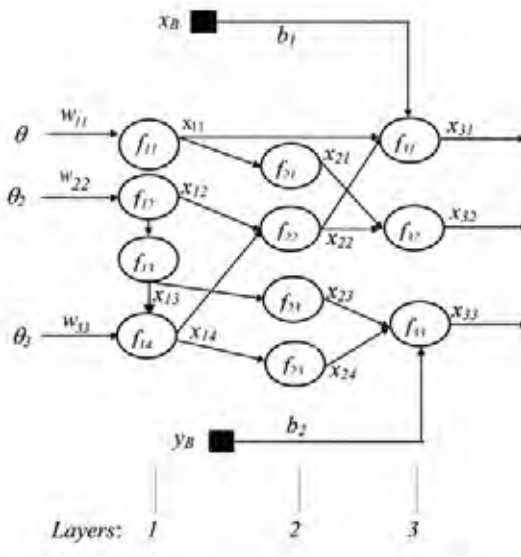
Figure 8. The adaptive graph of operations

For convenience we define x_{ij} as the outputs of the network, and which x_{31}, x_{32}, x_{33} are the outputs of the network, and wich designates the Cartesian coordinates of the task variable $E.(x_E^W, y_E^W z_E^W)$. Let $q(k)$ be the input vector, such that $q^T(k) = [\theta, \theta_2, \theta_3, x_B, y_B]$ and $^m X_E^W$ the measured output vector such that $X_E^W(k) = [^m x_E^W, ^m y_E^W, ^m z_E^W]^T,$, and the weighting vector W such that $W^T(k) = [w_{11}, w_{22}, w_{33}]$. If we set the criterion E_p be the tracking error given by equation (35), then the control objective is to design a control law, which guaranties: E_p to go zero when k goes to infinity,, k is the running time.

$$E_p = \sum_{k=1}^{N(3)} (x_{3k} - r_k)^2 = \left[(x_{31} - r_1)^2 + (x_{32} - r_2)^2 + (x_{33} - r_3)^2 \right] \tag{35}$$

Where

$^d X_E^W = (r_1, r_2, r_3)^T = (^d x_E^W, ^d y_E^W, ^d z_E^W)^T$, represent the desired operational coordinates and $^m X_E^W = (x_{31}, x_{32}, x_{33})^T = (^m x_E^W, ^m y_E^W, ^m z_E^W)^T$ the operational measured coordinates in the world frame. The effect of adjusting the weighting vector W to the error Ep is determined by the ordered derivatives $\partial^+ E_p / \partial W(k)$ (Werbos, 1974). Now, we apply the back-propagation learning rule to generate the appropriate parameter-weighting vector $W(k)$ (Rumelhart $et\ al$, 1986). Once determined, the weights are used to update the input vector q. The elements of this vector will serve as inputs to the low level controllers of the two agents as

illustrated by the block diagram of Fig. 9. The reference states of the plant at time *k+1* are functions of the reference states at time *k* and the computed weights at time *k+1*, and can be expressed symbolically as

$$q(k+1) = \Psi(q(k), W(k+1)) \tag{36}$$
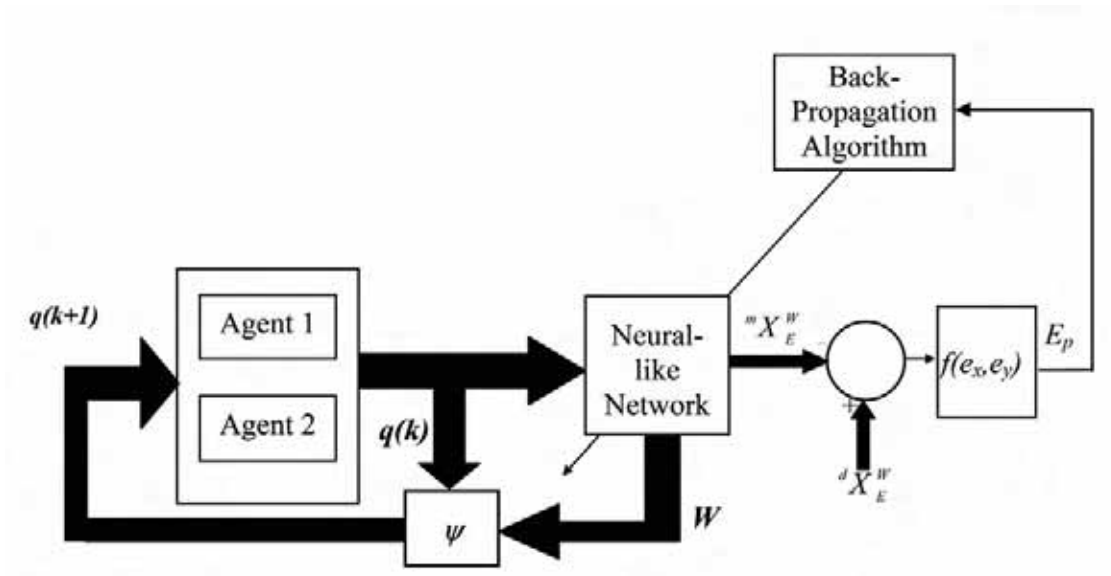
Figure 9. Configuration of the controlled system including the adaptive graph of operations

6. Back-Propagation Learning Rule

6.1 Output Layer

The error signal for the *j-th* output node can be calculated directly:

$$\varepsilon_{3,i} = \frac{\partial^+ E_p}{\partial x_{3,i}} = \frac{\partial E_p}{\partial x_{3,i}} \tag{37}$$

Therefore

$$\varepsilon_{31} = 2(x_e^d - x_{31}), \ \varepsilon_{32} = 2(y_e^d - x_{32}), \ \varepsilon_{33} = 2(z_e^d - x_{33})$$

6.2 Internal Layers

The error signals of these internal nodes at the *j-th* position are calculated using the following equation

$$
\varepsilon_{l,i} = \underbrace{\frac{\partial^+ E_p}{\partial x_{l,i}}}_{\substack{\text{Error} \\ \text{Signal of} \\ \text{layer l}}} = \sum_{m=1}^{N(l+1)} \underbrace{\frac{\partial^+ E_p}{\partial x_{l+1,m}}}_{\substack{\text{Error} \\ \text{Signal of} \\ \text{layer l+1}}} \times \frac{\partial f_{l+1,m}}{\partial x_{l,i}}
\tag{38}
$$

$$
\varepsilon_{l,i} = \sum_{m=1}^{N(l+1)} \varepsilon_{l+1,m} \frac{\partial f_{l+1,m}}{\partial x_{l,i}}
\tag{39}
$$

Such that: $0 \leq l \leq L-1$

$$
\varepsilon_{2,j} = \sum_{m=1}^{2} \varepsilon_{3,m} \frac{\partial f_{3,m}}{\partial x_{2,j}}. \qquad j = 1,2.
\tag{40}
$$

Therefore the error signals of the nodes at the internal layer are as follows:

$$
\varepsilon_{2,1} = \varepsilon_{3,2}.x_{2,2}, \quad \varepsilon_{2,2} = \varepsilon_{3,1}.x_{1,1} + \varepsilon_{3,2}.x_{2,1}, \quad \varepsilon_{2,3} = -\varepsilon_{33}, \quad \varepsilon_{2,4} = -\varepsilon_{3,3}
$$

6.3 Input Layer

The first layer contains four neurons arranged in the way presented by Figure 8. The general form for the error signal is given by the following equation:

$$
\varepsilon_{1,i} = \sum_{m=1}^{4} \varepsilon_{2,m} \frac{\partial f_{2,m}}{\partial x_{1,i}}
\tag{41}
$$

Explicitly the error signals are found to be

$$
\varepsilon_{1,1} = -\varepsilon_{2,1} \frac{x_{1,1}}{\sqrt{1-x_{1,1}^2}} + \varepsilon_{3,1}.x_{2,2}
\tag{42}
$$

$$
\varepsilon_{1,2} = -\varepsilon_{2,2}.l_2 - \varepsilon_{1,3} \frac{1}{\sqrt{1-x_{1,2}^2}}
\tag{43}
$$

$$\varepsilon_{1,3} = \varepsilon_{2,3}.l_2\cos(x_{1,3}) + \varepsilon_{1,4} \tag{44}$$

$$\varepsilon_{1,4} = l_3\left[\varepsilon_{2,4}.\cos(x_{1,4}) - \varepsilon_{2,2}\sin(x_{1,4})\right] \tag{45}$$

6.4 Weight adjustment

To adjust the weights we make use of the following known equation:

$$w_{ij}^l(k+1) = w_{ij}^l(k) - \mu\frac{\partial E_p}{\partial w_{ij}^l(k)}\bigg|_{w_{ij}^l(k)} \tag{46}$$

Where

$$\frac{\partial^+ E_p}{\partial w} = \frac{\partial^+ E_p}{\partial x_{l,i}}\frac{\partial^+ f_{l,i}}{\partial w} = \varepsilon_{l,i}\frac{\partial f_{l,i}}{\partial w} \tag{47}$$

Therefore the weights are updated according to the following resulting equations:

$$w_{11}(k+1) = w_{11}(k) + \eta\varepsilon_{1,1}.\theta.\sin(w_{11}.\theta) \tag{48}$$

$$w_{22}(k+1) = w_{22}(k) + \eta\varepsilon_{1,2}.\theta_2\sin(w_{22}\theta_2) \tag{49}$$

$$w_{33}(k+1) = w_{33}(k) - \eta\theta_3\varepsilon_{1,4} \tag{50}$$

$$b_1(k+1) = b_1(k) - \eta x_B\varepsilon_{3,1} \tag{51}$$

$$b_2(k+1) = b_2(k) - \eta y_B\varepsilon_{3,2} \tag{52}$$

Where, the two last equations represent the updates of the biases b_1 and b_2. Nevertheless, the steepest descent algorithm is slower for on-line applications. For that reason, it is rather advisable to use the Levenberg-Marquardt algorithm, which has proved to be an effective way to accelerate the convergence rate. Its principal advantage is that it uses information about the first and second derivatives and does not need to invert the Hessian matrix.

$$x_{k+1} = x_k - \left[J^T J + \mu I\right]^{-1} J^T\varepsilon \tag{53}$$

The weights are updated on each iteration by the following found expressions:

$$w_{11}(k+1) = w_{11}(k) - \frac{j_{11}}{j_{11}^2 + \mu}\varepsilon_{1,1}$$

$$w_{22}(k+1) = w_{22}(k) - \frac{j_{22}}{j_{22}^2 + \mu}\varepsilon_{1,2}$$

$$w_{33}(k+1) = w_{33}(k) - \frac{j_{33}}{j_{33}^2 + \mu}\varepsilon_{1,4}$$

$$b_1(k+1) = b_1(k) - \frac{j_{44}}{j_{44}^2 + \mu}\varepsilon_{3,1}$$

$$b_2(k+1) = b_2(k) - \frac{j_{55}}{j_{55}^2 + \mu}\varepsilon_{3,2}$$

$$(54)$$

Where

$$j_{11} = -(\theta_1 + \varphi)\sin(w_{11}(\theta_1 + \varphi))$$
$$j_{22} = -\theta_2 \sin(w_{22}\theta_2)$$
$$j_{33} = \theta_3$$
$$j_{44} = x_B$$
$$j_{55} = y_B$$

$$(55)$$

Where, *I* is the identity matrix. At this end stage, we give the reference state variables at time *k+1* and which are represented by the following expressions:

$$\theta(k+1) = w_{11}(k+1)\theta(k)$$
$$\theta_2(k+1) = w_{22}(k+1)\theta_2(k)$$
$$\theta_3(k+1) = w_{33}(k+1)\theta_3(k)$$
$$\theta_1 = \theta - \varphi$$
$$x_B(k+1) = b_1(k+1)x_B(k)$$
$$y_B(k+1) = b_2(k+1)y_B(k)$$

$$(56)$$

7. Simulation Results

Simulation examples are carried out in order to evaluate the developed approach. It is desirable to move the end effector from its initial position P_1 *(1, 1, 0.2)* to its final position P_2 *(5,5,0.5)*, by tracking instantaneously a linear specified trajectory of the end effector generated by a uniform Cartesian movement. Neural-like-network learns the desired values presented and adjusts the weights appropriately in order to present to the system the corresponding reference state variables. The results of the simulation are shown in Figures 12 to 15 and indicate how successfully the Cartesian coordinates of the endeffector track their corresponding reference values very closely. We notice that the small departures from the reference trajectories are due to the cumulated tolerable errors from the learning process. The learning algorithm was run by using a learning rate µ=*0.05* for a laps of time not exceeding real time control. All the weights have been initialised to the value of one. At each step, the learning rate is updated depending on the behaviour obtained. If the overall error is improved, then the learning rate is increased by the value µ=µ*µ_*inc;* otherwise, it is deceased by the value µ=µ*µ_*dec,* knowing that µ_*inc* and µ_*dec* take the values of *1.05*, and *0.95* respectively. Figures 16 to 19 show the plots of the manipulator angular values as well as the orientation of the mobile platform, and Figure 20 clearly shows the trajectories of the end effector and the mobile platform in the *xyz* space. Figure 21 depicts a 3D perspective of the simulation environment.
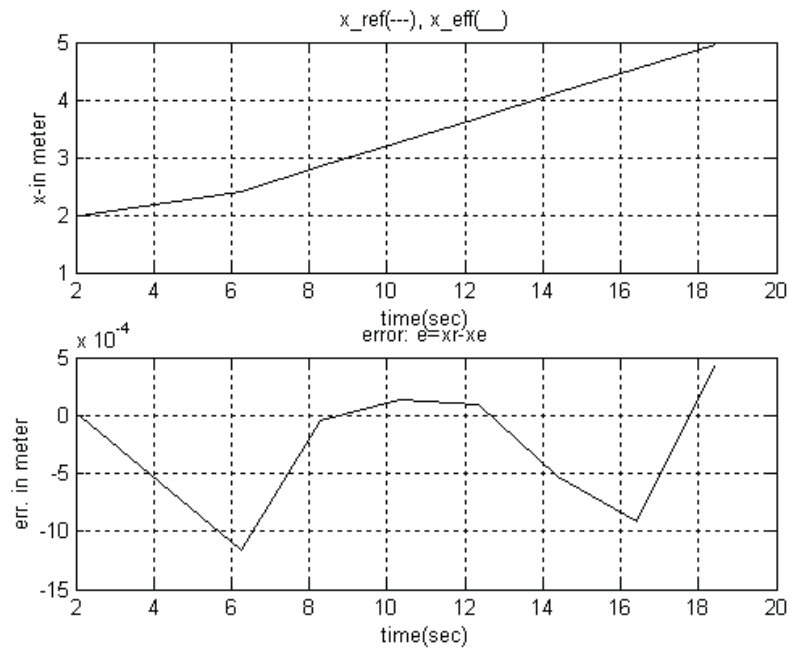
Figure 12. Desired and measured x-trajectory plots and the error resulted
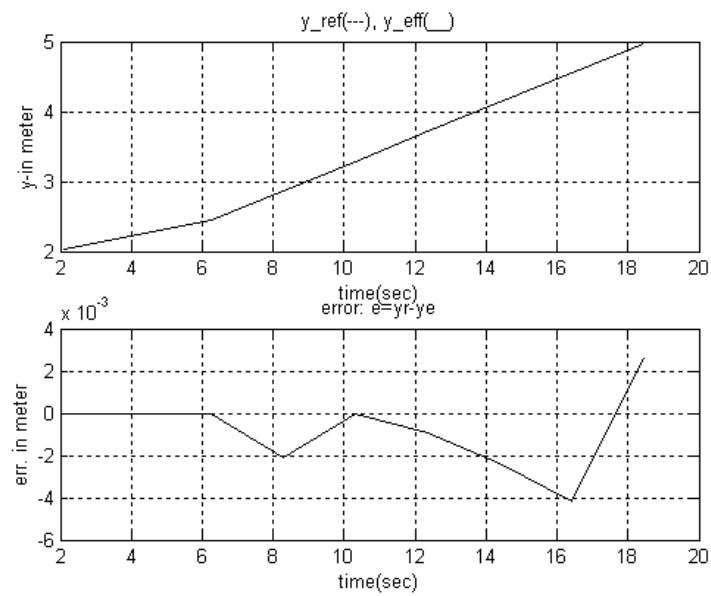
Figure 13. Desired and measured y-trajectory plots and the error resulted
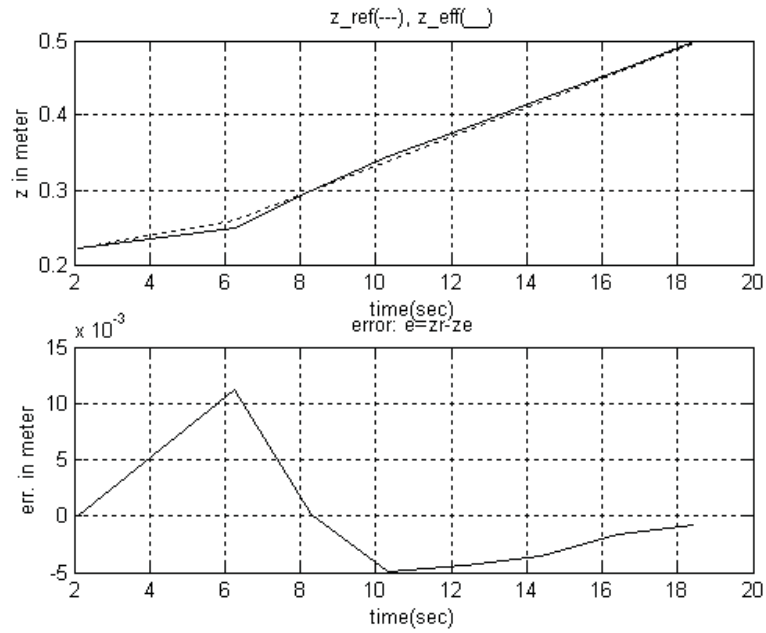
Figure 14. Desired and measured z-trajectory plots and the error resulted.
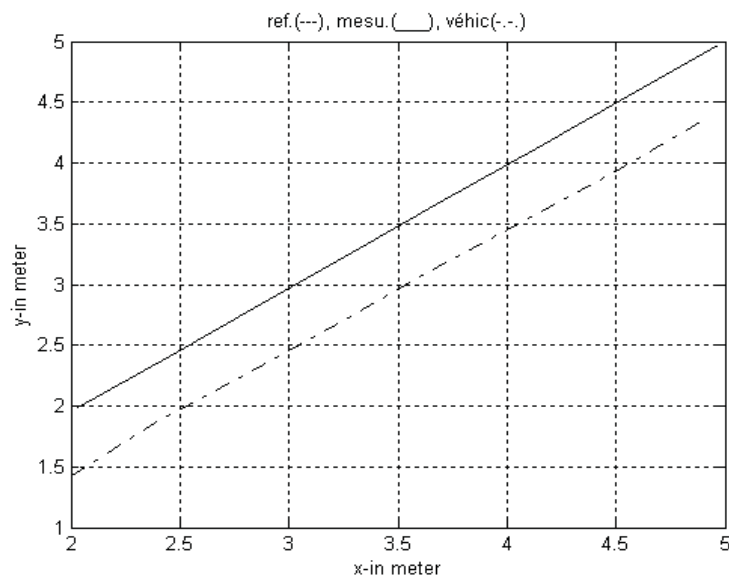
Figure 15. X-Y Plots of the end-effector an d the mobile platform trajectories
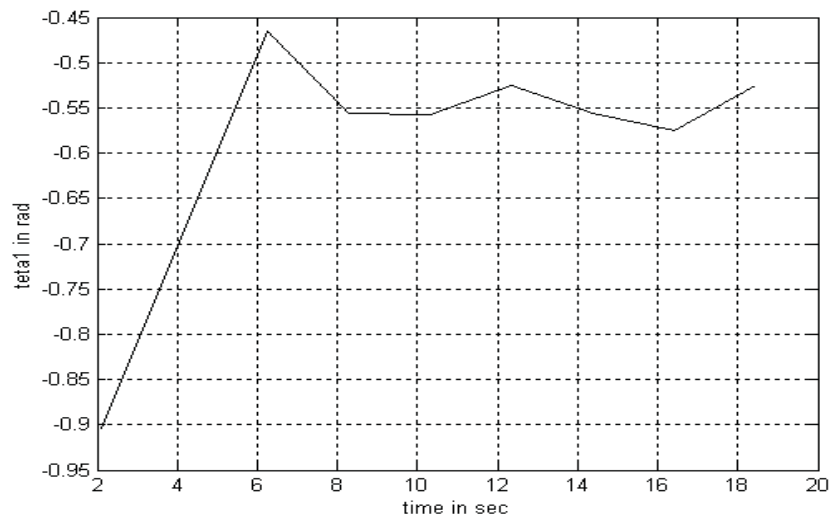
Figure 16. Angular trajectory θ_1
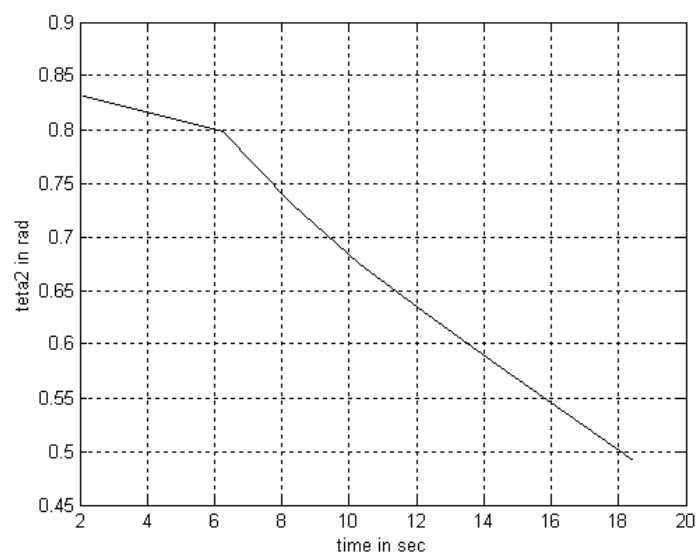
Figure 17. Angular trajectory θ_2
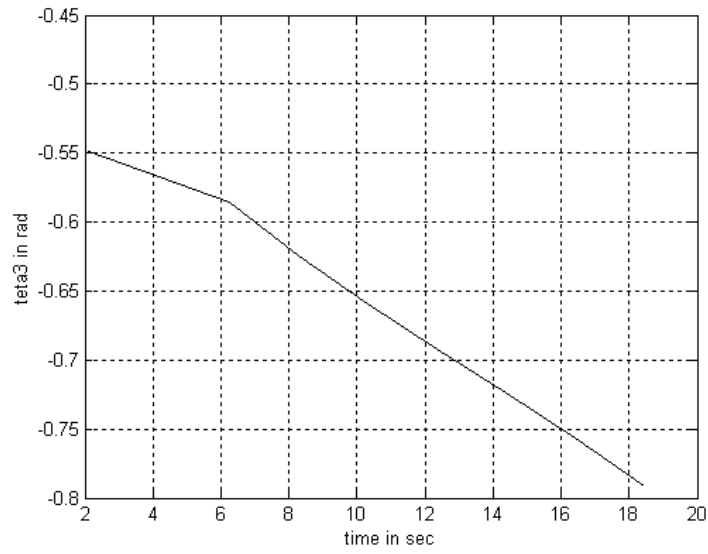
Figure 18. Angular trajectory θ_3
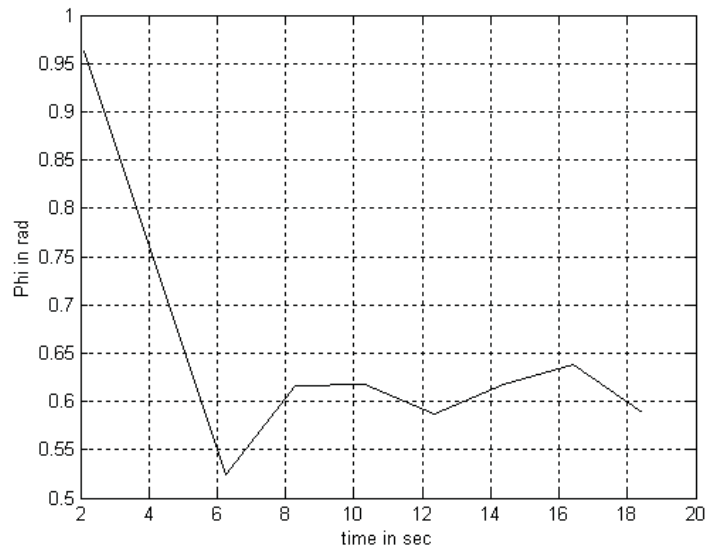
Figure 19. Angular trajectory φ
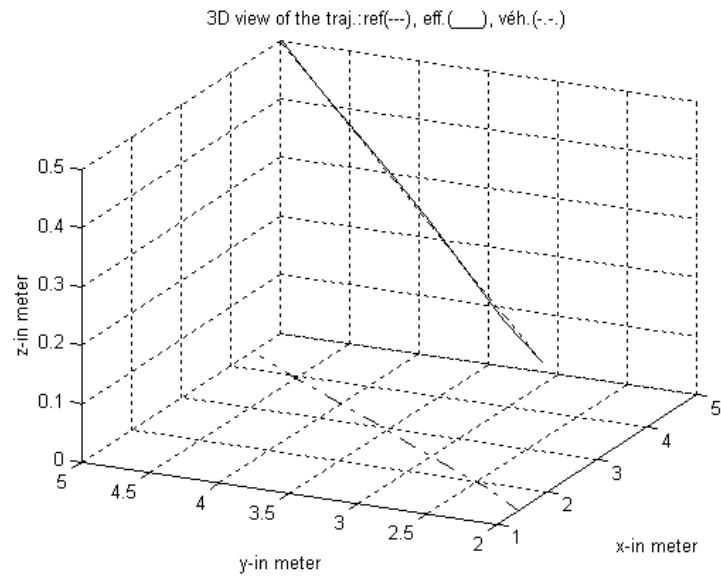
Figure 20. X-Y-Z plots of the end-effector and the mobile platform trajectories
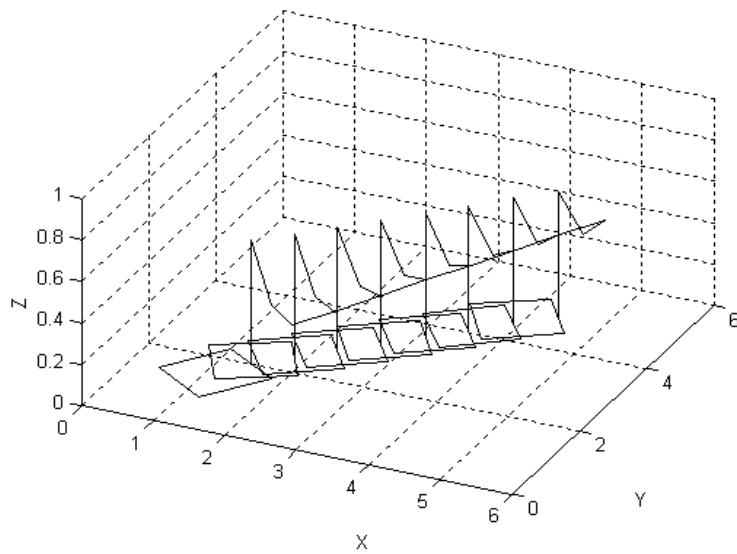
Figure 21. A 3D perspective of the simulation environment

8. Conclusion

Soft computing is an emerging field that consisting of complementary elements of fuzzy logic, neural computing, evolutionary algorithms, and machine reasoning. In this
paper we propose the use of a back propagation to train a neural-like-network to coordinate the motion of a robotic manipulator with the motion of the mobile platform over which a robot manipulator is mounted. The network pr vides reference output values of the desired motion to the mobile manipulator system. The parameter weighting vector determined is used to compute inputs to the platform and to the manipulator so that the end-effector trajectory is tracked with minimum error. Robot manipulator and platform control is predominately motion control. The mobile platform is considered as a "macro-mechanism" with coarse, slow dynamic response, and the arm is a fast and accurate "mini-device. For this reason we consider the kinematics model for the mobile platform and the dynamic model for the robot manipulator. Fuzzy controllers are used as means to control each of the two subsystems separately; and the overall system demonstrates very good control characteristics.

9. References

Abdessemed, F. & Benmahammed, K. (2001). A Two-layer robot controller design using evolutionnary algorithms, Journal of Intelligent and Robotic System 30, pp. 73-94, 2001.

Abdessemed, F. & Benmahammed, K. (2004). A Fuzzy-based reactive controller for non-holonomic mobile robot, Journal of Robotics and Autonomous Systems 47, pp. 31-46, 2004.

Bejczy, A. K., (1974). Robot Arm Dynamics and Control, *Technical Memorandum 33-669*, Jet Propulsion Laboratory, Feb. 1974.

Brock, O., Khatib, O. & Viji, S. (2002). Task-Consistent Obstacle Avoidance and Motion Behavior for Mobile Manipulator, In Proc. of the IEEE Int. Conf. on Robotics and Automation, 2002.

Craig, J.J., Hsu, P. & Sastry, S. (1987). Adaptive Control of Mechanical Manipulators, *The Inernational Journal of Robotics Research*, pp 16-28, Vol. 6. No. 2. Suymmer 1987.

Dubowsky, S., desforges, D. T., (1979). The Application of Model reference Adaptive Control to Robotic Manipulators, *Transaction of the ASME, Journal of Dynamic Systems, Measurement and control*, pp 193-200, Vol. 101, Sep. 1979.

Karr, C.L. (1991). Design of an Adaptive Fuzzy Logic Controller Using a Genetic Algorithm, in Proc. Fourth .Int..Conf. on Genetic Algorithms. pp. 450-457, San Diego, July 13-16, 1991.

Lee, J. K., & Cho, H. S. (1997). Mobile Manipulator Motion Planning for Multiple Tasks Using Global Optimization Approach, Jour. of Intell. and Robotic Systems 18, pp. 169-190, 1997.

Mamdani, E.H. & Assilian, S. (1974). Application of fuzzy algorithms for control of simple dynamic plant, Proc, Inst.Elec.Eng., pp.1585-1588, Vol. 121. 1974.

Mamdani, E.H. (1974). Application of Fuzzy Algorithms for Control of Simple Dynamic Plant, *Proc.IEE,121(12)*; pp1585-1588.-1974

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by Error Propagation, *in Parallel Distributed Processing: Exploration, in the Microstructure of Cognition, MA: MIT Press*, pp. 318-362,Vol.1, D.E.Rumelhart and James L. McClelland, Eds, Cambridge, Chap.8, 1986.

Samsung, C. & Abderrahim, K.A. (1990). Mobile robot control Part.I: Feedback control of a nonholonomic wheeled cart in cartesian space, Technical report, INRIA, France, 1990

Spong, M. W. & Ortega, R. (1988). On Adaptive Inverse Dynamic Control of Rigid robots, IEEE trans. Automat., 1988.

Werbos, P. (1974). Beyond Regression: New tools for Prediction and Analysis in the Behavioral Sciences, *Ph.D. Dissertation*, Harvard Univ., 1974.

Whiteney, D. E. (1969). Resolved Motion Rate Control of Manipulators and Human Protheses, *IEEE Transactions on Man Machine System*, pp. 47-53, Vol. MMS-10, No. 2, June 1969.

Yamamoto, Y. & Yun, X. (1994). Coordination locomotion and manipulation of a mobile manipulator, *IEEE Transactions on Automatic Control*, pp. 1326-1332, 39(6),1994

Zadeh, L. (1994). Fuzzy logic, Neural network, and Soft computing, commun. ACM, pp. 77-84, Vol. 37, no. 3, 1994.

Zadeh, L.A. (1973). Outline of a New Approach to the Analysis of Complex Systems and Decisions, IEEE Trans.SMG, pp. 28-44. Vol. 3, 1973

Control of Redundant Robotic Manipulators with State Constraints

Mirosław Galicki

1. Introduction

Recently, redundant manipulators have attracted much attention due to their potential abilities which are interesting from both a theoretical and practical point of view. Redundant degrees of freedom make it possible to perform some useful objectives such as collision avoidance in the work space with both static and moving obstacles, joint limit avoidance, and/or avoidance of singular configurations when the manipulator moves. Most of practical tasks, for example, inserting a shaft into co-operating elements (bearing, sleeve, or ratchet-wheel), require the knowledge of geometric paths (given in the work space) and a proper tolerance of matching that specifies the corresponding accuracy of the path following. In many other industrial tasks such as laser cutting or arc welding, the accuracy of path following is vital, and it is reasonable to assume that designers and manufacturers will specify precision using an absolute tolerance on tracking error. The application of redundant manipulators to such tasks complicates their performance, since these manipulators in general do not provide unique solutions. Consequently, some objective criteria should be specified to solve the robot tasks uniquely. The minimization of performance time is mostly considered in the literature . Several approaches may be distinguished in this context. Using the concept of a regular trajectory and the extended state space, the structure of path-constrained time-optimal controls has been studied in the works (Galicki, 1998b; Galicki, 2000) for kinematically redundant manipulators. Moreover, the efficient numerical procedures able to find such controls were also proposed in the works (Galicki, 1998a; Galicki & Pajak, 1999). Nevertheless, these algorithms require full knowledge of manipulator Jacobian matrix and robot dynamic equations, too.

Although all the aforementioned algorithms produce optimal solutions, they are not suitable to real-time computations due to their computational complexity. Therefore, it is natural to attempt other techniques in order to control the robot in real-time. Using on-line trajectory time scaling, a dynamic and computed torque laws respectively, a nearly time-optimal path tracking control for

non-redundant robotic manipulators with partially uncertain dynamics has been presented in works (Dahl, 1994; Kiefer et al., 1997). However, these algorithms require the solution of inverse kinematic problem along the path. A technique which avoids solving an inverse of robot kinematic equations and uses the exact Jacobian matrix, has been offered in (Galicki, 2001; Galicki 2004) for determining a collision-free trajectory of redundant manipulators operating in both a static environment (Galicki, 2001) and in a dynamic one (Galicki, 2004). Recently, a generalized transpose Jacobian controller with gravity compensation and a non-linear (saturating) derivative term has been introduced in (Galicki, 2006a) to generate robot controls subject to geometric path and actuator constraints.

As is known, many robotic controllers have been proposed to solve both a set point control problem (a regulation task) (Takegaki & Arimoto, 1981; Arimoto, 1996; Canudas de Wit et al., 1996; Sciavicco & Siciliano, 1996; Arimoto, 1990; Kelly, 1999; Galicki, 2002; Galicki, 2005) and the trajectory tracking (Slotine & Li, 1987; Slotine &Li, 1991; Feng & Palaniswami, 1992; Berghuis et al., 1993; Lewis et al., 1993; Tomei, 2000), respectively. However, most of these controllers have assumed full knowledge of manipulator kinematic equations. Recently, several approximate Jacobian setpoint controllers have been proposed (Cheach et al., 1999; Yazarel & Cheach, 2002; Cheach et al., 2003) to tackle uncertainties in both robot kinematics and dynamics. The controllers proposed do not require the exact knowledge of Jacobian matrix and dynamic equations. However, the results in (Cheach et al., 1999; Yazarel & Cheach, 2002; Cheach et al., 2003) are applicable only to a setpoint control of a robot.

This paper, which is based on our recent work (Galicki, 2006b), introduces a new class of adaptive path following controllers not requiring the full knowledge of both kinematic and dynamic equations in the control laws. Consequently, they are suitable for controlling uncertain robotic manipulators. Motivated in part by the dissipativity and adaptivity methodology (Slotine & Li, 1987), we develop path following controllers whose structure is composed of transpose adaptive Jacobian controller plus a non-linear term including an estimated control. Under the assumption of the full rank adaptive Jacobian matrix, the proposed control scheme has been derived based on the Lyapunov stability theory. By using sensory feedback of the end-effector position, it is also shown that the end-effector is able to follow a prescribed geometric path for robots with both uncertain kinematics and dynamics. Furthermore, new adaptive laws extending the adaptive algorithm from (Slotine & Li, 1987) to tackle kinemetic uncertainties too, are proposed. It is to notice that approximate Jacobian setpoint controllers from (Cheach et al., 1999; Yazarel & Cheach, 2002; Cheach et al., 2003) can not be directly applicable to our task. The reason is that approximate Jacobian matrix in (Cheach et al., 1999; Yazarel & Cheach, 2002; Cheach et al., 2003) does not include kinematic parameters to be adapted and the error of approximation is a'priori bounded. On the other

hand, the controller proposed in our work adaptively varies both kinematic parameters of the Jacobian matrix and the dynamic ones in such a way as to stably follow by the end-effector a geometric path. The paper is organized as follows. Section 2 formulates the robotic task to be accomplished in terms of a control problem. Section 3 describes how to employ the Lyapunov stability theory to determine controls (if they exist). Section 4 provides us with a computer example of generating the robot controls in a two dimensional task space for a planar redundant manipulator comprising three revolute kinematic pairs. Finally, some conclusions are drawn.

2. Formulation of the adaptive control problem

The control scheme designed in the next section is applicable to holonomic mechanical
systems comprising both non-redundant and redundant manipulators considered here which are described, in general, by the following dynamic equations, expressed in generalized co-ordinates (joint co-ordinates) $q = (q_1, ..., q_n)^T \in R^n$

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = u \tag{1}$$

where $M(q)$ denotes the $n \times n$ inertia matrix; $C(q, \dot{q})\dot{q}$ is the n-dimensional vector representing centrifugal and Coriolis forces; $G(q)$ stands for the n-dimensional vector of gravity forces; $u = (u_1, ..., u_n)^T$ is the vector of controls (torques/forces); n denotes the number of kinematic pairs of the V-th class. In most applications of robotic manipulators, a desired path for the end-effector is specified in task space such as visual space or Cartesian space. The aim is to follow by the end-effector a prescribed geometric path (given in the m-dimensional task space) described by the following equations

$$p(q) - \Theta(s) = 0 \tag{2}$$

where $p : R^n \to R^m$ denotes an m-dimensional, non-linear (with respect to vector q) mapping constructed from the kinematic equations of the manipulator; $p(q) = (p_1(q), ..., p_m(q))^T$; $\Theta(s) = (\Theta_1(s), ..., \Theta_m(s))^T$ stands for a given geometric path; s is the current parameter of the path (e.g. its length); $s \in [0, s_{max}]$; s_{max} is the maximal path length. The mapping Θ is assumed to be bounded together with the first and second derivatives and not degenerated, i.e. $\left\| \dfrac{d\Theta}{ds} \right\| > 0$.

It should be: The kinematic equations of a manipulator are independent $rank\left(\frac{\P p}{\P q}\right) = m$. In general (i.e. when $rank\left(\frac{\P p}{\P q}\right) \pounds m$, we should require that $rank\left(\frac{\P p}{\P q}\right) = rank\left(\left[\frac{\P p}{\P q} \frac{d\Theta}{ds}\right]\right)$ in order to guarantee consistency of the robotic task (2).

The problem is to determine control u which generates manipulator trajectory $q = q(t)$ and path parameterization $s = s(t)$ satisfying the equation (2) for each $t \in [0,T]$, where T denotes an (unknown) time horizon of task performance. It is natural to assume that at the initial moment $t = 0$, for which $s(0) = 0$, a given (by definition) initial configuration $q(0) = q_0$ satisfies (2), i.e.

$$p(q_0) - \Theta(0) = 0 \tag{3}$$

Final path parameterization fulfils the equality

$$s(T) - s_{max} = 0 \tag{4}$$

Furthermore, at the initial and the final time moment, the manipulator and path velocities equal zero, i.e.

$$\dot{q}(0) = \dot{q}(T) = 0 \tag{5}$$

and

$$\dot{s}(0) = \dot{s}(T) = 0 \tag{6}$$

As is known, task space velocity $\dot{p}$ is related to joint space velocity $\dot{q}$ as follows

$$\dot{p} = J(q, Y)\dot{q} \tag{7}$$

where $J(q,Y) = \frac{\partial p}{\partial q}$ is the $m \times n$ Jacobian matrix; Y stands for an ordered set of kinematic parameters $Y = (Y_1, ..., Y_k)^T$ such as link lengths, joint offsets; k denotes the number of kinematic parameters. Several important properties of dynamic equations (1) may be derived (Spong & Vidyasagar, 1989)

1. The inertia matrix $M(q)$ is symmetric and positive definite for all $q \in R^n$

2. Matrix $\dfrac{\dot{M}(q)}{2} - C(q,\dot{q})$ is skew-symmetric so that

$$\forall v, q, \dot{q} \in R^n \left\langle v, \left(\frac{\dot{M}(q)}{2} - C(q,\dot{q}) \right) v \right\rangle = 0 \tag{8}$$

where $\langle , \rangle$ is the scalar product of vectors.

3. The dynamic equations (1) are linear with respect to an ordered set of physical parameters $X = (X_1, ..., X_d)^T$, i.e.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = D(q,\dot{q},\dot{q},\ddot{q})X \tag{9}$$

where where $D(q,\dot{q},\dot{q},\ddot{q})$ is called the $(n \times d)$ dynamic regressor matrix; d stands for the number of the physical parameters such as link masses and inertias.

Differential equation (7) has the following property.

4. The right hand side of (7) is linear with respect to $e\,Y$. Consequently, equation (7) can be expressed as follows

$$\dot{p} = J(q,Y)\dot{q} = K(q,\dot{q})Y \tag{10}$$

where $K(q,\dot{q})$ is called the $(m \times k)$ kinematic regressor matrix.
In order to simplify further computations s_{max} is assumed to be equal to 1, i.e. $s_{max} = 1$. Let us define errors e and e_{m+1} of path following (task errors) as

$$e = (e_1, ..., e_m)^T = p(q) - \Theta(s) = (p_1 - \Theta_1, ..., p_m - \Theta_m)^T \tag{11}$$
$$e_{m+1} = s - 1.$$

Many commercial sensors are available for measurement of end-effector position p, such as vision systems, electromagnetic measurement systems, position sensitive detectors or laser tracking systems. Hence, path following error e in (11) is also assumed to be available (for a given s) from measurement. For revolute kinematic pairs, considered here, mapping $p(.)$ is bounded. Consequently, we have the following property.

5. Boundedness of mappings $p(\cdot), \Theta(\cdot)$, implies that task error e is bounded.

Expressions (1)-(6) formulate the robot task as a control problem. The fact that there exist state equality constraints makes the solution of this problem difficult. The next section will present an approach that renders it possible to solve the control problem (1)-(6) making use of the Lyapunov stability theory.

3. Adaptive path control of the manipulator

Our aim is to control the manipulator such that the end-effector fulfils (2)-(6). Therefore, we propose adaptive Jacobian path following controllers for robots with both uncertain kinematics and dynamics. In our approach, the exact knowledge of both robot dynamic equations and Jacobian matrix is not required in updating the uncertain parameters.

In the presence of kinematic uncertainty, the parameters of the Jacobian matrix are uncertain and hence equality (10) can be expressed as follows

$$\hat{J}\dot{q} = K(q,\dot{q})\hat{Y} \tag{12}$$

where $\hat{J} = J(q,\hat{Y}) \in R^{m \times n}$ is an adaptive Jacobian matrix and $\hat{Y}$ stands for the vector of estimated kinematic parameters. In order to show the stability of the path following control system in the presence of both kinematic and dynamic uncertainties, we define an adaptive joint-space sliding vector z as

$$z = \lambda \hat{J}^T e + \dot{q} \tag{13}$$

where λ is a positive scalar coefficient. The adaptivity of vector z is understood in the sense that the parameters of the adaptive Jacobian matrix will be updated by a parameter update law, defined later. Differentiating equation (13) with respect to time yields

$$\dot{z} = \lambda \dot{\hat{J}}^T e + \lambda \hat{J}^T \dot{e} + \ddot{q} \tag{14}$$

Based on equations (1) and (13)-(14), we can easily express robot dynamic equations by means of vector z and its time derivative, as

$$M(q)\dot{z} + C(q,\dot{q})z + M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + G(q) = u \tag{15}$$

Where $\ddot{q}_r = \left(\ddot{q}_{r,1}, ..., \ddot{q}_{r,n}\right)^T = -\lambda\dot{\hat{J}}^T e - \lambda\hat{J}^T \dot{e}$ and $\dot{q}_r = \left(\dot{q}_{r,1}, ..., \dot{q}_{r,n}\right)^T = -\lambda\hat{J}^T e$

Moreover, we know from property 3., that the last three terms of equation (15) become linear with respect to vector X and hence they can be written as follows

$$M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + G(q) = D(q,\dot{q},\dot{q}_r,\ddot{q}_r)X. \qquad (16)$$

Inserting the right hand side of (16) into (15), robot dynamic equations (15) take the following form

$$M(q)\dot{z} + C(q,\dot{q})z + D(q,\dot{q},\dot{q}_r,\ddot{q}_r)X = u \qquad (17)$$

Based on (13), (14) and (17) , lets propose the following adaptive Jacobian controller

$$u = -kz - k_P\hat{J}^T e + D(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{X} \qquad (18)$$

where $\hat{X}$ stands for the estimated physical parameter vector defined below; k and k_p are some positive scalars which could be replaced by diagonal matrices of positive constants without affecting the stability results obtained further on (this should lead to improved performance). Here, scalar constants are chosen for simplicity of presentation. In order to measure error e, path parameterization $s = s(t)$ is required which is computed by solving the following scalar differential equation

$$\ddot{s} = -k_s\dot{s} - k_P\left(\left\langle e, -\frac{d\Theta(s)}{ds}\right\rangle + \gamma(s-1) + \frac{1}{2}\frac{d\gamma}{ds}(s-1)^2\right) \qquad (19)$$

where k_s denotes a positive coefficient; γ is assumed to be a strictly positive function $\left(inf\{\gamma\}\right) > 0$ of s with bounded first and second derivatives for any $|s| \le 1 + \sqrt{\frac{\gamma_0}{inf\{\gamma\}}}$, where $\gamma_0 = \gamma(0)$ (as will be seen further on, γ_0 determines the upper bound on the accuracy of the path following and may be specified by the user).

The choice of function γ is crucial for computational effectiveness of control scheme (18), (19). One possibility is $\gamma = \gamma(s)$ with $\frac{d\gamma}{ds} > 0$. An alternative choi-

ce could be $\gamma = \gamma\left(e^2\right)$, where γ attains its maximum for $e = 0$ and smoothly decreases as $\|e\|$ increases. For simplicity of further considerations, we take the first form of γ.

Assumption 1. Function γ is required not to satisfy differential equation

$$\gamma + \frac{1}{2}\frac{d\gamma}{ds}(s-1) = 0$$

As will be seen further on, Assumption 1. results in an asymptotic convergence of s to 1.

Let us note, that the first two terms from (18) present an adaptive transpose Jacobian controller with adaptively varying kinematic parameter vector $\hat{Y}$. The last term in dependence (18) is an estimated control based on equation (16). Estimated kinematic parameters $\hat{Y}$ of the adaptive Jacobian matrix $\hat{J} = J\left(q, \hat{Y}\right)$ are updated according to the following law

$$\dot{\hat{Y}} = w_k k_P K^T(q, \dot{q}) e \tag{20}$$

and estimated physical parameters $\hat{X}$ of the dynamic equations are updated by

$$\dot{\hat{X}} = -w_d D^T\left(q, \dot{q}, \dot{q}_r, \ddot{q}_r\right) z \tag{21}$$

where $w_{k,} w_d$, are, similarly as before, positive gains (scalars) which could be replaced by diagonal matrices of positive constants without affecting the stability of controller (18). Although some kinematic parameters appear in $\hat{X}$, we should adapt on them separately in $\hat{Y}$ to preserve linearity.

Estimated kinematic parameters $\hat{Y}$ (updated according to rule (20)) are then used to compute adaptive Jacobian $\hat{J}$ and its time derivative $\dot{\hat{J}}$ using for this purpose the right hand side of (20) which is only a mapping of q, e, and $\dot{q}$. Having obtained the adaptive Jacobian matrix and its time derivative, we determine quantities $\dot{q}_r$ and $\ddot{q}_r$. It is worth noticing, that their computation does not require any pseudoinverse of matrix $\hat{J}$ which results in numerical stability of controller (18). Finally, based on $q, \dot{q}, \dot{q}_r$, , and $\ddot{q}_r$, we may determine dynamic regressor matrix $D\left(q, \dot{q}, \dot{q}_r, \ddot{q}_r\right)$, which is then used to update estimated physical parameter vector $\hat{X}$.

Let us note, that setpoint controllers proposed in works (Cheach et al., 1999; Yazarel & Cheach, 2002; Cheach et al., 2003) , which are computationally somewhat simpler, can not be applicable to our task. The reason is that the error of approximation in (Cheach et al., 1999; Yazarel & Cheach, 2002; Cheach et al., 2003) is bounded by a constant and approximate Jacobian matrix does not include parameters to be adapted. Due to adaptation law (20), one can not guarantee in our task to satisfy assumption regarding the approximation error made in works (Cheach et al., 1999; Yazarel & Cheach, 2002; Cheach et al., 2003) .

The closed-loop error dynamics is obtained by inserting the right hand side of equation (18) into equation (17)

$$
\begin{aligned}
&M(q)\dot{z} = -C(q,\dot{q})z + D(q,\dot{q},\dot{q}_r,\ddot{q}_r)\widetilde{X} - kz - k_P \hat{J}^T e \\
&\dot{e} = J(q,Y)\dot{q} - \frac{d\Theta(s)}{ds}\dot{e}_{m+1} \\
&\ddot{e}_{m+1} = -k_s \dot{e}_{m+1} - k_P\left(\left\langle e, -\frac{d\Theta(s)}{ds}\right\rangle + \gamma e_{m+1} + \frac{1}{2}\frac{d\gamma}{ds}e_{m+1}^2\right) \\
&\dot{\widetilde{Y}} = w_k k_P K^T(q,\dot{q})e \\
&\dot{\widetilde{X}} = -w_d D^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)z
\end{aligned}
\tag{22}
$$

where $\widetilde{Y} = \hat{Y} - Y$; $\widetilde{X} = \hat{X} - X$. Applying the Lyapunov stability theory, we derive the following result.

Theorem 1. If there exists a solution to the problem (1)-(6) and adaptive Jacobian matrix $\hat{J}$ is non-singular along end-effector path (2) and function γ fulfils Assumption 1., then control scheme (18) generates manipulator trajectory whose limit point $(\dot{q}(\infty), \dot{e}_{m+1}(\infty), e(\infty), e_{m+1}(\infty)) = (0,0,0,0)$, i.e. satisfying state constraints (2)-(6), is asymptotically stable.

Proof. Consider a Lyapunov function candidate

$$
V = \frac{1}{2}\left(\langle Mz,z\rangle + \frac{1}{w_d}\widetilde{X}^2 + \frac{1}{w_k}\widetilde{Y}^2 + \dot{e}_{m+1}^2 + k_P\gamma e_{m+1}^2 + k_P e^2\right).
\tag{23}
$$

The time derivative of V is given by

$$\dot{V} = \langle z, M\dot{z} \rangle + \left\langle \frac{\dot{M}}{2}z, z \right\rangle + k_P \langle J^T e, \dot{q} \rangle + \frac{1}{w_d} \langle \tilde{X}, \dot{\tilde{X}} \rangle + \frac{1}{w_k} \langle \tilde{Y}, \dot{\tilde{Y}} \rangle +$$

$$\dot{e}_{m+1} \ddot{e}_{m+1} + k_P \left\langle e, -\frac{d\Theta}{ds} \right\rangle \dot{e}_{m+1} + k_P \gamma e_{m+1} \dot{e}_{m+1} + k_P \frac{1}{2} \frac{d\gamma}{ds} e_{m+1}^2 \dot{e}_{m+1}.$$

(24)

Substituting $M\dot{z}, .\dot{e}, .\ddot{e}_{m+1}, \dot{\tilde{Y}}$ and $\dot{\tilde{X}}$ from $\dot{V}$ for the right-hand sides of closed-loop error dynamics (22) and using the skew-symmetric property of matrix $\frac{\dot{M}}{2} - C$ [property 2. eqn. (8)], we obtain after simple calculations, that

$$\dot{V} = -k \langle z, z \rangle - k_P \langle \hat{J}\hat{J}^T e, e \rangle - k_s \dot{e}_{m+1}^2.$$

Since $\dot{V} \le 0$, function V is bounded. Therefore, $z, \tilde{X}$, and $\tilde{Y}$ are bounded vectors. This implies that $\hat{X}$ and $\tilde{Y}$ are bounded, too. Consequently, it follows from (13), that $\dot{q}$ is also bounded. Moreover, s and $\dot{s}$ are bounded, too. As can be seen, $\dot{V}$ is negative for all $(z, e, \dot{e}_{m+1}) \ne 0$ and is zero only when $(z, e, \dot{e}_{m+1}) = 0$, which implies (using LaSalle-Yoshizawa invariant theorem (Krstic et al., 1995) that $(z, e, \dot{e}_{m+1})$ tends asymptotically to zero, i.e. $z(T) \to 0, e(T) \to 0$, and $\dot{e}_{m+1} \to 0, as. T \to \infty$, as. By differentiating $\ddot{e}_{m+1}$ in (22) with respect to time, it is also easy to see, that $\frac{d^3 e_{m+1}}{dt^3}$ is bounded function by assumptions regarding Θ and γ. This means, that $\ddot{e}_{m+1}$ is uniformly continuous. Hence, $\ddot{e}_{m+1}(T) \to 0,$, as $T \to \infty,$, too. The convergence of path velocity and acceleration yields the following equation

$$\gamma e_{m+1}(\infty) + \frac{1}{2} \frac{d\gamma}{ds} e_{m+1}^2(\infty) = 0.$$

(25)

Consequently, $e_{m+1}(\infty) = 0$ or $\gamma + \frac{1}{2} \frac{d\gamma}{ds} e_{m+1}(\infty) = 0$. On account of Assumption 1, the second equality is not fulfilled. Thus, $e_{m+1}(\infty) = 0$ (or equivalently $s(\infty) = 1$). On account of (13), $\dot{q}(T) \to 0$, as $T \to \infty$, too. Consequently, boundary conditions (4)-(6) are (asymptotically) fulfilled and limit point $(\dot{q}(\infty), \dot{e}_{m+1}(\infty), e(\infty), e_{m+1}(\infty)) = (0, 0, 0, 0)$ is asymptotically stable. Finally, it should be emphasized, that the chosen Lyapunov function does not guarantee convergence of parameter estimations $\hat{X}$ and $\hat{Y}$ to their true values.

On account of (3)-(6), we have $V_{t=0} = \dfrac{\tilde{X}_{t=0}^2}{2w_d} + \dfrac{\tilde{Y}_{t=0}^2}{2w_k} + \dfrac{k_p \gamma_0}{2}$.

For sufficiently large w_d and w_k, the first two terms in this equality may be omitted. Hence, we obtain $V_{t=0} \cong \dfrac{k_p \gamma_0}{2}$

Since $\dot{V}$ is not positive, function V fulfils the inequality $V \leq \dfrac{k_p \gamma_0}{2}$.

Consequently, the following bound on $\|e\|$ may easily be obtained, based on (23) and the last dependence

$$\|e\| \leq \sqrt{\gamma_0} \tag{26}$$

An important remark may be derived from the proof carried out. Namely, Inequality (26) presents an upper bound (path independent) on the accuracy of path following by the end-effector according to the control law (18). Let us note that estimation of the upper bound on path following error (26) is very conservative. Consequently, control gains w_d and w_k do not require large values to achieve a good path following accuracy, as the numerical simulations (given in the next section) show.

Moreover, several observations can be made regarding the control strategy (18). First note, that the proposed control law requires, in fact no information concerning the robot kinematic and dynamic equations. Second, the choice of controller parameters k, k_p, k_s, w_d and w_k according to dependencies (18)-(21) guarantees asymptotic stability of the closed-loop error dynamics (22) during the manipulator movement.

Moreover, the transpose of $\hat{J}$ (instead of a pseudoinverse) in control scheme (18) does not result in numerical instabilities due to (possible) kinematic singularities met on the robot trajectory. Nevertheless, (18) has been derived under the assumption of full-rank adaptive Jacobian matrix along the path. Furthermore, controller (18) does not require the knowledge of task space velocity. Due to conservative estimation of the path following accuracy, control algorithm (18) results in a better accuracy of the path following as compared to upper bound given from (26), as the numerical computations carried out in the next section show. In order to prevent control (torque) oscillations at the very beginning of time histories (caused by e.g. the non-zero initial path following error) k_s from (19) should be a bounded, quickly decreasing time dependent function as $t \to \infty$ (see the next section).

Due to real-time nature of robot controller (18), we shall try to estimate the number of arithmetic operations required to implement the algorithm presented in this section. The dimension of the robot task space is assumed in estimation to be constant. Operations required for computation of sin, cos, and $\Theta(\cdot)$ functions are not taken into account. Furthermore, matrices $\hat{J}$, K and D are assumed in estimation to be given. Moreover, estimations are carried out at any time instant of the robot task accomplishment. It follows from (13) and (18) that terms $kz, k_p \hat{J}^T e$, require $O(n)$ operations. Computation of the right hand sides (19) and (20) involves $O(1)$ and $O(n)$ operations, respectively assuming that $k = O(n)$. Computational complexity for the right hand side of (21) equals $O(n^2)$ by assumption that $d = O(n)$. Computation of estimated control $D(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\hat{X}$ requires also the same order of complexity, i.e. $O(n^2)$ operations. Finally, computational complexity of the whole robot controller (18) is of the order $O(n^2)$.

4. A numerical example

The aim of this section is to illustrate the performance of the proposed adaptive
control algorithm using a dynamic three-joint direct-drive arm $(n = 3)$ of
SCARA-type robotic manipulator operating in a two-dimensional $(m = 2)$ task space. Kinematic scheme of this manipulator and the task to be accomplished is shown in Fig. 1. In the simulations, SI units are used. The components of dynamic equations
of this manipulator are as follows (Spong & Vidyasagar, 1989):

$$M = \left\lfloor M_{ij} \right\rfloor_{1 \le i, j \le 3}$$

where $M_{11} = X_1 + 2X_4 c23 + X_6 c3$; $ci = cos(q_i)$, ; $si = sin(q_i)$, ; $cij = cos(q_i + q_j)$; $sij = sin(q_i + q_j)$, $cijk = cos(q_i + q_j + q_k)$, $sijk = sin(q_i + q_j + q_k)$, $M_{21} = X_2 + X_4 c2 + X_5 c23$, $M_{31} = X_3$, $M_{22} = X_2 + 2X_6 c3$, $M_{32} = X_3 + X_6 c3$, $M_{33} = X_3$, $M_{12} = M_{21}$, $M_{13} = M_{31}$, $M_{23} = M_{32}$.
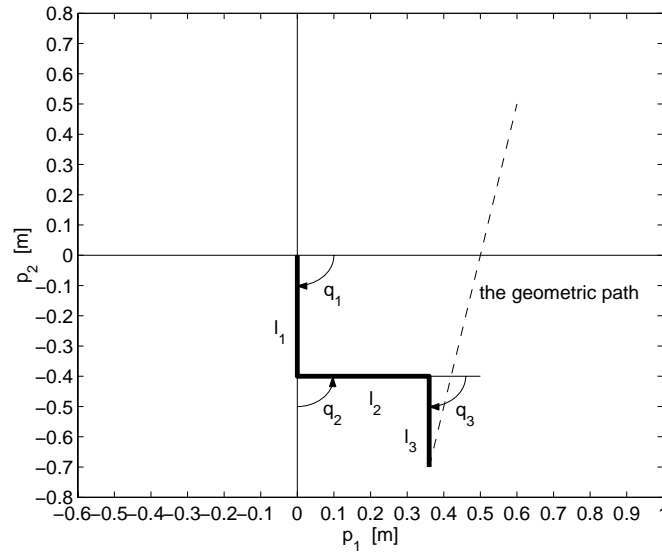
Figure.1 A kinematic scheme of the manipulator and the task to be accomplished

$$C = \left[C_{ij}\right]_{1 \le i, j \le 3}$$

where

$$C_{11} = -(X_4 s2 + X_5 s12)\dot{q}_2 - (X_5 s13 + X_6 s13)\dot{q}_3 ;$$

$$C_{12} = -(X_4 s2 + X_5 s12)(\dot{q}_1 + \dot{q}_2) - (X_5 s12 + X_6 s12)\dot{q}_3 ;$$

$$C_{13} = (X_5 s12 + X_6 s12)(-\dot{q}_1 + \dot{q}_2 + \dot{q}_3);$$

$$C_{21} = (X_5 s2 + X_5 s12)\dot{q}_1 + X_6 s3 \dot{q}_3 ;$$

$$C_{22} = -(X_5 s12 + X_6 s12)\dot{q}_3 ;$$

$$C_{23} = -X_6 s3 (3\dot{q}_1 + \dot{q}_2 + \dot{q}_3);$$

$$C_{31} = (X_4 s2 + X_5 s12)\dot{q}_1 - X_6 s3 \dot{q}_2 ;$$

$$C_{32} = X_6 s3 (\dot{q}_1 + \dot{q}_2);$$

$$C_{33} = 0 .$$

$$G = (G_1, G_2, G_3)^T$$

where

$$G_1 = X_7 c1 + X_8 c12 + X_9 c123 ; \quad G_2 = X_8 c12 + X_9 c123 ; \quad G_3 = X_9 c123 .$$

Parameters X_i, $i = 1:9$ take the following nominal values:

$X_1 = 1.1956$,
$X_2 = 0.3946$,
$X_3 = 0.0512$,
$X_4 = 0.4752$,
$X_5 = 0.1280$,
$X_6 = 0.1152$,
$X_7 = (m_1 l_{c1} + m_2 l_1 + m_3 l_1)g$,
$X_8 = (m_2 l_{c2} + m_3 l_2)g$,
$X_9 = m_3 l_{c3} g$,

where g stands for the gravity acceleration; m_i, l_i, and l_{ci}, $i = 1:3$ denote link mass, length and location of the mass center which is assumed to be equal to

$l_{ci} = l_i / 2$; $l_1 = 0.4$; $l_2 = 0.36$; $l_3 = 0.3$; $m_1 = 3.6$; $m_2 = 2.6$; $m_3 = 2$.

Jacobian matrix $J(q, Y)$ equals

$$ J = \begin{bmatrix} -Y_1 s1 - Y_2 s12 - Y_3 s123 & -Y_2 s12 - Y_3 s123 & -Y_3 s123 \\ Y_1 c1 + Y_2 c12 + Y_3 c123 & Y_2 c12 + Y_3 c123 & Y_3 c123 \end{bmatrix} $$

Where $Y_i = l_i; i = 1:3$;

and the kinematic regressor matrix takes the following form

$$ K = \begin{bmatrix} -s1\dot{q}_1 & -s12(\dot{q}_1 + \dot{q}_2) & -s123(\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \\ c1\dot{q}_1 & c12(\dot{q}_1 + \dot{q}_2) & c123(\dot{q}_1 + \dot{q}_2 + \dot{q}_3) \end{bmatrix} $$

The end-effector position $p = (p_1 p_2)^T$ (see Fig. 1) represents in the simulations the task space coordinates $(m = 2)$. The upper bound on the accuracy of the path following in all the computer simulations, is assumed to be equal to $\sqrt{\gamma_0} = 0.06$, where $\gamma(s) = 0,002 + 0,002 e^{2,7s}$. Let us introduce path following errors

$$
\begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} p_1 - \Theta_1 \\ p_2 - \Theta_2 \\ s - 1 \end{pmatrix}
$$

to evaluate the performance of the robot controller (18). In order to examine the effects of both kinematic and dynamic uncertainties, initial values for vectors $\hat{X}$ and $\hat{Y}$ were set in the simulations as $\hat{X}(0) = (32110.80.5302010)^T$, $\hat{Y}(0) = (0.550.450.4)^T$. The task of the robot is to transfer the end-effector along the geometric path (the dotted line in Fig. 1), expressed by the following equations

$$
\Theta_1(s) = 0.36 + 0.24s
$$
$$
\Theta_2(s) = -0.7 + 1.2s
$$

where, $s \in [0,1]$. The initial configuration q_0 equals $q_0 = (-\pi / 2 \pi / 2 - \pi / 2)^T$ Parameters $k = 10, .k_p = 3000, .\lambda = 1$ and $k_s = 6,9(1 + 100e^{-10t}) + 30$ have been chosen experimentally to achieve practically reasonable time horizon of task performance and relatively small controls with $w_k = diag(101010)$ and $w_d = diag(4.854.854.854.854.854.854.858.554.76)$. The results of computer simulation are presented in Figs 2-20.
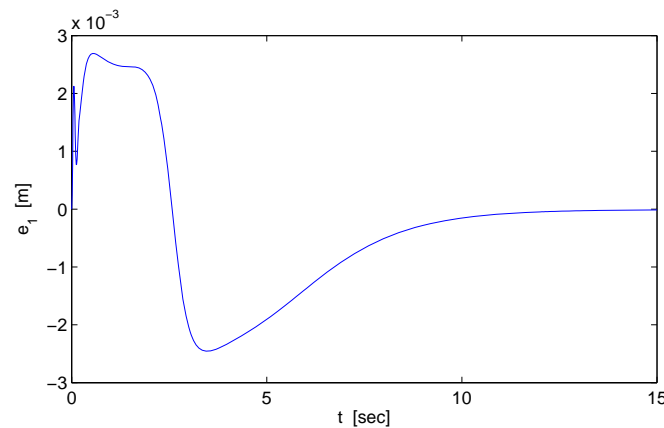
Figure 2. Path following error e_1 vs. time
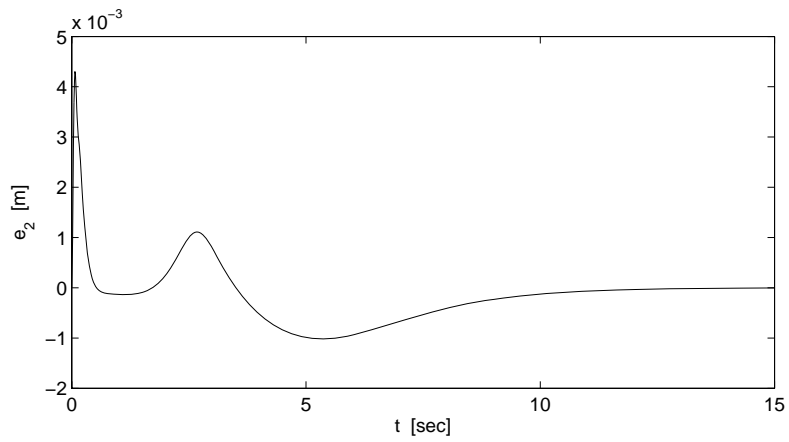
Figure 3. Path following error e_2 vs. time
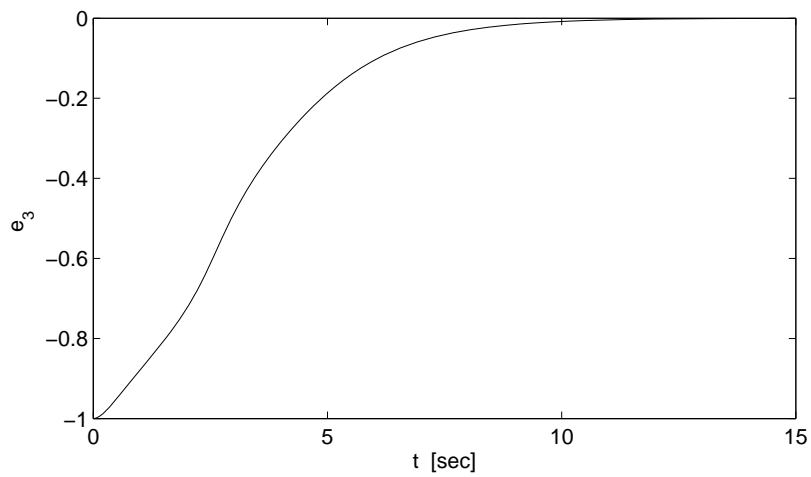
Figure 4. Path following error e_3 vs. time
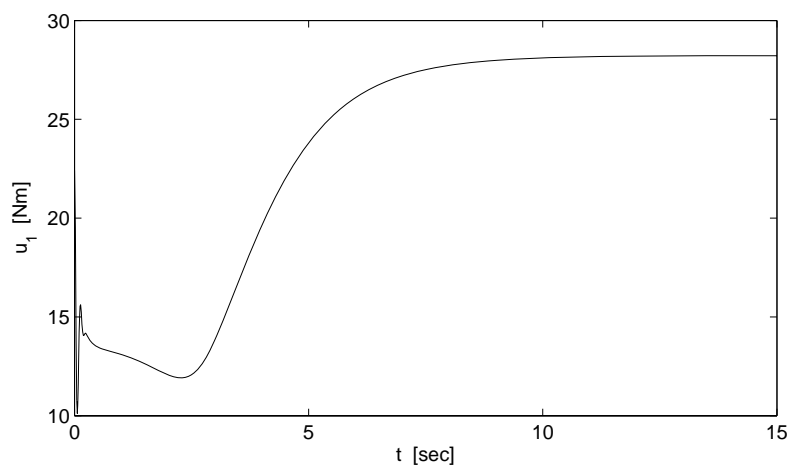
Figure 5. Input torque u_1 vs. time
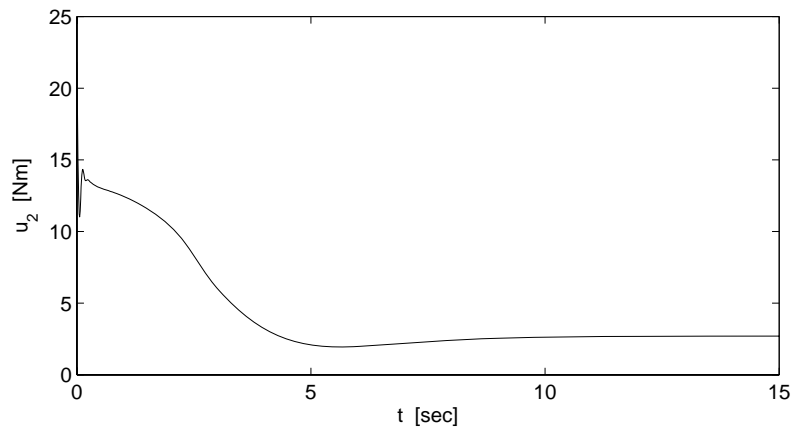
Figure 6. Fig. 6 Input torque u_2 vs. time
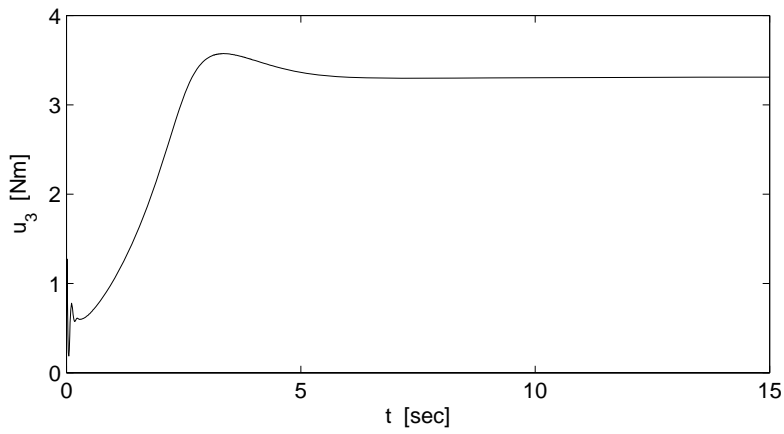
Figure 7. Input torque u_3 vs. time
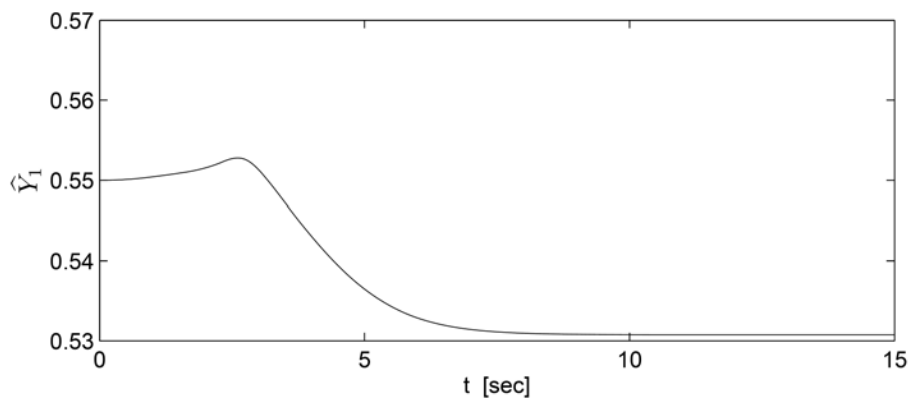
Figure 8. Time course of adaptive estimate $\hat{Y}_1$
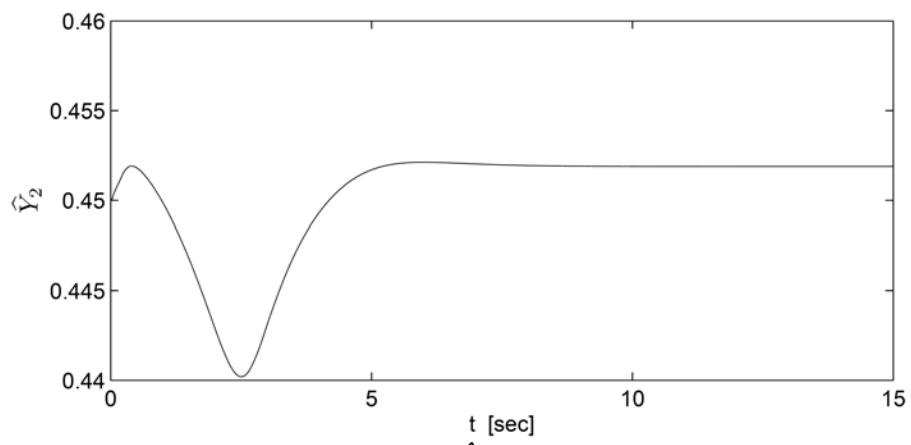
Figure 9. Time course of adaptive estimate $\hat{Y}_2$
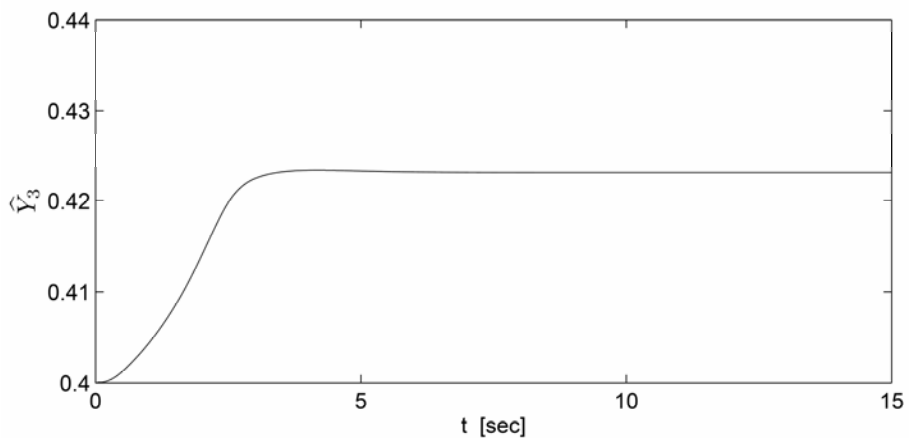
Figure 10. Time course of adaptive estimate $\hat{Y}_3$
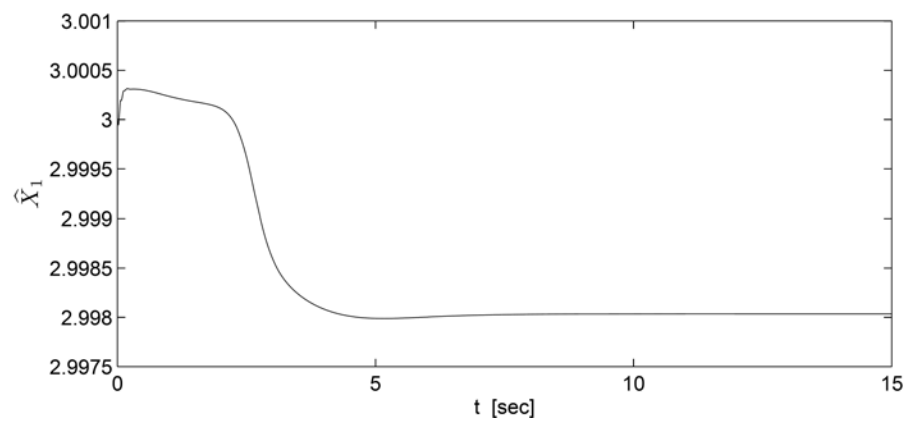
Figure. 11. Time course of adaptive estimate $\hat{X}_1$
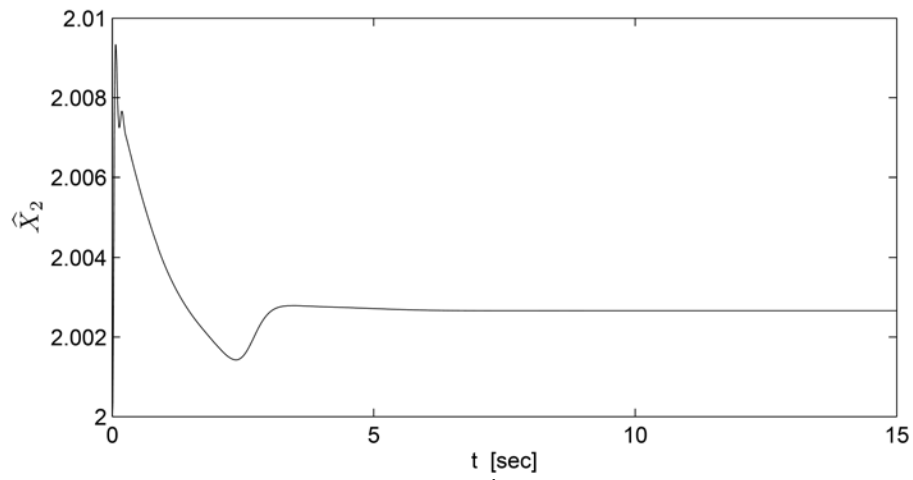
Figure 12. Time course of adaptive estimate $\hat{X}_2$
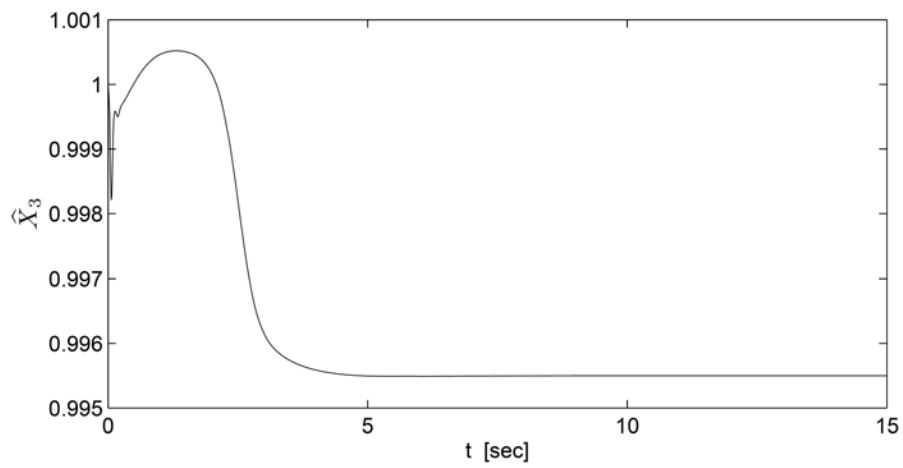
Figure 13. Time course of adaptive estimate $\hat{X}_3$
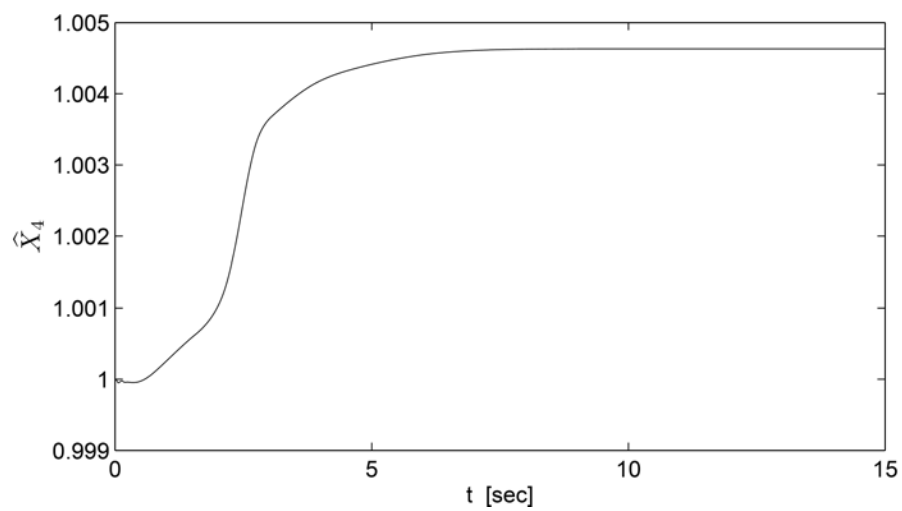
Figure 14. Time course of adaptive estimate $\hat{X}_4$
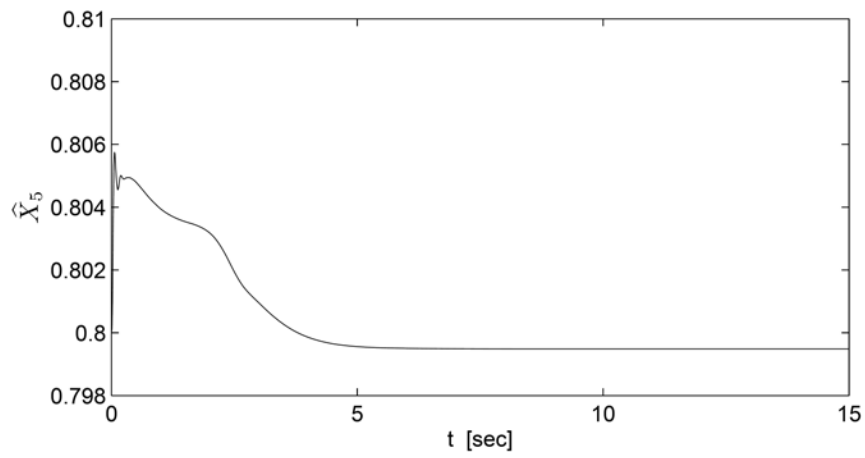
Figure 15. Time course of adaptive estimate $\hat{X}_5$
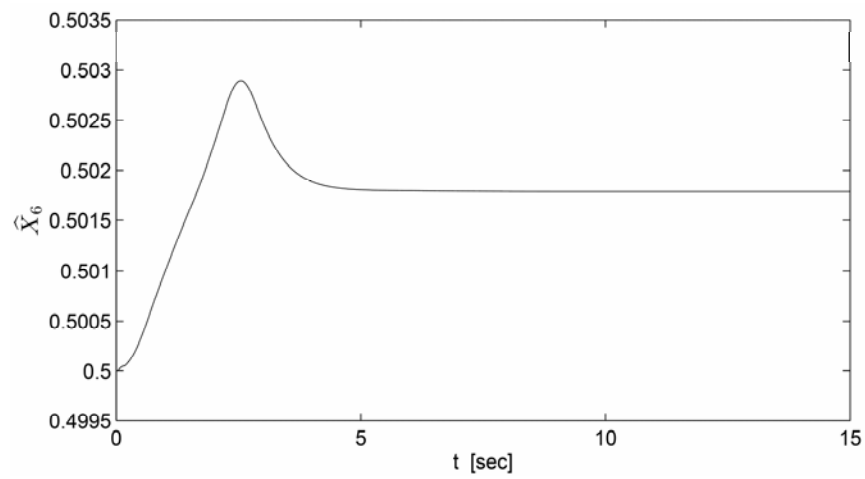
Figure 16. Time course of adaptive estimate $\hat{X}_6$
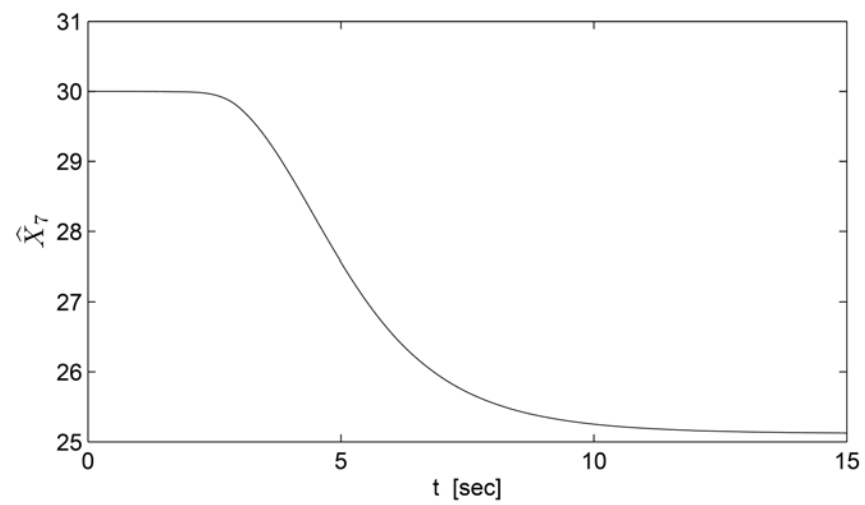
Figure 17. Time course of adaptive estimate $\hat{X}_7$
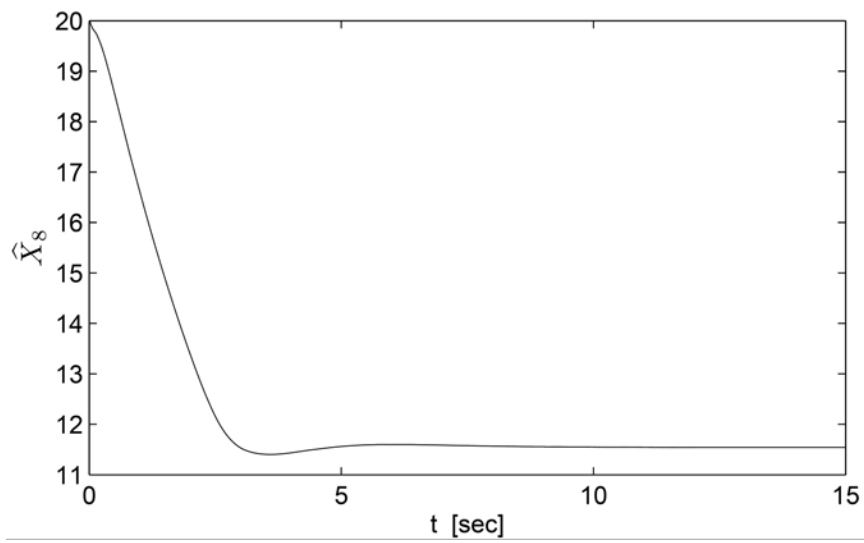
Figure 18. Time course of adaptive estimate $\hat{X}_8$
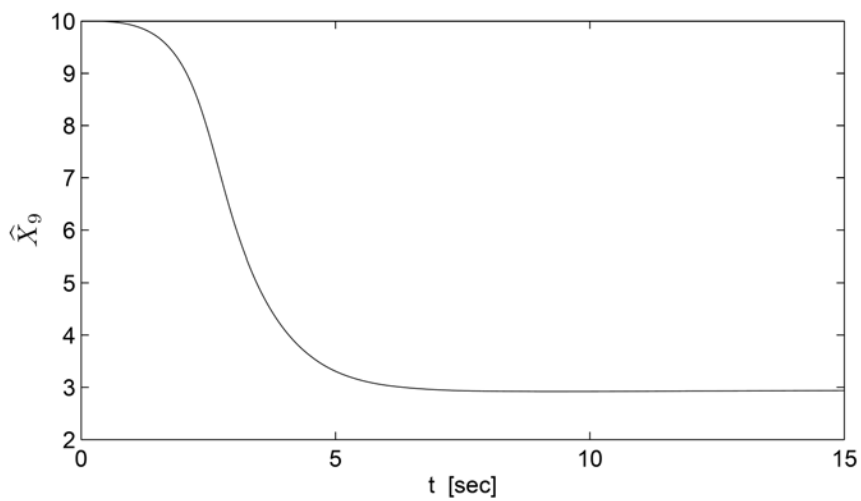
Figure 19. Time course of adaptive estimate $\hat{X}_9$
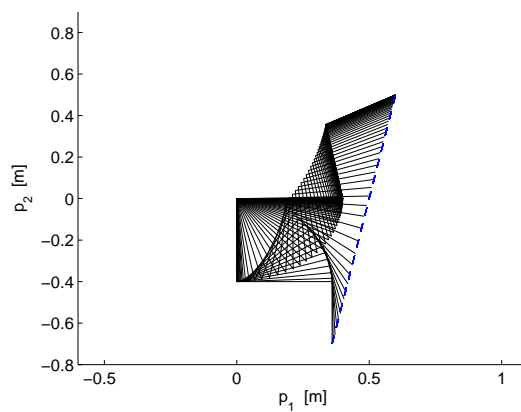
Figure 20. Manipulator motion along the geometric path

As might be expected, the path following errors from Figs 2-3 are much smaller than those obtained from the conservative dependence (26). Moreover, as one can observe from Figs 2-7, the time dependent damping function k_s decreases (eliminates) errors and torques oscillations at the very beginning of time histories. Furthermore, as seen from Figs 8-19, estimations $\hat{X}$, $\hat{Y}$ do not converge to their real (nominal) values.

5. Conclusion

This study has presented an adaptive robot controller for the path following by the end-effector. The control generation scheme has been derived using the Lyapunov stability theory. An advantage of the proposed control law (18) is that it requires, in fact no information regarding the parameters of the robot dynamic equations. The control strategy (18) is shown to be asymptotically stable (by fulfilment of practically reasonable assumptions). The proposed robot controller has been applied to a planar redundant manipulator of three revolute kinematic pairs operating in a two dimensional task space. Numerical simulations have shown that the results obtained are in accordance with the theoretical analysis. The novelty of the strategy proposed lies in its relative simplicity in design, program code and real-time implementation. The approach presented here will also be in future directly applicable to cooperating kinematically redundant manipulators.

6. References

Arimoto, S. (1990). Design of robot control systems, Adv. Robot., vol. 4, no. 1, pp. 79-97.

Arimoto, S. (1996). Control theory of non-linear mechanical systems, Oxford, U. K., Clarendon.

Berghuis, H.; R. Ortega, & H. Nijmeijer (1993). A robust adaptive robot controller, IEEE Trans. on Robotics and Automat., vol. 9, no. 6, pp. 825-830.

Canudas, de Wit, C.; B. Siciliano & G. Bastin (1996). Theory of robot control, New York: Springer-Verlag.

Cheah, C. C.; S. Kawamora & S. Arimoto (1999). Feedback control for robotic manipulators with an uncertain Jacobian matrix, J. Robot. Syst., vol. 6, no. 2, pp. 119-134.

Cheach, C. C.; M. Hirano; S. Kawamura & S. Arimoto (2003). Approximate ja-cobian control for robots with uncertain kinematics and dynamics, IEEE Trans. on Robotics and Automation, vol. 19, no. 4, pp. 692-702.

Dahl, O. (1994). Path-constrained robot control with limited torques - Experi-mental evaluation, IEEE J. Robot. Automat., vol. 10, pp. 658--669.

Feng, G. & M. Palaniswami (1992). Adaptive control of robot manipulators in task space, IEEE Trans. on Automatic Control, vol. 38, no. 1, pp. 100-104.

Galicki, M. (1998a). The planning of robotic optimal motions in the presence of obstacles, Int. J. Robotics Res., vol. 17, no. 3, pp. 248-259.

Galicki M. (1998). The structure of time optimal controls for kinematically re-dundant manipulators with end-effector path constraints, Proc. IEEE Conf. Robotics Automat. , pp.101-106, Leuven, Belgium.

Galicki, M. & I. Pajak (1999). Optimal motion of redundant manipulators with state equality constraints, Proc. IEEE Int. Symp. on Assembly and Task Planning, pp. 181-185, Porto, Portugal.

Galicki, M. (2000). Time-optimal controls of kinematically redundant manipu-lators with geometric constraints, IEEE Trans. Robotics Automat., vol. 16, no. 1, pp. 89-93.

Galicki, M. (2001). Real-time trajectory generation for redundant manipulators with path constraints, Int. J. Robotics Res., vol. 20, no. 7, pp. 673-690.

Galicki, M. (2002). Motion control of robotic manipulators in task space, Proc. IEEE/RSJ Int. Conf. Intell. Robots and Systems , pp. 2061-2066, Lau-sanne.

Galicki, M. (2004). Path following by the end-effector of a redundant manipu-lator operating in a dynamic environment, IEEE Trans. Robotics, vol. 20, no 6, pp. 1018--1025.

Galicki, M. (2005). Collision-free control of robotic manipulators in the task space, J. Robot., Syst., vol. 22, no 8, pp. 439--455.

Galicki, M. (2006a). Path-constrained control of a redundant manipulator in a task space, Robotics and Autonomous Systems, vol. 54, pp. 234-243.

Galicki, M. (2006b). Adaptive path-constrained control of a robotic manipula-tor in a task space, Robotica (to appear).

Kelly R. (1999). Regulation of manipulators in generic task space: An energy shaping plus damping injection approach, IEEE Trans. Rob. Automat., vol. 15, no. 2, pp. 381-386.

Kiefer, J.; A. J. Cahill & M. R. (1997). James, Robust and accurate time-optimal path-tracking control for robot manipulators, IEEE Trans. Automat. Ro-bot., vol. 13, pp. 880--890.

Krstic,M.; I. Kanellakopoulos & P. Kokotovic (1995). Nonlinear and adaptive control design, J. Wiley and Sons, New York.

Lewis, F. L.; C. T. Abdallach & D. M. Dawson (1993). Control of robot manipu-lators, New York: Macmillan Publishing Company.

Sciavicco, L. & B. Siciliano (1996). Modeling and control of robot manipulators, New York: McGraw-Hill.

Slotine, J. J. E. & W. Li (1987). On adaptive control of robot manipulators, Int. J. Robotics Res., no. 6, pp. 49-59.

Slotine, J. J. E. & W. Li (1991). Applied nonlinear control, Englewood Cliffs, New Jersey: Prentice Hall.

Spong, M. & M. Vidyasagar (1989). Robot dynamics and control, Wiley, New York.

Takegaki, M. & S. Arimoto (1981). A new feedback method for dynamic control of manipulators, Trans. ASME: J. Dyn. Syst., Meas., Contr., vol. 102, pp. 119-125.

Tomei, P. (2000). Robust adaptive friction compensation for tracking control of robot manipulators, IEEE trans. on Automatic Control, vol. 45, no. 11, pp. 2164-2169, 2000.

Yazarel, H. & C. C. Cheach (2002). Task-space adaptive control of robotic manipulators with uncertainties in gravity regressor matrix and kinematics, IEEE Trans. on Automatic Control, vol. 47, no. 9, pp. 1580-1585.

Model-Based Control for Industrial Robots: Uniform Approaches for Serial and Parallel Structures

Houssem Abdellatif and Bodo Heimann

1. Introduction

Nowadays, there are still two major challenges for industrial robotics in automated production. These are enhancing manufacturing precision and reducing cycle-times. Beside the advances made in the classic robotics over the last decades, new technologies are emerging in the industrial field aiming more flexible high-speed and accurate manufacturing. Robots with parallel structures are attracting the attention of automation industry as an innovative product with high dynamic potentials. Such robots, like the tricpet are integrated nowadays by BMW, Volvo or Airbus in their manufacturing lines. Compared to each other, the classic serial (or open) chain robots and the parallel (or closed) chain robots have their specific benefits and suffer from own drawbacks. The proposed chapter gives a comparison of the two types in the scope of their suitability for solving modern problems in industrial robotics. Additionally, appropriate approaches are proposed to remove drawbacks of classic industrial control solutions. Hereby, it is focussed on model-based strategies for ameliorating control accuracy at high dynamics and therefore to expose solutions towards high-speed automation.

One of the main purposes of the proposed chapter is to contribute to extending the state of the art in industrial robotics by the innovative class of parallel robots. Furthermore, classic and advanced model-based control approaches are discussed for both robot types. Uniform methodologies for both classes are given. It is focused on crucial issues for practical application in the industrial filed.

The first aspect is surely the modelling of kinematics (see section 2) and dynamics (see section 3) for serial and parallel robots. Here, an opposite duality in formalism is shown. By appropriate choice of minimal coordinates and velocities, the inverse dynamics of the two robot classes can be derived by the principle of virtual power. This yields computational highly efficient models that are well appropriate for real-time applications. Since the success of such feedforward control depends on the estimation quality of the model parame-

ters, appropriate strategies for experimental identification are provided in section 4. Thereby, two main categories of procedures are discussed: direct and indirect identification. The direct procedure tries to estimate model parameters from measurements achieved by one optimized trajectory. Indirect identification uses standard Point-to-Point motions that are distributed within the workspace. The choice of the method in praxis depends on the used control hardware and sensors. Each approach has own advantages and drawbacks for the here discussed two classes of robotic manipulators.

In section 5, further enhancement of control accuracy are demonstrated by providing pre-correction techniques, like iterative learning control, training or nonlinear pre-correction. Such powerful tools are highly appropriate for manufacturing or automation tasks that are repeated over and over. Furthermore, it is advantageous not only due to the simple requirement of standard position-correction interface but because complex modeling of disturbances is not necessary. The methodology is exposed uniformly for serial and parallel robots. Practical issues and some differences are pointed out. Experimental results prove than the suitability and effectiveness of the proposed methods for the studied classes of robots. All proposed approaches are substantiated by experimental results achieved on three different robots: the *Siemens Manutec-r15*, the *KUKA KR15* and the prototype *PaLiDA* as a parallel robot. The chapter is closed with conclusions and an outlook on the possible future of industrial robotics.

2. Kinematic Analysis

To enable giving uniform approaches for serial and parallel robots, elementary assumptions and definitions at the formal level have to be revised. As mentioned in the introduction, we will concentrate on the case of industrial relevant robotic systems, i.e. $n = 6$-DOF non redundant mechanisms. Both mechanisms are supposed to have n_a actuated joints grouped in the vector q_a, that defines the actuation space A. Additionally, passive joints are denoted by q_p. Both vectors can be grouped in the joint vector $q = \left[q_a^T \, q_p^T \right]^T$ that correspond consequently to the joint space Q. The operational or work-space W of an industrial robot is defined by the 6-dimensional pose vector x containing the position and orientation of the end-effector (EE) with respect to the inertial frame. Let the vector z now denotes the generalized (or minimal) coordinates, which contains the independent coordinates that are necessary to uniquely describe the system. Its dimension coincides therefore with the number of DOF's (Meirovitch, 1970; Bremer, 1988) and it defines the configuration space C.

Already at this formal level, important differences between serial open-chain robots and parallel closed-chain robots are necessary to consider. For classic industrial robots, the case is quite simple and well known. Such systems do

not have passive joints, the actuated joints correspond to the minimal coordinates, which yields the coincidence of almost all coordinate spaces:

$$q = q_a \Rightarrow Q \equiv A \text{ and } z = q_a = q \Rightarrow C \equiv Q \equiv A.$$

The case of 6-DOF parallel robot is more complicated. The pose vector x defines uniquely the configuration of the system. Besides the robot contains passive joints (Merlet, 2000)

$$z = x \Rightarrow C \equiv W \text{ and } q \neq q_a, \ q_a \neq z \Rightarrow C \neq Q \neq A$$

Consequently, more transformations have to be considered while operating parallel robots. A more serious issue in industrial praxis is that the configuration of parallel robots can not be directly measured, since only the positions of actuated joints are available. It is than necessary to consider this limitation in control issues. To keep uniform handling of both robotic types, it is recommended to focus on the configuration space defined by z. From this point of view the most important notions of kinematics are revisited in the following. The interested reader may be referred to standard books for deeper insight (Tsai, 1999; Sciavicco & Siciliano, 2000; Angeles, 2003; Merlet, 2000; Khalil & Dombre, 2002)

2.1 Kinematic Transformations

In robotics, the motion of each link is described with respect to one or more frames. It is though necessary to define specifications to transform kinematic quantities (positions, velocities and accelerations). Homogenous transformations are state of the art in robotics. Besides the fixed inertial frame $(KS)_0$ and the end-effector frame $(KS)_E$, each link i is associated with body-fixed frame $(KS)_i$. For efficient formulation and calculation of the homogenous transformations (given by T_i^{i-1}) between two adjacent links $i-1$ and i, it is recommended to use the modified DENAVIT-HARTENBERG-notation (or MDH), that yields unified formalism for open and closed-chain systems (Khalil & Kleinfinger, 1986; Khalil & Dombre, 2002). We obtain:

$$T_i^{i-1} = \begin{bmatrix} R_i^{i-1} & {}^{(i-1)}r_i^{i-1} \\ 000 & 1 \end{bmatrix} = \begin{bmatrix} c_{\vartheta i} & s_{\vartheta i} & 0 & a_i \\ s_{\vartheta i}c_{\alpha i} & c_{\vartheta i}c_{\alpha i} & -s_{\alpha i} & -d_i s_{\alpha i} \\ s_{\vartheta i}s_{\alpha i} & c_{\vartheta i}s_{\alpha i} & c_{\alpha i} & d_i c_{\alpha i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

which is a function of the MDH-parameters ϑ_i, d_i, α_i and a_i (Khalil & Klein-finger, 1986). The abbreviations s_x and c_x denote $\sin(x)$ and $\cos(x)$ respectively. The matrix $\boldsymbol{R}_i^{i-1}$ and the vector $_{(i-1)}\boldsymbol{r}_i^{i-1}$ define orientation and position of frame i with respect to frame $i-1$. The kinematics of any kinematic chain gives an analytic determination of the joint variables ϑ_i (for revolute joints) and d_i (for prismatic joints) as well as their time derivatives. The velocity $_{(i)}\boldsymbol{v}_i$ and angular velocity $_{(i)}\boldsymbol{\omega}_i$ of each link i and the corresponding accelerations can be calculated recursively by the following equations:

$$_{(i)}\boldsymbol{v}_i = {}_{(i)}\boldsymbol{v}_{i-1} + {}_{(i)}\widetilde{\boldsymbol{\omega}}_{i-1}\ {}_{(i)}\boldsymbol{r}_i^{i-1} + \boldsymbol{e}_z\,\dot{d}_i \tag{2}$$

$$_{(i)}\dot{\boldsymbol{v}}_i = {}_{(i)}\dot{\boldsymbol{v}}_{i-1} + {}_{(i)}\dot{\widetilde{\boldsymbol{\omega}}}_{i-1}\ {}_{(i)}\boldsymbol{r}_i^{i-1} + {}_{(i)}\widetilde{\boldsymbol{\omega}}_{i-1}\ {}_{(i)}\widetilde{\boldsymbol{\omega}}_{i-1}\ {}_{(i)}\boldsymbol{r}_i^{i-1} + \ddot{d}_i\,\boldsymbol{e}_z + 2\dot{d}_i\ {}_{(i)}\widetilde{\boldsymbol{\omega}}_{i-1}\,\boldsymbol{e}_z \tag{3}$$

$$_{(i)}\boldsymbol{\omega}_i = {}_{(i)}\boldsymbol{\omega}_{i-1} + \boldsymbol{e}_z\,\dot{\vartheta}_i \tag{4}$$

$$_{(i)}\dot{\boldsymbol{\omega}}_i = {}_{(i)}\dot{\boldsymbol{\omega}}_{i-1} + \dot{\vartheta}_i\ {}_{(i)}\widetilde{\boldsymbol{\omega}}_{i-1}\,\boldsymbol{e}_z + \ddot{\vartheta}_i\,\boldsymbol{e}_z \tag{5}$$

where $\boldsymbol{e}_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}}$. The Tilde-operator $(\widetilde{\ })$ defines the cross product $\widetilde{\boldsymbol{a}}\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$.

2.2 Direct and Inverse Kinematics

Industrial applications are characterized by being defined in the operational space W, whereas the robot is controlled in the actuation space A. It is therefore necessary to define and to calculate transformations between the two spaces. Calculating the resulting robot poses from given actuator positions correspond to the direct (or forward) kinematic transformation:

$$f : A \to W$$
$$\mathrm{q}_a \to x = \mathrm{f}(\boldsymbol{q}_a)$$

Reciprocally, the inverse (or backward) kinematic transformation is used to obtain actuator positions from a given robot pose:

$$g : \mathrm{W} \to \mathrm{A}$$
$$x \to q_a = g(x)$$

We mentioned above, that only the minimal coordinates describe the system uniquely. Consequently, only the transformations having the argument set be-

ing the configuration space can be computed or given in a closed form. This fact explains, that the solution of the inverse problem is quite simple and available analytically for parallel robots $(C \equiv W)$. Whereas the solution of the forward kinematics can be generally obtained only in a numerical way (Tsai, 1999; Merlet, 2000). In contrast, the forward kinematics can be easily obtained for serial-chain robots $(C \equiv A)$, whereas the inverse problem is generally cumbersome to solve. As it will be discussed in following sections, such system-inherent properties have an important impact on the practical implementation of control. E.g. the well-known computed-torque feedback approach is not suitable for parallel robots, since the minimal coordinates $z = x$ can not be measured.

For both robotic types the pose vector is defined as:

$$x = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma \end{bmatrix}^{\mathrm{T}},$$

where the end-effector position being $r_{\mathrm{E}} = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}}$ and its orientation $\pi = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^{\mathrm{T}}$ being defined according to the Roll-Picth-Yaw (RPY) Euler-convention (Tsai, 1999; Sciavicco & Siciliano, 2000). The homogeneous transformation between $(KS)_0$ and $(KS)_{\mathrm{E}}$ is given by

$$T_{\mathrm{E}}^0(x) = \begin{bmatrix} c_\beta c_\gamma & -c_\beta s_\gamma & s_\beta & x \\ c_\alpha s_\gamma + s_\alpha s_\beta c_\gamma & c_\alpha c_\gamma - s_\alpha s_\beta s_\gamma & -s_\alpha c_\beta & y \\ s_\alpha s_\gamma - c_\alpha s_\beta c_\gamma & s_\alpha c_\gamma + c_\alpha s_\beta s_\gamma & c_\alpha c_\beta & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3 Differential Kinematics

The differential kinematics maps the velocity of the end-effector into the velocity of the actuated joints $\dot{q}_{\mathrm{a}}$ and vice versa. It is necessary to relate a desired motion in the task-space to the necessary motion of the actuated joints. This is achieved by the jacobian matrix

$$\dot{x} = \begin{bmatrix} \dfrac{\partial f_1}{\partial q_{\mathrm{a},1}} & \cdots & \dfrac{\partial f_1}{\partial q_{\mathrm{a},n_{\mathrm{a}}}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial q_{\mathrm{a},1}} & \cdots & \dfrac{\partial f_n}{\partial q_{\mathrm{a},n_{\mathrm{a}}}} \end{bmatrix} \dot{q}_{\mathrm{a}} \tag{7}$$

or simply

$$\dot{x} = J_{\mathrm{A}} \dot{q}_{\mathrm{a}}. \tag{8}$$

The analytical Jacobian J_A relates the time derivative of the pose vector to the articulated velocities. Since the orientation vector $\boldsymbol{\pi}$ is composed of pseudo-coordinates, whose time derivative has no physical meanings (Bremer, 1988, Meirovitch, 1970) it is convenient to define the rotational velocities of the end-effector in respect to the fixed frame: $\boldsymbol{\omega}_E = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$, such that

$$\boldsymbol{\omega}_E = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & s_\beta \\ 0 & c_\alpha & -s_\alpha c_\beta \\ 0 & s_\alpha & c_\alpha c_\beta \end{bmatrix}}_{R_K(\alpha,\beta)} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} \tag{9}$$

and therefore the definition of the geometric jacobian matrix J:

$$v_E = \begin{bmatrix} \dot{r}_E \\ \boldsymbol{\omega}_E \end{bmatrix} = J\dot{q}_a, \text{ with } J = \begin{bmatrix} I & 0 \\ 0 & R_K \end{bmatrix} J_A \tag{10}$$

By regarding eq. (7) it is obvious that the analytic derivation of the jacobian is only available, when the direct kinematic solution $f(q_a)$ is given in a closed form. This is the case for classic open-chain robots, whereas for parallel robots, the inverse jacobian J^{-1} is available (Merlet, 2000). For such mechanisms, the jacobian is obtained by numerical inversion of its analytically available inverse (Merlet, 2000; Abdellatif et al., 2005a). The mobility of robots depends on the structure of the related jacobian that describes the velocity and also the force transmission between the operational space and the actuation space. It is well known, that singularities occur at configurations, when the jacobian loses its rank $(\det(J) = 0)$. The study of singularities is omitted in this paper. The interested reader may be referred to standard and excellent literature in this area (Gosselin & Angeles, 1990; Sciavicco & Siciliano, 2000; Merlet, 2000; Bonev, 2002).

It is now necessary to define further quantities to describe the motion of robotic manipulators. In analogy to the generalized coordinates, the generalized velocities are introduced (Meirovitch, 1970; Bremer, 1988) and are denoted by $\dot{s}$. They always present a linear combination of the time-derivatives of the generalized coordinates $\dot{z}$. The simplest case is when these combinations correspond to the identity:

$$\dot{s} = I\dot{z} \Rightarrow \dot{s} = \dot{z}$$

This is the case of classic open-chain robots: $\dot{s} = \dot{q}_a$. For parallel manipulators, the end-effector's velocities are chosen to be the generalized coordinates:

$\dot{s} = v_E \neq \dot{x} = \dot{z}$. This formal property has also an important impact in the practice. The symbolic derivation of the Lagrangian equations of motions becomes very messy for parallel robots, such that its implementation in real-time control systems is very restrictive (Tsai, 1999).

The last fundamental step of our revised kinematic analysis is the definition of limb's jacobians J_{T_i} and J_{R_i} that relate its translational and its angular velocities to the generalized velocities of the robot, respectively:

$$J_{T_i} = \frac{\partial_{(i)} v_i}{\partial \dot{s}} \qquad \text{and} \qquad J_{R_i} = \frac{\partial_{(i)} \omega_i}{\partial \dot{s}}$$

The use of the modified DENAVIT-HARTENBERG-notation allows also a recursive calculation of the limb's jacobians:

$$J_{T_i} = R_{i-1}^i \left(J_{T_{i-1}} - {}_{(i-1)}\tilde{r}_i^{i-1} J_{R_{i-1}} \right) + e_z \frac{\partial \dot{d}_i}{\partial \dot{s}} \tag{11}$$

$$J_{R_i} = R_{i-1}^i J_{R_{i-1}} + e_z \frac{\partial \dot{\vartheta}_i}{\partial \dot{s}} \tag{12}$$

The next subsection demonstrates the efficiency and uniformity of the proposed method for deriving the kinematics of a serial and a parallel industrial robot.

2.4 Application of the Kinematic Analysis of Industrial Robots

2.4.1 Serial Manipulators: Case Study KUKA KR15

The direct kinematics of serial-chain robots is straight forward. The transformation matrix can be calculated by starting from the base and evaluating the single T_i^{i-1}. By solving

$$T_E^0(q_a) = \prod_{i=0}^n T_i^{i-1}(q_{a,i}) \overset{!}{=} T_E^0(x),$$

we obtain the pose vector x. The jacobian is also joint-wise simple to obtain:

$$J(q) = \begin{bmatrix} J_1 & | & J_2 & | & \dots & | & J_n \end{bmatrix} \text{ with } J_i = \begin{bmatrix} {}_{(0)} e_z^i \times {}_{(0)} r_E^i \\ {}_{(0)} e_z^i \end{bmatrix} \tag{13}$$

This can be deduced by using the MDH-notation and the recursive formulae given above. Although the solution of the inverse kinematics is generally hard to obtain for open-chain mechanisms, industrial robots are characterized by simple geometry, such that a closed-form solution exists. This is the case here, where the three last revolute joint axes intersect at a common point (spherical wrist) (Sciavicco & Siciliano, 2000).

2.4.2 Parallel Manipulators: Case Study PaLiDA

The general method of calculating the inverse kinematics of parallel robots is given by splitting the system into a set of subchains. The structure is opened and separated into "legs" and an end-effector-platform. Hereby the enclosure constraints have to be calculated, which are the vectors connecting A_j with B_j

$$r_{B_j}^{A_j} = \begin{bmatrix} x_j & y_j & z_j \end{bmatrix}^{\mathrm{T}} = -r_{A_j}^0 + r_{\mathrm{E}}^0 + R_{\mathrm{E}\,(\mathrm{E})}^0 \, r_{B_j}^{\mathrm{E}}. \tag{14}$$

Thus, every chain can now be regarded separately as a conventional open-chain robot with a corresponding end-effector position at $r_{B_j}^{A_j}$. MDH-Parameters are defined for each subchain and the direct kinematics is solved as described above. Since we consider non-redundant mechanisms, the resulting serial chains are very simple, such that a closed form solution always exists. For the studied case *PaLiDA*, the definition of the MDH-parameters and frames are depicted in Figure 2. The solution of the full inverse kinematics is obtained by

$$q_{\mathrm{a}_j} = l_j = \sqrt{x_j^2 + y_j^2 + z_j^2} \tag{15}$$

$$\alpha_j = \arctan\left(\frac{x_j}{-z_j}\right) \tag{16}$$

$$\beta_j = \arctan\left(\frac{y_j}{r_j}\right), \tag{17}$$

which are quite simple equations. The differential kinematics can be deduced analytically for the inverse problem by the inverse jacobian:

$$J(x)^{-1} = \frac{\partial \dot{q}_{\mathrm{a}}}{\partial \dot{s}} = \begin{bmatrix} \dfrac{\partial \dot{q}_{\mathrm{a}}}{\partial \dot{r}_{\mathrm{E}}} & \dfrac{\partial \dot{q}_{\mathrm{a}}}{\partial \dot{\omega}_{\mathrm{E}}} \end{bmatrix} \tag{18}$$
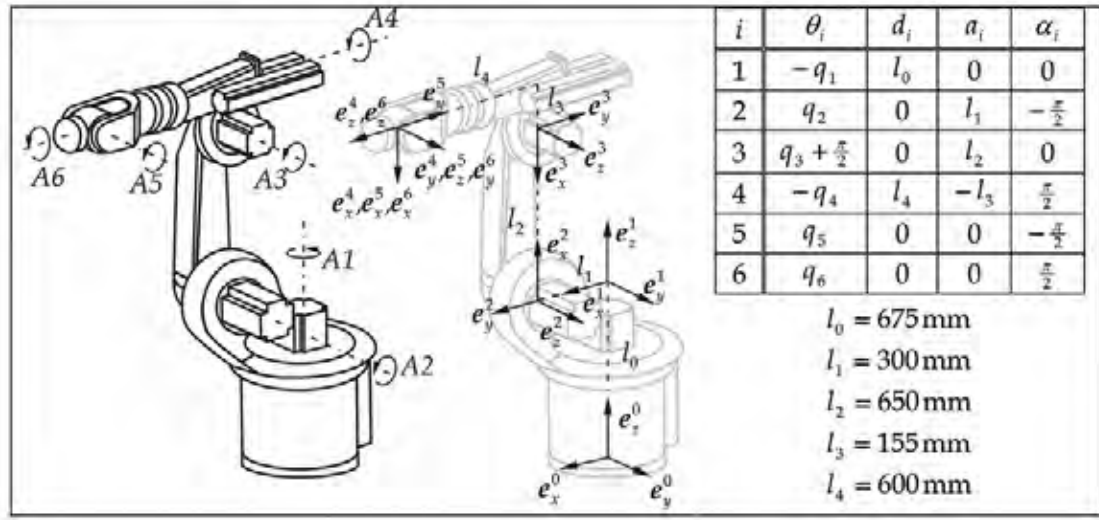
Figure 1. Definition of the MDH Coordinate systems and parameter for the *KUKA KR 15*

Many methods are proposed in the literature for calculating the inverse jacobian. We propose here the most straight-forward way in our case. Every single chain j corresponds to the jth raw of the inverse jacobian:

$$
J_j^{-1} = \frac{\partial \dot{q}_{a_j}}{\partial v_{B_j}} \quad \frac{\partial v_{B_j}}{\partial \dot{s}} = \frac{\partial \dot{q}_{a_j}}{\partial v_{B_j}} \left[\frac{\partial v_{B_j}}{\partial \dot{r}_E} \quad \frac{\partial v_{B_j}}{\partial \omega_E} \right] = \frac{\partial \dot{q}_{a_j}}{\partial v_{B_j}} \left[I - \tilde{r}_{B_j}^E \right]
\tag{19}
$$

The velocities of the points B_j can be obtained by simply differentiating the constraint equation (14):

$$
v_{B_j} = \dot{r}_E + \tilde{\omega}_E r_{B_j}^E = \dot{r}_R + \tilde{r}_{B_j}^E \omega_E
\tag{20}
$$

By using the recursive laws given by eq. (3-5) the complete inverse kinematics of the subchains can be solved, yielding velocities and accelerations of each limb and moreover a functional relationship between q_{a_j} and v_{B_j} that is needed for (19).
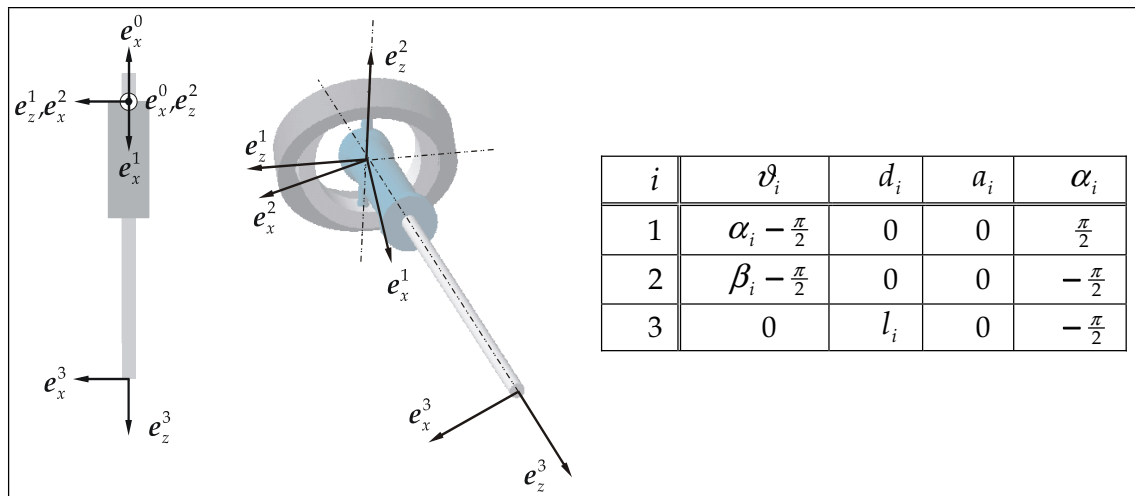
Figure 2. Definition of the MDH-parameters for a serial subchain of the hexapod *PaLiDA*

As conclusions, we can retain that the formal differences between parallel and serial robots have to be taken into account. A unified approach can be given if the notions of minimal coordinates and velocities are kept in mind. The MDH-notation provide the same procedure when solving kinematics for both robotic types. For parallel robots it is sufficient to formulate the constraint equations. Hereafter the mechanism is separated into serial subchains that can be treated exactly as any other open-chain manipulator.

3. Efficient Formulation of Inverse Dynamics

Model-based and feedforward control in industrial robotics requires computational efficient calculation of the inverse dynamics, to fulfill real-time requirements of standard control systems. The real-time calculation of the desired actuator forces Q_a depends on the used approach for the derivation of the inverse Model. For the sake of clarity we concentrate first on rigid-body dynamics. The corresponding equations of motions for any manipulator type can be derived in the following four forms:

$$Q_a = B_a(z, \dot{s}, \ddot{s}) \tag{21}$$

$$Q_a = M_a(z)\ddot{q}_a + N(z, \dot{s}) \tag{22}$$

$$Q_a = M_a(z)\ddot{q}_a + c_a(z, \dot{s}) + g_a(z) \tag{23}$$

$$Q_a = A_a(z, \dot{s}, \ddot{s})p_{min} \tag{24}$$

where Q_a being the vector of actuation forces. The massmatrix is denoted by M. The vectors c and g contain the centrifugal and Coriolis, and the gravitational terms, respectively. The vector N includes implicitly c and g. Analogically, the vector $B(z, \dot{s}, \ddot{s})$ includes implicitly all terms of rigid-body dynamics. We notice here, that the index 'a' is used to distinguish the quantities that are related to the actuation space. A trivial but very important remark is that all model forms have in common, that the inputs are always given in the configuration space by z, $\dot{s}$ and $\ddot{s}$, whereas the outputs are always given in the actuation space: Q_a. Although, equations (21-24) yield exactly the same results, they are very different to derive and to calculate. Although eq. (23) is the most computational intensive form, it is very reputed in robotics because it is highly useful for control design and planning. The case of open-chain manipulators is easier. The coincidence of configuration space with the actuation space allows a straight-forward implementation of the Lagrangian formalism for its derivation. This is not the case for the parallel counterpart, where the same formalism leads to messy symbolic computation or in the worst case to non-closed form solution (Tsai, 1999). Therefore, we focus in the following on the most efficient[1] form (21) that can be derived uniformly for parallel and serial robots.

3.1 Derivation of the Rigid-Body Dynamics

The suggested approach is the Jourdainian principle of virtual power that postulates power equality balances with respect to the forces in different coordinate spaces (Bremer, 1988). For instance, a power balance equation is obtained as

$$\partial \dot{s}^{\mathrm{T}} \tau = \partial \dot{q}_a^{\mathrm{T}} Q_a \Leftrightarrow \tau = \left(\frac{\partial \dot{q}_a}{\partial \dot{s}} \right)^{\mathrm{T}} Q_a \tag{25}$$

where τ is the vector of the generalized forces. Equation (25) means that the virtual power resulting in the space of generalized velocities is equal to the actuation power. The power balance can be applied for rigid-body forces:

$$Q_{a,rb} = \left(\frac{\partial \dot{q}_a}{\partial \dot{s}} \right)^{-\mathrm{T}} \tau_{rb} \tag{26}$$

---

[1] Parameterlinear equations of motions (24) are actually more computational efficient. Since they are derived from (21), they are discussed later on.

The generalized forces are computed as the summation of the power of all N_K limbs:

$$\boldsymbol{\tau}_{\text{rb}} = \sum_{i=1}^{N_K} \left[\left(\frac{\partial v_{S_i}}{\partial \dot{s}} \right)^{\text{T}} m_i a_{S_i} + \left(\frac{\partial \boldsymbol{\omega}_i}{\partial \dot{s}} \right)^{\text{T}} \left(I_i^{(S_i)} \dot{\boldsymbol{\omega}}_i + \tilde{\boldsymbol{\omega}}_i \left(I_i^{(S_i)} \boldsymbol{\omega}_i \right) \right) \right] \tag{27}$$

with $a_{S_i} = \dot{v}_{S_i} - g$ being the absolute acceleration of the ith link's center of gravity S_i. The velocity of the center of gravity, the mass and the inertia-tensor with respect to S_i are denoted by $\dot{v}_{S_i}$, m_i and $I_i^{(S_i)}$, respectively. To be able of using the recursion calculation of kinematic quantities (2-5, 11), eq. (27) is transformed to

$$\boldsymbol{\tau} = \sum_{i=1}^{N_K} \left[\underbrace{\left(\frac{\partial_{(i)} v_i}{\partial \dot{s}} \right)^{\text{T}}}_{J_{T_i}^{\text{T}}} \left(m_{i\,(i)} a_i + {}_{(i)} \dot{\tilde{\boldsymbol{\omega}}}_i s_i + {}_{(i)} \tilde{\boldsymbol{\omega}}_{i\,(i)} \tilde{\boldsymbol{\omega}}_i s_i \right) \right.$$

$$\left. + \underbrace{\left(\frac{\partial_{(i)} \boldsymbol{\omega}_i}{\partial \dot{s}} \right)^{\text{T}}}_{J_{R_i}^{\text{T}}} \left({}_{(i)} I_i^{(i)} {}_{(i)} \dot{\boldsymbol{\omega}}_i + {}_{(i)} \tilde{\boldsymbol{\omega}}_i \left({}_{(i)} I_i^{(i)} {}_{(i)} \boldsymbol{\omega}_i \right) + \tilde{s}_{i\,(i)} a_i \right) \right] \tag{28}$$

with s_i being the vector of the ith body's first moment[2] $s_i = \begin{bmatrix} s_{i_x} & s_{i_y} & s_{i_z} \end{bmatrix}^{\text{T}} = m_{i\,(i)} r_{S_i}^i$ ($r_{S_i}^i$: location of S_i with respect to the limb-fixed coordinate frame) and ${}_{(i)} I_i^{(i)}$ being the inertia tensor about the same coordinate frame.

It is obvious, that the calculation of $\boldsymbol{\tau}_{\text{rb}}$ requires the quantities of motions of all bodies. The latter can be obtained by applying the kinematic analysis as already explained in the former section 2. The transformation of the generalized forces into the actuation space according to (2) is trivial for the case of serial robots $\left(\dot{s} \equiv \dot{q}_{\text{a}} \right)$

$$\left(\frac{\partial \dot{q}_{\text{a}}}{\partial \dot{s}} \right)^{-\text{T}} = \left(\frac{\partial \dot{q}_{\text{a}}}{\partial \dot{q}_{\text{a}}} \right)^{-\text{T}} = I$$

---

[2] not to confuse with generalized velocities $\dot{s}$

For parallel manipulators, the numerical calculation of the jacobian is necessary (see also section 2.3):

$$Q_{\mathrm{a,rb}} = \left(\frac{\partial \dot{q}_{\mathrm{a}}}{\partial \dot{s}} \right)^{-\mathrm{T}} \boldsymbol{\tau}_{\mathrm{rb}} = J^{\mathrm{T}} \boldsymbol{\tau}_{\mathrm{rb}}$$

The inverse dynamics presented by (28) is already highly computational efficient. It can be implemented in real-time within nowadays standard control systems for parallel as well as for serial ones. Such model can be further optimized and transformed into a linear form with respect to the minimal dynamic parameters $\boldsymbol{p}_{\min}$.

3.2 Minimalparameter Form of the Equations of Motion

By transforming the dynamics into form (24), two main advantages result. At one hand, regrouping the parameters will further reduce the calculation burden and at the other hand, one obtains the set of identifiable parameters of the robotic system. We focus now on the dynamic parameters presented by m_i, $\boldsymbol{s}_i$ and $\boldsymbol{I}_i$. To regroup such parameters, the definition of two new operators $(.)^*$ and $(.)^\diamond$ are required first:

$$\boldsymbol{\omega}_i^* \boldsymbol{I}_i^\diamond := {}_{(i)}\boldsymbol{I}_i^{(i)}{}_{(i)}\boldsymbol{\omega}_i \tag{29}$$

with $\boldsymbol{\omega}_i^* := \begin{bmatrix} \omega_{i_x} & \omega_{i_y} & \omega_{i_z} & 0 & 0 & 0 \\ 0 & \omega_{i_x} & 0 & \omega_{i_y} & \omega_{i_z} & 0 \\ 0 & 0 & \omega_{i_x} & 0 & \omega_{i_y} & \omega_{i_z} \end{bmatrix}$ and $\boldsymbol{I}_i^\diamond = \begin{bmatrix} I_{i_{xx}} & I_{i_{xy}} & I_{i_{xz}} & I_{i_{yy}} & I_{i_{yz}} & I_{i_{zz}} \end{bmatrix}^{\mathrm{T}}$

The inverse dynamics (28) can be written as

$$\boldsymbol{\tau}_{\mathrm{rb}} = \sum_{i=1}^{N_K} \underbrace{\begin{bmatrix} J_{\mathrm{T}_i}^{\mathrm{T}} & J_{\mathrm{R}_i}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{0} & {}_{(i)}\dot{\tilde{\boldsymbol{\omega}}} + {}_{(i)}\tilde{\boldsymbol{\omega}}_i {}_{(i)}\tilde{\boldsymbol{\omega}}_i & {}_{(i)}\boldsymbol{a}_i \\ {}_{(i)}\dot{\boldsymbol{\omega}}_i^* + {}_{(i)}\tilde{\boldsymbol{\omega}}_i {}_{(i)}\boldsymbol{\omega}_i^* & -{}_{(i)}\tilde{\boldsymbol{a}}_i & \boldsymbol{0} \end{bmatrix}}_{H_i} \underbrace{\begin{bmatrix} \boldsymbol{I}_i^\diamond \\ \boldsymbol{s}_i \\ m_i \end{bmatrix}}_{\boldsymbol{p}_{\mathrm{rb},i}}$$

$$= \underbrace{\begin{bmatrix} H_1 & H_2 & \dots & H_{N_K} \end{bmatrix}}_{H(z,\dot{s},\ddot{s})} \underbrace{\begin{bmatrix} \boldsymbol{p}_1^{\mathrm{T}} & \boldsymbol{p}_2^{\mathrm{T}} & \dots & \boldsymbol{p}_{N_K}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}}_{\boldsymbol{p}_{\mathrm{rb}}} \tag{30}$$

which is now linear in respect to the parameter set $\boldsymbol{p}_{\mathrm{rb}}$, that groups all physical parameters of all limbs of the robot. Since each limb contributes with 1 mass parameter, 3 first-moment parameters and 6 inertiatensor elements, we obtain

for the robot the number of $(1 + 3 + 6) \times N_K$ physical parameters. The contribution of each single parameter to the dynamics is presented by the corresponding column of the matrix H_i. The dimension of p_{rb} has to be reduced for more computational efficiency and identifiability of the dynamics model. In the field of robotics, many researches have been achieved on this subject, especially for serial robots (Khalil & Kleinfinger, 1987; Gauier & Khalil, 1990; Fisette et al., 1996). Recently the problem was also addressed for parallel mechanisms (Khalil & Guegan, 2004; Abdellatif et al., 2005a). Generally, the procedure consists in a first step of grouping the parameters for the open chains. Afterwards, one looks for further parameter reduction that is due to eventually existing closed kinematic loops. In Praxis, the first step is common for serial and parallel robots. For the latter, the structure is subdivided in single serial chains. The second step is achieved of course, only for parallel robots.

The matrices H_i in (30) can be grouped for single serial kinematic chains, such that a recursive calculation:

$$H_i = H_{i-1} L_i + K_i \tag{31}$$

can be achieved. The matrices L_i and K_i are given in (Khalil & Dombre, 2002; Grotjahn & Heimann, 2000). The first step considers in eliminating all parameters $p_{\mathrm{rb},j}$ that correspond to a zero row h_j of H, since they do not contribute to the dynamics. The remaining parameters are then regrouped to eliminate all linear dependencies by investigating H. If the contribution of a parameter $p_{\mathrm{rb},j}$ depends linearly on the contributions of some other parameters $p_{\mathrm{rb},1j}, \ldots, p_{\mathrm{rb},kj}^*$, the following equation holds

$$h_j = \sum_{l=1}^{k} a_{lj} h_{lj} \tag{32}$$

Then $p_{\mathrm{rb},j}$ can be set to zero and the regrouped parameters $p_{\mathrm{rb},lj,\mathrm{new}}$ can be obtained by

$$p_{\mathrm{rb},lj,\mathrm{new}} = p_{\mathrm{rb},lj} + a_{lj} p_{\mathrm{rb},j}^* \tag{33}$$

The recursive relationship given in (31) can be used for parameter reduction. If one column or a linear combination of columns of L_i is constant with respect to the joint variable and the corresponding columns of K_i are zero columns, the parameters can be regrouped. This leads to the rules which are formulated in (Gautier & Khalil, 1990; Khalil & Dombre, 2002) and in (Grotjahn &

Heimann, 2000). The use of MDH-notation is a benefit for applying the reduction rule in an analytical and a straight-forward manner. For revolute joints with variable ϑ_i, the other MDH-parameters are constant. This means that the 9th, the 10th and the sum of the 1st and 4th columns of L_i and K_i comply with the mentioned conditions. Thus, the corresponding parameters $I_{i_{yy}}$, s_{i_x} and m_i can be grouped with the parameters of the antecedent joint $i-1$. For prismatic joints however, the moments of inertia can be added to the carrying antecedent joint, because the orientation between both links remain constant. For a detailed insight, it is recommended to consider (Khalil & Dombre, 2002) and (Grotjahn & Heimann, 2000).

In the case of parallel robots, where the end-effector platform closes the kinematic loops, further parameter reduction is possible. The velocities of the platform joint points B_j and those of the terminal MDH-frames of the respective leg are the same. The masses can be grouped to the inertial parameter of the EE-platform according to steiner's laws (Khalil & Guegan, 2004; Abdellatif et al., 2005a).

3.3 Integration of friction and motor inertia effects

For further accuracy enhancement of the inverse dynamics models, the effects of friction and motor inertia should be considered. Especially the first category is important for control applications (Grotjahn & Heimann, 2002; Armstrong-Hélouvry, 1991; Bona & Indri, 2005). The general case is considered, when friction is modeled in all active as well as in passive joints. The friction is given in the joint space Q, usually as nonlinear characteristics $Q_{f_i}(\dot{q}_i) = f(\dot{q}_i)$ with respect to the joint velocity, i.e.

$$Q_{f_i}(\dot{q}_i) = r_{1_i}\text{sign}(\dot{q}_i) + r_{2_i}\dot{q}_i \qquad (34)$$

The joint losses have to be mapped into the actuation (or control) space. Uniformly to the rigid-body dynamics, the Jourdainian principle of virtual power is recommended. The power balance for friction can be derived as

$$Q_{a,f} = \left(\frac{\partial \dot{q}}{\partial \dot{q}_a}\right)^{\mathrm{T}} Q_f = J^{\mathrm{T}}\left(\frac{\partial \dot{q}}{\partial \dot{s}}\right)^{\mathrm{T}} Q_f \qquad (35)$$

that means: the friction dissipation power in all joints (passive and active) has to be overcome by an equivalent counteracting actuation power. From the latter equation it is clear that the case of classic open-chain robots is restrictive, when the joint-jacobian $\partial \dot{q}/\partial \dot{q}_a$ is equal to the identity matrix. In the more general case of parallel mechanisms, friction in passive joints should not be

neglected like it is almost always assumed in control application for such ro-
bots (Ting et al., 2004; Cheng et al., 2003). The compensation of friction is sim-
pler and more accurate for serial robots, since it can be achieved directly in all
actuated joints. For the parallel counterpart, the compensation of the physical
friction Q_f is only possible indirectly via the projected forces $Q_{a,f}$ to account
for passive joints. Since the latter are usually not equipped with any sensors,
friction compensation in parallel robots is less accurate, which is one of the
typical drawbacks of such robotic systems.

By using friction models that are linear with respect to the friction coefficients,
like (34) it is more or less easy to derive a linear form of (36). The following re-
lationship results:

$$Q_{a,f} = A_f(z, \dot{s}) p_f \qquad\qquad (36)$$

where the friction coefficients are grouped in a corresponding parameter vec-
tor p_f.

The inertial effects of drives and gears can be also considered and integrated in
the dynamics with standard procedures like described in (Sciavicco & Sicili-
ano, 2000; Khalil & Dombre, 2002). One of the advantages provided by parallel
robots is the fact, that the motors are mainly installed on the fixed platform,
such that they do not contribute to the dynamics. This issue remains - at least
for industrial application – exclusive for conventional serial manipulators,
where the motors are mounted on the respective limbs.

3.4 Example: Minimal rigid-body parameters

The illustrative example of minimal rigid-body parameters is considered to
give an interesting comparison between serial and parallel manipulators in
terms of dynamics modeling. The above described uniform approach is ap-
plied for the 6-DOF robots *KUKA KR15* and *PaLiDA*. According to the nota-
tions defined in the former section, the minimal parameters are derived for
both systems. The results are illustrated in Table 1. Despite higher structural
complexity, the minimal parameters of the parallel robot are less numerous
and complex than those of the serial one. The single sub-chains of a parallel
robot are usually identical and have simple structure, which yields identical
and simple-structured parameters for the different chains. The kinematic cou-
pling yields a further parameter reduction as the example demonstrates for
p_6 - p_{10}. The inertial effects of the limbs directly connected to the moving plat-
form are counted to the dynamics of the end-effector by taking
$_{(E)}r_{B_j}^E = \begin{bmatrix} r_{Bx_j} & r_{By_j} & r_{Bz_j} \end{bmatrix}^T$ into account (see also eq. (14)). The derivation of mini-
mal parameters is of a major interest, since they constitute the set of identifi-

able ones (Gautier & Khalil, 1990). Following section discusses the experimental identification of parameters and the implementation of identified inverse models in control.

	KUKA KR15	PaLiDA
p_1	$I_{M_1} + I_{1_{zz}} + I_{2_{yy}} + I_{3_{yy}} + l_1^2(m_2 + m_3)$ $+ m_3 l_2^2 + (m_4 + m_5 + m_6)(l_1^2 + l_2^2 + l_3^2)$	$I_{1_{zz}} + I_{2_{yy}} + I_{3_{zz}}$
p_2	$I_{2_{xx}} - I_{2_{yy}} - l_2^2(m_3 + m_5 + m_6)$	$I_{2_{xx}} + I_{3_{xx}} - I_{2_{yy}} - I_{3_{zz}}$
p_3	$I_{2_{xz}}$	$I_{2_{zz}} + I_{3_{yy}}$
p_4	$I_{M_2} + I_{2_{zz}} + l_2^2(m_3 + m_4 + m_5 + m_6)$	s_{2_y}
p_5	$s_{2_x} + l_2(m_3 + m_4 + m_5 + m_6)$	s_{3_z}
p_6	$I_{3_{xx}} - I_{3_{yy}} + I_{4_{yy}} + 2l_4 s_{4_z}$ $+ (l_4^2 - l_3^2)(m_4 + m_5 + m_6)$	$I_{xx_E} + m_3 \sum_{j=1}^{6} \left(r_{By_j}^2 + r_{Bz_j}^2 \right)$
p_7	$I_{3_{xy}} - l_3 s_{4_z} - l_3 l_4 (m_4 + m_5 + m_6)$	$I_{yy_E} + m_3 \sum_{j=1}^{6} \left(r_{Bx_j}^2 + r_{Bz_j}^2 \right)$
p_8	$I_{3_{xx}} - I_{3_{yy}} + I_{4_{yy}} + 2l_4 s_{4_z}$ $+ (l_3^2 + l_4^2)(m_4 + m_5 + m_6)$	$I_{zz_E} + m_3 \sum_{j=1}^{6} \left(r_{Bx_j}^2 + r_{By_j}^2 \right)$
p_9	$s_{3_x} - l_3(m_4 + m_5 + m_6)$	$s_{z_E} + m_3 \sum_{j=1}^{6} r_{Bz_j}$
p_{10}	$s_{3_y} - s_{4_z} - l_4(m_4 + m_5 + m_6)$	$m_E + 6m_3$
p_{11}	$I_{4_{xx}} - I_{4_{yy}} + I_{5_{yy}}$	–
p_{12}	$I_{4_{zz}} + I_{5_{yy}}$	–
p_{13}	s_{4_y}	–
p_{14}	$I_{5_{xx}} - I_{5_{yy}} + I_{6_{xx}}$	–
p_{15}	$I_{5_{zz}} + I_{6_{xx}}$	–
p_{16}	$s_{5_y} - s_{6_z}$	–
p_{17}	$I_{6_{zz}}$	–
p_{18}	I_{M_3}	–
p_{19}	I_{M_4}	–
p_{20}	I_{M_5}	–
p_{21}	I_{M_6}	–

Table 1. Minimal rigid-body parameter set for the 6-DOF robots *KUKA KR15* and *PaLiDA*.

4. Dynamics Model Identification

A main result retained from the last section is that the inverse dynamics of robotic manipulators can be written as

$$Q_a = Q_{a,rb} + Q_{a,f} = A_{rb}(z, \dot{s}, \ddot{s})p_{rb} + A_f(z, \dot{s})p_f = A(z, \dot{s}, \ddot{s})p$$

with the vector p containing the minimal rigid-body parameters and friction coefficients. With such an LP (*linear in the parameter*)-structure computational efficient linear estimators can be applied for identification. The formulation of the related problem is derived for an experiment producing N data vectors as follows

$$\begin{bmatrix} Q_{a_1} \\ \vdots \\ Q_{a_N} \end{bmatrix} = \underbrace{\begin{bmatrix} A(z_1, \dot{s}_1, \ddot{s}_1) \\ \vdots \\ A(z_N, \dot{s}_N, \ddot{s}_N) \end{bmatrix}}_{\Psi} p + \underbrace{\begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix}}_{\eta} \qquad (37)$$

with the measurement vector Γ, the information or regression matrix Ψ and the error vector η that accounts for disturbances. The solution of the over-determined equation system (37) yields Weighted Least-Squares estimate $\hat{p}_{WLS}$ of the parameter vector (Walter & Pronzato, 1997)

$$\hat{p}_{WLS} = \left(\Psi^T W^{-1} \Psi\right)^{-1} \Psi^T W^{-1} \Gamma \qquad (38)$$

where W is a weight matrix. The classical Least-Squares (LS) estimator results from (38) by setting W to a diagonal matrix with equal entries

$$\hat{p}_{LS} = \left(\Psi^T \Psi\right)^{-1} \Psi^T \Gamma \qquad (39)$$

The quality of the estimation results depends on the so called experiment design that define how and which data has to be collected to guarantee good estimate. Here, two main categories exist: the direct and indirect approach. The first one suggests collecting the data along one single trajectory that excite all parameters. The second approach proposes collecting different measurements in single configurations within the workspace. Each approach has characteristic advantages and drawbacks that depend also on the regarded robot type. It can be stated generally, that the identification procedure for parallel robots is more difficult, because the necessary information about the minimal coordinates, velocities and accelerations can not be directly measured (Abdellatif et al., 2005c). A systematic comparison for both approaches is given in Table 2 as

well as an evaluation of its appropriateness for the studied manipulator types. The main drawback of indirect approaches is the time-consuming data collection procedure, since many measurements are necessary. Moreover, they require high quality measurements to enable handling narrow measurement intervals, otherwise the data quality will be poor (Grotjahn et al., 2004). These drawbacks are eliminated by direct approaches, since we obtain quick and fast identification trajectories (Abdellatif et al., 2005c). The availability of analytical form of the trajectories helps a highly efficient frequency-domain handling of the measurements. On the other hand, optimized trajectories are needed (Swevers et al., 1997) that can not be simply achieved by conventional industrial control setups. Additional programming or hardware interfaces are necessary. This is not the case with the indirect approach because simple PTP-motions with trapezoidal velocity profile are used, which can be directly programmed and executed. Furthermore, the indirect approach enables identification of submodels independently on each other, i.e. friction and rigid-body or inertial and gravitational parameters (Grotjahn et al., 2001; Grotjahn et al., 2004).

	direct approach		indirect approach	
	serial	parallel	serial	parallel
time cost	+	+	–	–
signal processing	+	+	0	–
interface requirement	–	–	+	+
sub-model identification	–	–	+	+
Refs	Swevers et al., 1997	Abdellatif et al., 2005c	Grotjahn et al., 2001	Grotjahn et al., 2004

Table 2. Direct vs. indirect identification approach: appropriateness for serial and parallel industrial robots

The recommended approach depends on the real setup and equipment. If the available control system allows the achievement of arbitrarily complex trajectories, it is recommended to measure optimized trajectories. Otherwise, long measuring processes have to be taken into account, if the indirect approach is chosen. E.g. the identification of friction models for the *KUKA KR15* and for *PaLiDA* required 45 min and 120 min measurement time, respectively.

The validation of the identified dynamics models is achieved by considering reference trajectories that were not used for identification. For the industrial *KUKA KR15* robot, the *ISO*-92833 standard trajectory is validated. The measured torques are compared with those calculated by the identified model. The results are illustrated in Figure 3.
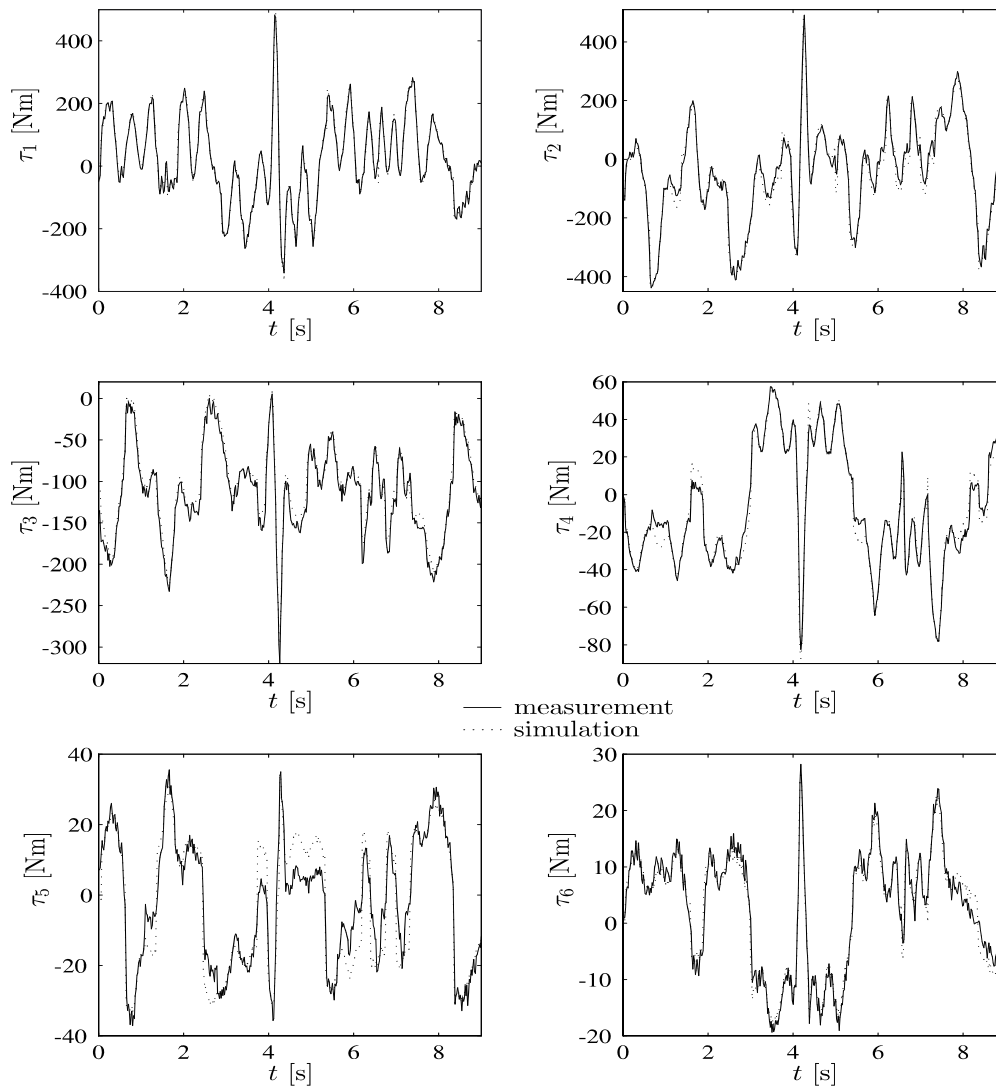
Figure 3. Accuracy validation of identified model of the *KUKA KR15*: Torques for *ISO-9283* trajectory

Unfortunately, standard benchmarks are not defined for parallel robots yet and the *ISO*-92833 violates the workspace of the here studied robot. Thus, an inclined circular motion in the middle of the workspace is chosen as a benchmark test for validating the accuracy of the identified model for *PaLiDA*. The results are shown in Figure 4.
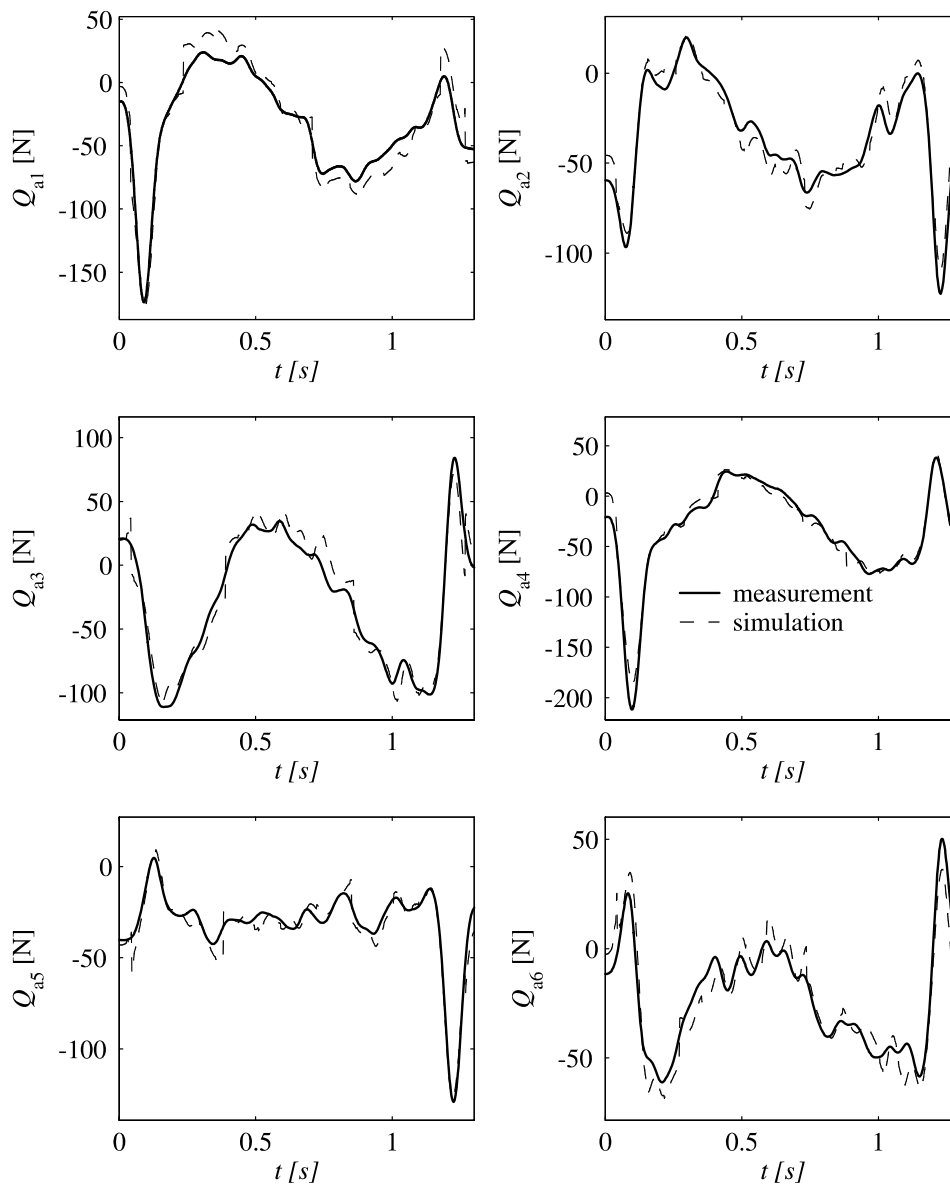
Figure 4. Accuracy validation of identified model of *PaLiDA*: Forces for a circular benchmark trajectory.

For both robot types, very good agreement between model output and experiment is noticeable. Of course, some deviations still remain, since a complete agreement of the model and the reality is quite impossible. Nevertheless, these results are an excellent base for the compensation of nonlinearities of both robots. For place reasons, the values of the identified parameters of the studied systems are not illustrated. We refer though to former publications with deeper insight and more discussion on dynamics identification of robots (Abdellatif et al., 2005b; Abdellatif et al., 2005d; Grotjahn et al., 2004; Grotjahn et al., 2001).

5. Model-based Feedforward control

The basics for implementing model-based feedforward control are now achieved. We have adequate modeled systems with accurate identified parameters. The next challenge is to use the attained knowledge for practical and industrial relevant control. The compensation of nonlinear dynamics can only be performed by using feedforward approaches because there is usually no possibility to change the feedback controller structure provided by the robot manufacturers. Anticipating for possible next commercial technologies, industrial control systems can have also a force/torque interface. In that case, the controller input can be directly used by the robot's operator. However, the standard in industrial applications remains that only the desired path can be changed by the feedforward controller as other interfaces do not exist. The case of parallel robots is a prime example, that conventional and commercial control systems are less adequate for such new industrial application. It is believed, that the use of conventional technology widespread for serial robots or machine tools should be reconsidered. The paradigm of single-joint control do not regard highly nonlinear kinematic coupling and is one of the main reasons, that parallel robots still did not reach their promised potentials in practice. In the following we want to consider both possibilities of direct force/torque input and the position pre-correction. Respecting the scope of this paper, it is more focused on the second case. First nonlinear feedforward control strategies are discussed, that are based on the friction and rigid-body model presented in section 3. Subsequently, compensation methods are presented which use linear models for improvement of path accuracy.

5.1 Nonlinear Compensation Control

5.1.1 Computed Force/Torque Feedforward Control

The computed force/torque feedforward control is one of the most classic approaches in robotics (Sciavicco & Siciliano, 2000; Khalil & Dombre, 2002). A uniform scheme can be given according to the formalism defined in previous sections and is depicted in Fig 5. The global input consists in the minimal coordinates z with an explicit or implicit dependency on time t, to enable the derivation of velocities and accelerations $\dot{s}$ and $\ddot{s}$. Only the nonlinear block $\left(z \rightarrow q_{a,d}\right)$ depends on the robot's type. It consists in the trivial identity transformation for serial robots and the inverse kinematics (14) for parallel manipulators. For both systems, the desired forces (or torques) $Q_{a,d}$ can be only derived from z, $\dot{s}$ and $\ddot{s}$. Of course, such an approach depends on the presence of a motor-current interface to achieve the direct forward feeding of the calculated forces or torques.
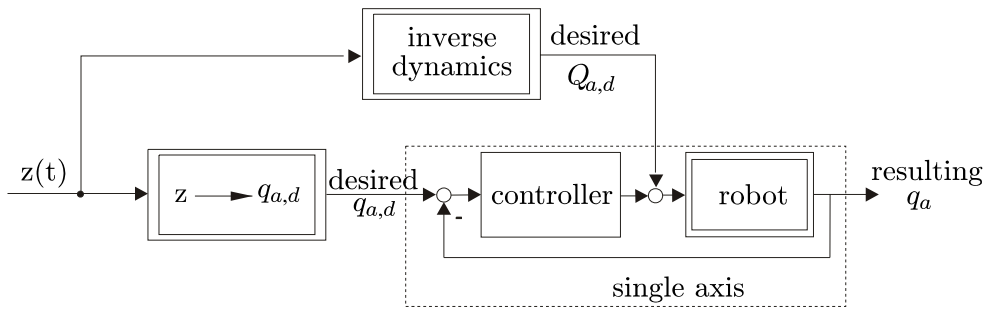
Figure 5. Uniform Scheme of Computed Force/Torque Feedforward Control

The impact of the feedforward compensation of dynamics is depicted for the robot *PaLiDA* in Figure 6.
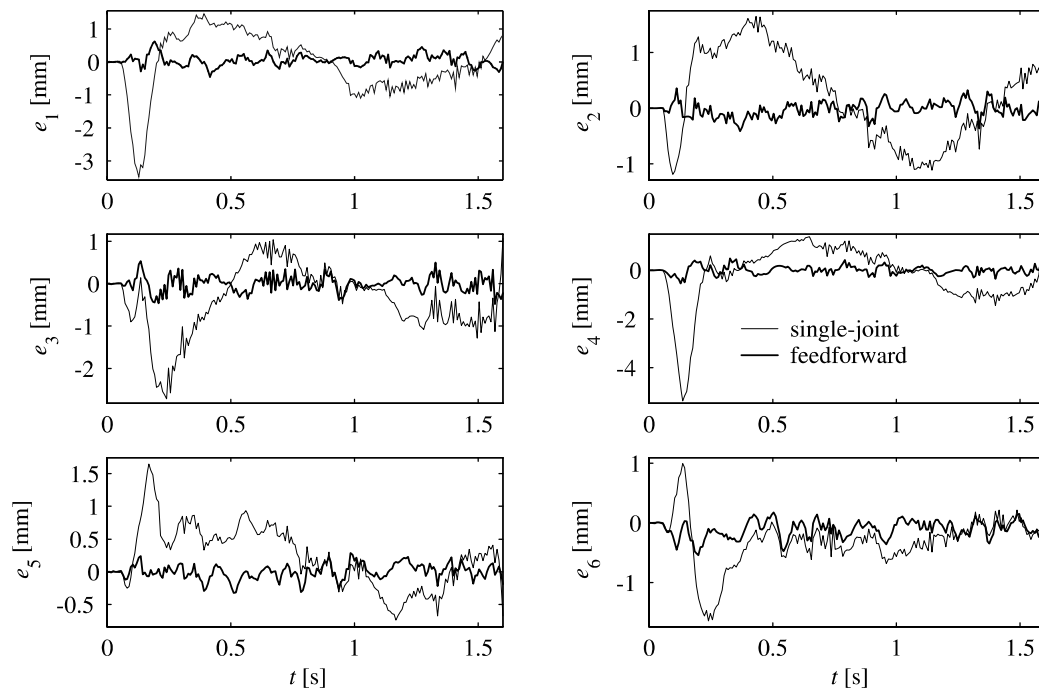
Figure 6. Tracking performance of the parallel robot *PaLiDA*. A Comparison between the single-joint and the feedforward approach.

The same circular motion presented in Figure 4 is investigated. Without doubt, the tracking errors for all six actuators were considerably decreased. As depicted, the control deviations resulting from the use of the simple decentralized single-joint control yields unacceptable errors (about 4 mm while acceleration). According to our experience, the compensation of the nonlinear dynamics is indispensable for operating parallel robots at high velocities and accelerations. Details on related experimental studies can be found in (Abdellatif et al., 2005a). A key role for the control improvement is assumed by the modeling of inverse dynamics and the appropriate identification of the related parameters.

5.1.2 Nonlinear Precompensation

By the absence of motor-current or force/torque interface, a similar approach to the feedforward control can be applied, which we call *nonlinear pre-correction* (see Figure 7). The inverse dynamics model is computed the same way but is now provided to the inverse controller F_R^{-1} that yields the pre-correction terms $\Delta q_{a,d}$. The advantage of nonlinear trajectory pre-correction compared to the computed force/torque method is that one only needs to convey path information to the robotic system. Force/torque information are not necessary. Only a path interface is necessary. The approach is applicable to standard industrial robot systems since such an interface is usually provided for the integration of external sensor information into the control circuit. The proposed approach was implemented within the KRC1 standard control for the robot *KUKA KR15*. The improvements of tracking errors are depicted in Figure 8. The disadvantage of the nonlinear pre-correction is the necessity of a reliable controller model to be inverted. If there is no knowledge about the controller, experimental identification of the controller dynamics has to be carried out (Grotjahn & Heimann, 2002).
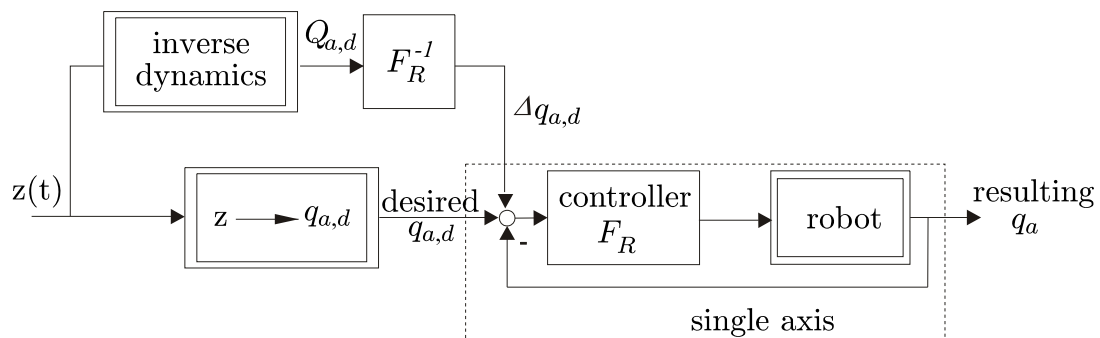
Figure 7. Uniform scheme of nonlinear pre-correction control.
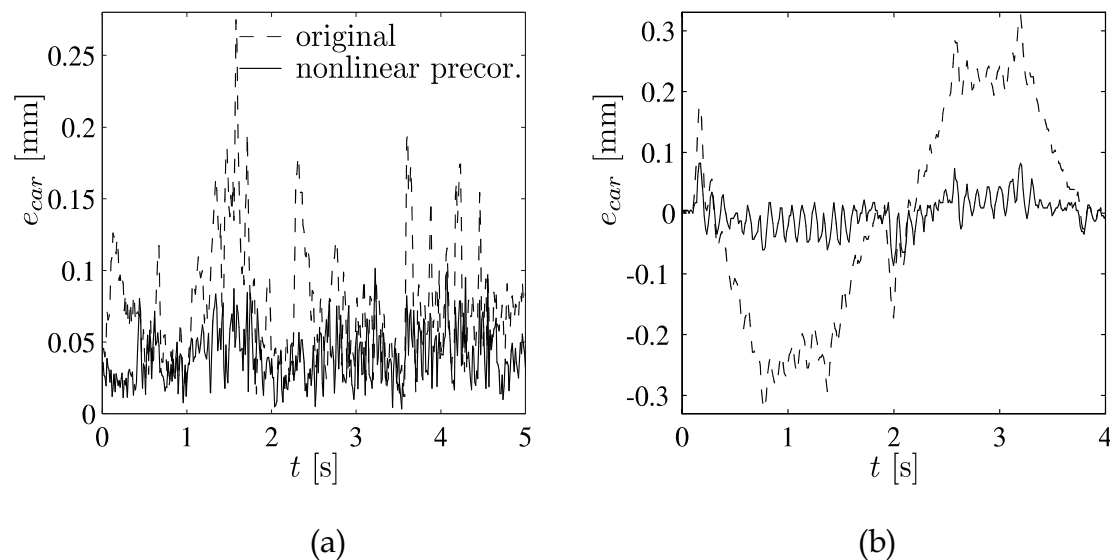
Figure 8.Path deviations of the *KUKA KR15*: (a) *ISO*-9283 trajectory; (b) single motion of the second joint.

5.2 Feedforward Control by Linear Model

Although the linear models disregard nonlinearities, they can be used to their compensation by taking into account actual path deviations. The advantage is that arbitrary effects can be compensated, even effects which are not included in the complex nonlinear model. This is done by 'iterative learning' algorithm that is presented in the following. Alternatively 'training' of a feedforward joint controller is explained. Both approaches are based on the actuation variables. Thus, their implementation is similar for parallel and serial robots.

5.2.1 Iterative Linear Learning

Learning control is based on the idea that measured path deviations can be used to adjust the trajectory that is transmitted to the controller so that the actual trajectory comes closer to the originally desired one (Moore, 1999; Longman, 2000). Thereby, the path accuracy is improved by iterative correction (see Figure 9).
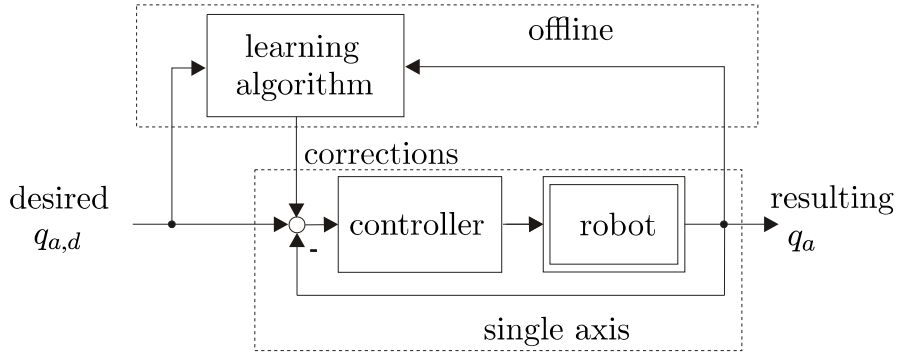
Figure 9. Principle of Iterative Learning Control.

For learning implementation, the robot is already under feedback-control, such that its closed-loop dynamics can be described in a linear discrete-time form by the state-space equations:

$$x(k+1) = A(k)x(k) + B(k)u(k) + w_1(k)$$
$$y(k) = C(k)x(k) + w_2(k) \qquad (40)$$

with u being the input and y being the output. It is assumed that w_1 represents some deterministic disturbance that appears every repetition and that w_2 is the measurement disturbance. The aim of Learning is to change the command input every trial j using the learning control law:

$$u_{j+1}(k) = f_L\big(u_j(k), y_j(k), y_d(k)\big) \qquad (41)$$

such that the desired trajectory y_d is tracked. Iterative learning control is called linear, when the learning law f_L makes an iterative change in the input that is a linear combination of the error $e_j = y_j - y_d$ measured in the previous repetition and the last input sequence

$$u_{j+1} = u_j + Le_j \Rightarrow q_{a,j+1} = q_{a,j} + L\big(q_{a,j} - q_{a,d}\big) \qquad (42)$$

The design of the learning gain matrix L has to achieve desired properties. It is simple to derive the iterative error dynamics as

$$e_{j+1} = (I - PL)e_j \qquad (43)$$

where I is the identity matrix and

$$
P = \begin{bmatrix} g(1) & & & \\ g(2) & g(1) & & \\ \vdots & \vdots & \ddots & \\ g(N) & g(N-1) & \cdots & g(1) \end{bmatrix} \tag{44}
$$

if a linear time-invariant actuator error dynamics is assumed, with an impulse response g. N is the length of the actual trajectory or input. Choosing L to be P^{-1} would lead to immediate zero tracking. The inversion of P is equivalent to an inversion of the joint's closed-loop. As it can not be guaranteed that the identified system is phase-minimum, an exact inversion is not always possible. In that case the resulting corrected trajectory would include unacceptable oscillations. To avoid this, many methods can be used, like filtering suggested in (Norrlöf & Gunnarsson, 2002; Longman, 2000) etc. We propose here two methods. The first one is an extended LS-method or the Inverse-Covariance Kalman-Filter (Grotjahn & Heimann, 2002). It requires a good quality of measurement signals, wich is the case for standard industrial robot. In the case when measurements of the actuator variables are importantly corrupted by noise, the inverse of P is not adequate. Alternatively, a contraction mapping approach can be used (Longman, 2000), where the learning matrix is defined by: $L = \Phi_1 P^{\mathrm{T}}$ for a learning gain Φ_1 that has to be chosen. The dynamics and therefore an estimate of the impulse response g of the closed-loop dynamics can be identified by applying standard procedures (Ljung, 1999).

The main advantage of learning control is the fact that it can compensate for influences and systematic deviations that are not captured by the model. This holds not only for non-modeled effects by the linear models used for learning, but also for effects not even reflected by the presented complex nonlinear model. Another advantage is that the actual deviations of end-effector position and orientation can be taken into account if they are measurable. The main disadvantage, however, is that every small change of the desired trajectory necessitates a new learning process. Each learning process needs some iterations until convergence, which can be undesirable by robot operators.

5.2.2 Training of Feedforward Controller

In order to avoid the disadvantages of learning control described in the previous section, Lange and Hirzinger proposed to '*train*' a feedforward controller from the learnt behavior (Lange & Hirzinger, 1996). The feedforward controller model has to be identified by using e.g. the extended LS-method or the Inverse-Covariance Kalman-Filter (Grotjahn & Heimann, 2002). The scheme of training is given by Fig 10.
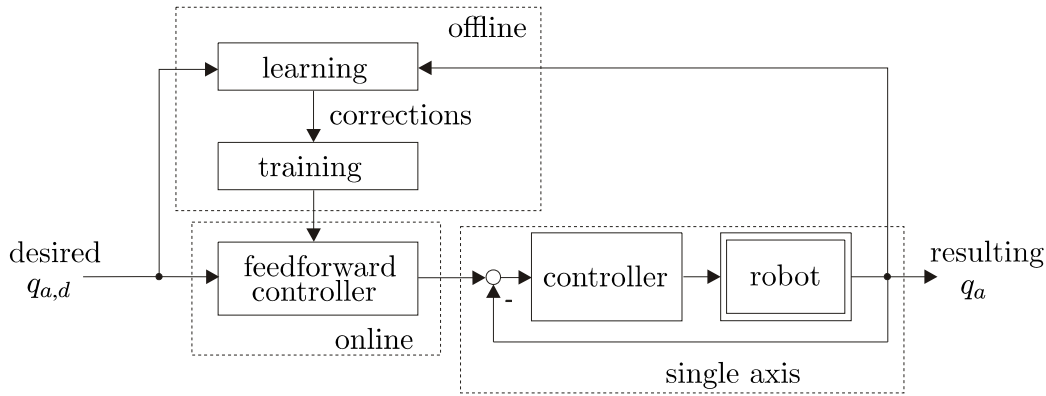
Figure 10. Training a Feedforward Controller.

As a matter of principal, a feedforward controller has the advantage that it can be calculated in real-time. Therefore, it can be implemented in commercial controls. Furthermore, a powerful feedforward controller offers the possibility to transfer the learned behavior to similar trajectories. Consequently, small trajectory changes would not require a new learning process. The choice of the controller structure is a key issue. A fundamental condition is that the model can satisfactorily reproduce the learned behavior. In (Lange & Hirzinger, 1994), linear models like

$$q_a(k) = q_{a,d}(k) + \sum_{l=m_t}^{m} r(l)\left(q_{a,d}(k+l) - q_{a,d}(k)\right) \tag{45}$$

were proposed. This approach, however, cannot compensate nonlinear effects, like friction. Fig 11 shows that the identification of the linear feedforward controller leads to a mixture of two effects. In addition to inertial influences, friction has a large impact at the beginning of the motion. Therefore, the corrections of the linear controller are too small in the beginning and too large at later zero-crossing of velocity. To improve this, the approach is extended by another term:

$$q_a(k) = q_{a,d}(k) + \sum_{l=m_t}^{m} r(l)\left(q_{a,d}(k+l) - q_{a,d}(k)\right)$$

$$\tag{46}$$

$$+ \sum_{l=p_t}^{p} s(l)\,\text{sign}\left(\dot{q}_{a,d}(k+l) - \dot{q}_{a,d}(k)\right)$$

This auxiliary summand is suited to separate friction from inertial influences. After its consideration, learned corrections are reproduced much better than by the linear model. As expected, better tracking behavior can be obtained (see Fig 11). Although the performance of 'training' remains less than that of 'learn-

ing', it seems to be an alternative that combines good results with practical simplicity. The values m_t, m, p_t and p can be chosen by operators corresponding to the desired error dynamics. Using only positive values is equivalent to the consideration of posterior or future errors, only. It is recommended to chose negative m_t and s_t, such that the influence of previous dynamics, like high accelerations or velocities, can be taken into account.
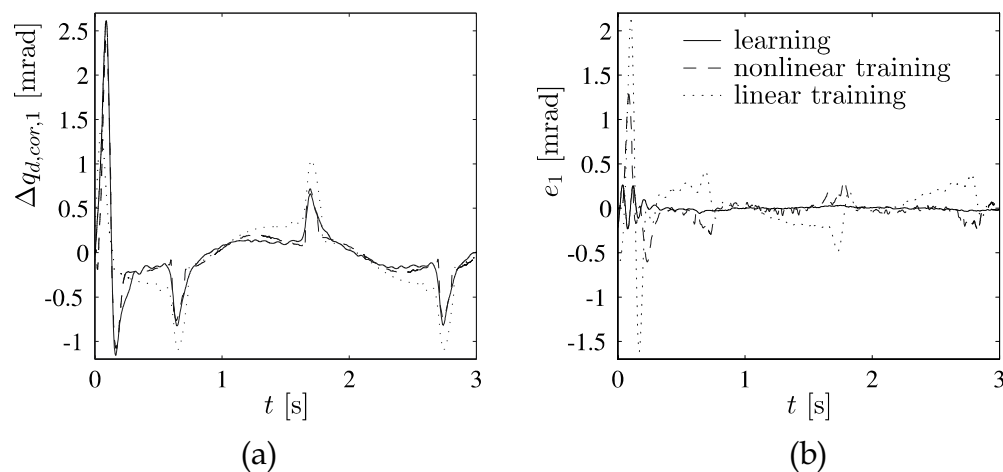
Figure 11. Behaviour of the first axis of the *manutec-r15* for a vertical circle:
(a) learned and trained corrections; (b) comparison of resulting tracking errors

5.2.3 Application to serial robots

The presented compensation methods are investigated by experimental application to a classic serial manipulator: the *manutec-r15*. It is emphasized in the following on the resulting performance and on the applicability. Several different test trajectories are used. Here, the results are explained by regarding two trajectories: a vertical circle and a tracking of planar corner in the x-y-plane with a change of orientation by 90°.
The vertical circle has a diameter of 40 cm and a path velocity of 0.6 m/s. This means that the circle is approximately completed after 2.1 s. The cartesian path deviations are depicted in Figure 12. It shows some general results which can be reproduced for all tested trajectories. The *'training'* yields the worst results of all methods. The nonlinear pre-correction is much better. Learning Control yields even further improvements. In order to numerically evaluate the results, the root mean square (RMS) of the cartesian errors are evaluated for the trajectories and for the different approaches. For all investigated trajectories, *'learning'* leads to a decrease of at least 80 % after four iterations. The nonlinear pre-correction reduces the criterion by at least 60 %, whereas the *'training'* leads to a minimum reduction of 35 %. Figure 12 depicts the tracking of the corner while changing the orientation by 90° for the *manutec-r15* robot. Although the

path velocity is only 0.02 m/s, joint velocities and accelerations are quite high. Therefore, the couplings between different links have strong impact. These effects can not be compensated by the decoupled '*trained*' feedforward joint controllers.
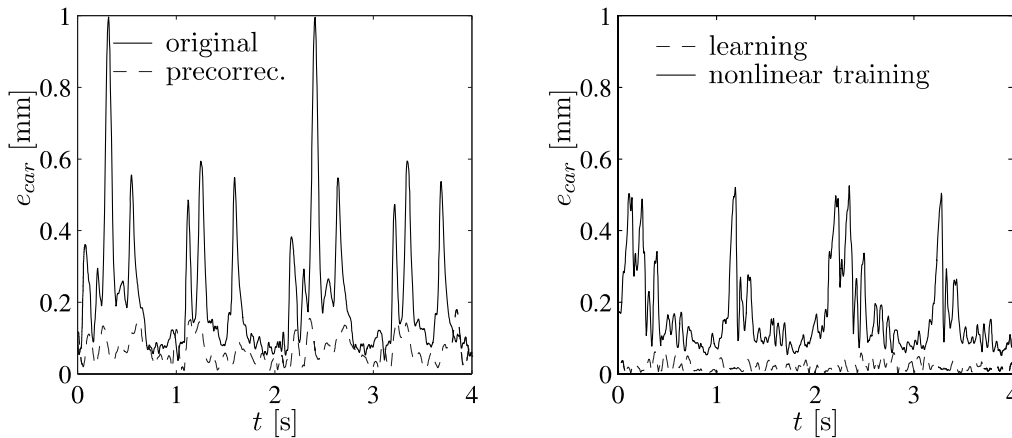
Figure 12. Comparison of cartesian path errors for a vertical circle of the *manutec-r15*

Nonlinear pre-correction is the only approach which combines efficient compensation of nonlinear deviations with practical applicability. '*Learning*' yields better results, but robustness and stability properties have to be improved for the integration in standard industrial controls. The proposed nonlinear '*training*' combines simplicity with applicability but is only suitable for slow trajectories for which couplings between the different links have only low impact. An extended experimental investigation can be found in (Grotjahn & Heimann, 2002).
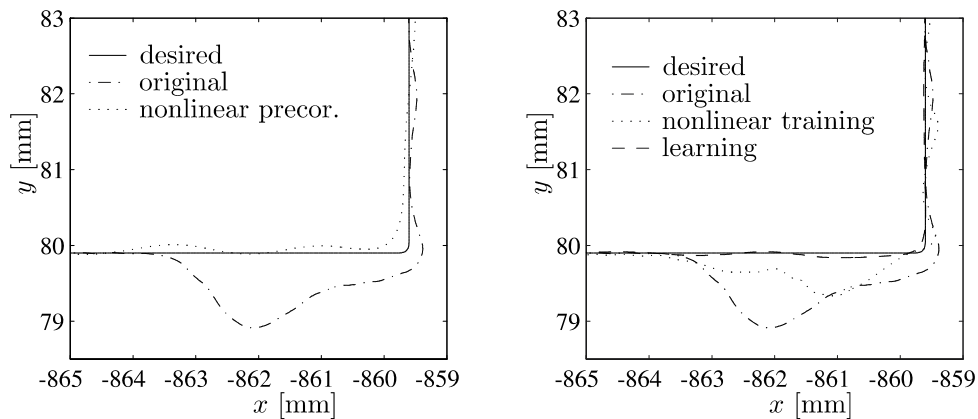
Figure 13. Tracking a Corner by the *Manutec-r15* Robot. Experimental Comparison of Different Feedforward Controllers

5.2.4 Application to parallel robots

In analogy to the case of serial robots, the compensation techniques were implemented to the parallel manipulator *PaLiDA*. It is important to notice, that the feedforward compensation techniques are all based on the actuation variables, for serial, as well as for parallel mechanisms. The procedure of experimental investigation and evaluation is very similar to that mentioned in the previous section. Figure 14 shows the improvement of tracking for the here considered robot. We compare the performances of simple single-joint control, Feedforward control and Learning control for a circular and a quadratic motion. The range of investigated velocity is here much higher than the case of serial robots. The average end-effector's velocity is equal to 1.5 ms^{-1} and 2 ms^{-1} for the circular and quadratic motion, respectively.
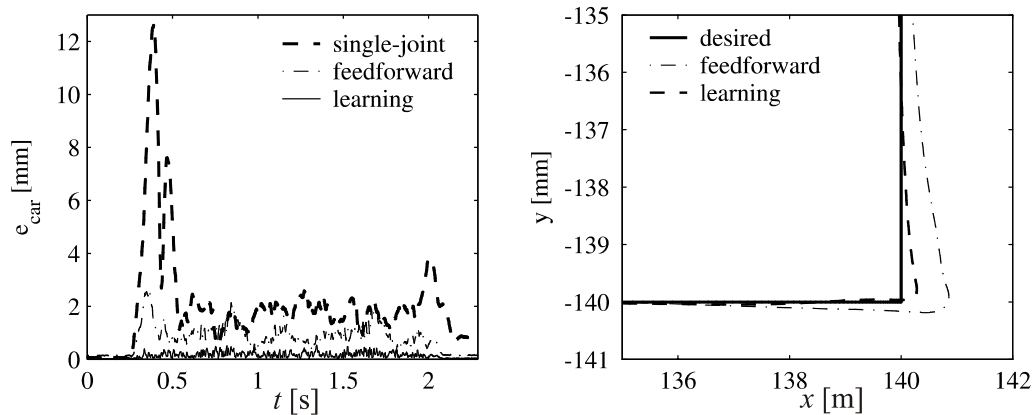
Figure 14. Comparison of Cartesian Path Errors for the Parallel Robot *PaLiDA*. Left: Circle Motion, Right: Quadratic Motion.

The feedforward control helps decreasing the cartesian RMS-error of about 60 %. Learning control decreased the errors of at least 90 %, which is very satisfactory. Such improvement can be clearly observed in Figure 14. Conventional control strategies are not acceptable for operating robots in an accurate manner. The Integration of model-based feedforward compensators, such inverse dynamics or learning controller yield impressive improvement of tracking accuracy.

6. Conclusions

This chapter presented a uniform and coherent methodology for model-based control of industrial robots. To take account of the technological evolution over the last years, classic approaches were extended to the class of parallel kinematic manipulators or parallel robots. A revision of the basics is necessary. Many assumptions that became common due to the reputation of classic open-chain robots were highlighted and revised. This is necessary to be able of developing uniform approaches for handling serial and parallel robots. The idea of this work was to exploit the similarities and to consider the differences between two types. The similarities can be provided by the same modules (calculation, control, etc.). The differences are considered by interfaces (transformations etc.) that account for robot inherent properties.

Already at the basic level of modeling the kinematics and dynamics, the uniformity of the methods can be achieved by considering the generalized coordinates and velocities to be the formal conjunction of serial and parallel robots. It is than possible to apply e.g. generic algorithms and efficient calculation of the inverse dynamics, such that the presented solutions remain valid for a wide class of robots. This was also the case for developing identification strategies of the model parameters. It was demonstrated, that with light adaptation, the same algorithms and experimental strategies can be applied for serial robots and for parallel manipulators.

In the praxis of control, it is the type of the control system that is more crucial for successful implementation, rather than the robot structure. If a force/torque interface is provided, all feedforward strategies can be applied in the same way for parallel and serial robots, since the desired motions are given. If only a position interface is supplied, it is practicable to use correction techniques at actuator levels. The chapter presented nonlinear and linear approaches. The nonlinear pre-correction techniques are recommended for typical industrial control systems and have demonstrated impressive performance. Iterative learning and training algorithms offer the possibilities to use computational efficient linear models. Like substantiated by experimental results the improvement of the control accuracy was investigated for serial and parallel robots.

7. References

Abdellatif, H.; Grotjahn, M. & Heimann, B. (2005a). High efficient dynamics calculation approach for computed-force control of robots with parallel structures, *Proceedings of 44th IEEE Conference on Decision and Control and the 2005 European Control Conference (CDC-ECC05)*, pp. 2024-2029, Seville, 2005.

Abdellatif, H.; Heimann, B. & Grotjahn, M. (2005b). Statistical approach for bias-free identification of a parallel manipulator affected with high measurement noise,

Proceedings of the 44th IEEE Conference on Decision and Control (CDC-ECC05), pp. 3357-3362, Seville, 2005.

Abdellatif, H.; Heimann, B. & Holz, C. (2005c). Time-effective direct dynamics identification of parallel manipulators for model-based feedforward control, *Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2005),* pp. 777-782, Monterey, 2005.

Abdellatif, H.; Heimann, B. & Holz, C. (2005d). Applying efficient computation of the mass matrix for decoupling control of complex parallel manipulators, *Preprints of the 16th IFAC World Congress,* Prag, 2005.

Angeles, J. (2003). *Fundamentals of Robotic Mechanical Systems*, Springer, New York, Berlin, Heidelberg, 2003.

Armstrong-Hélouvry, B. (1991). *Control of Machines with Friction*, Kluwer Academic Publishers, Boston.

Bremer, H. (1988). *Dynamik und Regelung mechanischer Systeme,* Teubner, Stuttgart.

Bona, B. & Indri, M. (2005). Friction compensation in robotics: an overview, *Proceedings of the 44th IEEE Conference on Decision and Control and the 2005 European Control Conference (CDC-ECC05),* pp. 4360-4367, Seville, 2005.

Bonev, I. (2002). *Geometric analysis of parallel mechanisms,* Faculté des études supérieures de l'Université Laval.

Cheng, H.; Kuen, Y. & Li, Z. (2003). Dynamics and control of redundantly actuated parallel manipulators. *IEEE/ASME Transactions on Mechatronics, 8, 4,* (2003) 483-491.

Fisette, P.; Raucent, B. & Samin, J. C. (1996). Minimal Dynamic Characterization of Tree-Like Multibody Systems. *Nonlinear Dynamics*, 9, 1-2 (1996) 165-184.

Gautier, M. & Khalil, W. (1990). Direct calculation of minimum set of inertial parameters of serial robots. *IEEE Transactions on Robotics and Automation*, 6, 3 (1990) 368-373.

Gosselin, C. & Angeles, J. (1990). Singularity analysis of closed-loop kinematic chains. *IEEE Transactions on Robotics and Automation*, 6, 3, (1990) 281-290.

Grotjahn, M. & Heimann, B. (2000). Determination of dynamic parameters of robots by base sensor measurements, *Proceedings of the sixth IFAC Symposium on Robot Control (SYROCO),* Vienna, 2000.

Grotjahn, M.; Daemi, M. & Heimann, B. (2001). Friction and rigid body identification of robot dynamics. *International Journal of Solids and Structures, 38,* (2001) 1889-1902.

Grotjahn, M. & Heimann, B. (2002). Model-based feedforward control in industrial robotic. *International Journal of Robotics research, 21, 1,* (2002) 99-114.

Grotjahn, M.; Heimann, B. & Abdellatif, H. (2004). Identification of friction and rigid-body dynamics of parallel kinematic structures for model-based control. *Multibody System Dynamics, 11, 3,* (2004) 273-294.

Khalil, W. & Kleinfinger, J. (1986). A new geometric notation for open and closed-loop robots, *Proceedings of the 1986 IEEE International Conference on Robotics and Automation,* pp. 1174-1179, San Francisco, 1986.

Khalil, W. & Kleinfinger, J.-F. (1987). Minimum Operations and Minimum Parameters of the Dynamic Models of Tree Structure Robots. *IEEE Transactions of Robotics and Automation, 3, 6* (1987) 517-526.

Khalil, W. & Dombre, E. (2002). *Modelling, Identification and Control of Robots,* Hermes, London

Khalil, W. & Guegan, S. D. (2004). Inverse and direct dynamics modeling of gough-stewart robots. *IEEE Transactions on Robotics,* 20, 4, (2004) 754-762.

Lange, F. & Hirzinger, G. (1994). Learning to improve the path accuracy of position controlled robots, *Proceedings of the Conference on Intelligent Robots and Systems,* pp. 494-501, Munich, 1994.

Lange, F. & Hirzinger, G. (1996). Learning of a controller for non-recurring fast movements. *Advanced Robotics,* 10, 2 (1996) 229-244.

Ljung, L. (1999). *System Identification: Theory for the User,* Prentice-Hall, New Jersey.

Longman, R. W. (2000). Iterative learning control and repetitive leraning control for engnieering practice. *International Journal of Control,* 73, 10, (2000) 930-954.

Meirovitch, L. (1970). *Methods of Analytical Dynamics,* McGraw-Hill, New York.

Merlet, J.-P. (2000). *Parallel Robots,* Kluwer Academic Publishers, Dordrecht.

Moore, K. L. (1999). Iterative learning control: An expository overview. *Applied and Computational Control, Signals and Circuits,* 1, (1999) 151-214.

Norrlöf, M. & Gunnarsson, S. (2002). Experimental Results of Some Classical Iterative Learning Control Algorithmus. *IEEE Transactions on Robotics and Automation,* 18, 4 (2002) 636-641.

Sciavicco, L. & Siciliano, B. (2000). *Modeling and Control of Robot Manipulators,* Springer, London.

Swevers, J.; Gansemann, C.; Tükel, D.; Schutter, J. d. & Brussel, H. v. (1997). Optimal robot excitation and identification. *IEEE Transactions on Robotics and Automation,* 13, 5, (1997) 730-740.

Ting, Y.; Chen, Y.-S. & Jar, H.-C. (2004). Modeling and control for a gough-stewart platform cnc-mashine. *Journal of Robotic Systems,* 21, 11, (2004) 609-623.

Tsai, L.-W. (1999). *Robot Analysis,* Wiley-Interscience, New York.

Walter, E. & Pronzato, L. (1997). *Identification of Parametric Models form Experimental Data,* Springer, London.

Parallel Manipulators with Lower Mobility

Raffaele Di Gregorio

1. Introduction

Parallel manipulators with lower mobility (LM-PMs) are multi-loop mechanisms with less than six degrees of freedom (dofs). This type of manipulators has attracted the attention both of academic researchers and of industries since the early appearance of the DELTA robot (Clavel 1988).
The DELTA robot showed that, when the manipulation task requires less than six dofs, the use of an LM-PM may bring some advantages (simple architecture of the machine, very fast machine, etc.) that are added to the known appealing features (high stiffness, good positioning precision, etc) of parallel manipulators (PMs).Planar motions, translational motions and spherical motions are important examples of motion tasks that require less than six dofs and are often necessary in industrial applications. Each of these types of motion has generated a class of LM-PMs. So, today, there is a great variety of planar PMs (PPMs), of translational PMs (TPMs) and of spherical PMs (SPMs).
This chapter attempts to provide a unified frame for the study of this type of machines together with a critical analysis of the vast literature about them.
The chapter starts with the classification of the LM-PMs, and, then, analyzes the specific subjects involved in the functional design of these machines. Special attention is paid to the definition of the limb topology, the singularity analysis and the discussion of the characteristics of some machines.

2. Classification of the Parallel Manipulators with Lower Mobility

Addressing the problem of classifying manipulators is neither a useless nor a trivial task. Indeed, an exhaustive classification is able to guide the designer towards the technical answers he is looking for.
Lower-mobility manipulators (LM-M) can be classified according to the type of motion their end effector performs by using the group theory (Hervé 1978, 1999). The set of rigid-body displacements (motions), {D}, is a six-dimensional group which, in addition to the identity subgroup, {E}, that corresponds to absence of motion, contains the following ten motion subgroups with dimension greater than zero and less than six (the generic element of a displacement sub-

group can be represented by the screw identifying the finite or infinitesimal motion belonging to the subgroup; the dimension of the subgroup is the number of independent scalar parameters that, in the analytic expression of the generic-element's screw, must be varied to generate the screws of all the elements of the subgroup):

(a) Subgroups of dimension 1:

(a.1) <u>linear translation subgroup</u>, {T(**u**)}, that collects all the translations parallel to the unit vector **u**. As many {T(**u**)} as unit vectors, **u**, can be defined. The identity subgroup {E} is included in all the {T(**u**)}. A prismatic pair (hereafter denoted with P) with sliding direction parallel to **u** physically generates the motions of {T(**u**)}.

(a.2) <u>revolute subgroup</u>, {R(O, **u**)}, that collects all the rotations around an axis (rotation axis) passing through point O and parallel to the unit vector **u**. As many {R(O, **u**)} as rotation axes, (O, **u**), can be defined. The identity subgroup {E} is included in all the {R(O, **u**)}. A revolute pair (hereafter denoted with R) with rotation axis (O, **u**) physically generates the motions of {R(O, **u**)}.

(a.3) <u>helical subgroup</u>, {H(O, **u**, p)}, that collects all the helical motions with axis (O, **u**) and finite pitch p that is different from zero and constant. As many {H(O, **u**, p)} as sets of helix parameters, (O, **u**, p), can be defined. The identity subgroup {E} is included in all the {H(O, **u**, p)}. A helical pair (hereafter denoted with H) with helix parameters (O, **u**, p) physically generates the motions of {H(O, **u**, p)}.

(b) Subgroups of dimension 2:

(b.1) <u>planar translation subgroup</u>, {T($\mathbf{u}_1$, $\mathbf{u}_2$)}, that collects all the translations parallel to a plane perpendicular to $\mathbf{u}_1 \times \mathbf{u}_2$ where $\mathbf{u}_1$ and $\mathbf{u}_2$ are two orthogonal unit vectors. As many {T($\mathbf{u}_1$, $\mathbf{u}_2$)} as unit vectors $\mathbf{u}_1 \times \mathbf{u}_2$ can be defined. The identity subgroup {E} and all the linear translation subgroups {T(**v**)} with **v** equals to $a\mathbf{u}_1 + \mathbf{u}_2\sqrt{1-a^2}$ are included in {T($\mathbf{u}_1$, $\mathbf{u}_2$)}. Two prismatic pairs in series, whose sliding directions are respectively parallel to two independent vectors that are linear combination of $\mathbf{u}_1$ and $\mathbf{u}_2$, physically generate the motions of {T($\mathbf{u}_1$, $\mathbf{u}_2$)}.

(b.2) <u>cylindrical subgroup</u>, {C(O, **u**)}, that collects all the motions obtained by combining a rotation around a rotation axis (O, **u**) and a translation parallel to the unit vector **u**. As many {C(O, **u**)} as (O, **u**) axes can be defined. The subgroups {E}, {T(**u**)}, {R(O, **u**)} and {H(O, **u**, p)} are all included in {C(O, **u**)}. A cylindrical pair (hereafter denoted with C) or, which is the same, a revolute pair with rotation axis (O, **u**) in series with a prismatic pair with sliding direction parallel to **u** physically generate the motions of {C(O, **u**)}.

(c) Subgroups of dimension 3:

(c.1) spatial translation subgroup, {T}, that collects all the spatial transla-
tions. Only one subgroup {T} can be defined. The identity subgroup {E},
all the {T($\mathbf{u}$)} subgroups and all the {T($\mathbf{u}_1$, $\mathbf{u}_2$)} subgroups are included in
{T}. Three prismatic pairs in series whose sliding directions are respec-
tively parallel to three independent unit vectors, $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$, physi-
cally generate the motions of {T}.

(c.2) spherical subgroup, {S(O)}, that collects all the spherical motions with
center O. As many {S(O)} as O points can be defined. The identity sub-
group {E} and all the {R(O, $\mathbf{u}$)} subgroups are included in {S(O)}. A
spherical pair (hereafter denoted with S) or, which is the same, three
revolute pairs in series with rotation axes that intersect one another in O
physically generate the motions of {S(O)}.

(c.3) planar subgroup, {G($\mathbf{u}_1$, $\mathbf{u}_2$)}, that collects all the planar motions with
motion plane perpendicular to $\mathbf{u}_1 \times \mathbf{u}_2$ where $\mathbf{u}_1$ and $\mathbf{u}_2$ are two orthogo-
nal unit vectors. As many {G($\mathbf{u}_1$, $\mathbf{u}_2$)} as unit vectors $\mathbf{u}_1 \times \mathbf{u}_2$ can be de-
fined. The subgroups {E}, {T($\mathbf{u}_1$, $\mathbf{u}_2$)}, {R(O, $\mathbf{u}_1 \times \mathbf{u}_2$)} and {T($\mathbf{v}$)} with $\mathbf{v}$
equals to $a\mathbf{u}_1 + \mathbf{u}_2 \sqrt{1-a^2}$ are included in {G($\mathbf{u}_1$, $\mathbf{u}_2$)}. A PPR kinematic
chain where the sliding directions of the two prismatic pairs are respec-
tively parallel to two independent vectors that are linear combination of
$\mathbf{u}_1$ and $\mathbf{u}_2$, and the revolute-pair axis is orthogonal both to $\mathbf{u}_1$ and to $\mathbf{u}_2$
physically generates the motions of {G($\mathbf{u}_1$, $\mathbf{u}_2$)}.

(c.4) pseudo-planar subgroup, {Y($\mathbf{u}_1$, $\mathbf{u}_2$, p)}, that collects all the motions
obtained by combining a planar translation belonging to {T($\mathbf{u}_1$, $\mathbf{u}_2$)} with
a helical motion belonging to {H(O, $\mathbf{u}_1 \times \mathbf{u}_2$, p)}. As many {Y($\mathbf{u}_1$, $\mathbf{u}_2$, p)} as
sets of parameters ($\mathbf{u}_1 \times \mathbf{u}_2$, p) can be defined. The subgroups {E}, {T($\mathbf{u}_1$,
$\mathbf{u}_2$)}, {H(O, $\mathbf{u}_1 \times \mathbf{u}_2$, p)} and {T($\mathbf{v}$)} with $\mathbf{v}$ equals to $a\mathbf{u}_1 + \mathbf{u}_2 \sqrt{1-a^2}$ are in-
cluded in {Y($\mathbf{u}_1$, $\mathbf{u}_2$, p)}. A RRH kinematic chain where the axes of the
two revolute-pairs and the helical-pair's axis are all parallel to one an-
other and orthogonal both to $\mathbf{u}_1$ and to $\mathbf{u}_2$ physically generates the mo-
tions of {Y($\mathbf{u}_1$, $\mathbf{u}_2$, p)}.

(d) Subgroups of dimension 4:

(d.4) Schoenflies subgroup, {X($\mathbf{u}_1$, $\mathbf{u}_2$)}, that collects all the motions ob-
tained by combining a planar translation belonging to {T($\mathbf{u}_1$, $\mathbf{u}_2$)} with a
cylindrical motion belonging to {C(O, $\mathbf{u}_1 \times \mathbf{u}_2$)}. As many {X($\mathbf{u}_1$, $\mathbf{u}_2$)} as
unit vectors $\mathbf{u}_1 \times \mathbf{u}_2$ can be defined. The subgroups {E}, {T}, {G($\mathbf{u}_1$, $\mathbf{u}_2$)},
{Y($\mathbf{u}_1$, $\mathbf{u}_2$, p)}, {T($\mathbf{u}_1$, $\mathbf{u}_2$)}, {C(O, $\mathbf{u}_1 \times \mathbf{u}_2$)}, {H(O, $\mathbf{u}_1 \times \mathbf{u}_2$, p)} and {T($\mathbf{v}$)} with $\mathbf{v}$
equals to $a\mathbf{u}_1 + \mathbf{u}_2 \sqrt{1-a^2}$ are included in {X($\mathbf{u}_1$, $\mathbf{u}_2$)}. A RRC kinematic
chain where the axes of the two revolute pairs and the cylindrical-pair's

axis are all parallel to one another and orthogonal both to $\mathbf{u}_1$ and to $\mathbf{u}_2$ physically generates the motions of $\{X(\mathbf{u}_1, \mathbf{u}_2)\}$.

According to this (Rico et al. 2006), the set of the LM-Ms can be separated into two subsets: (i) the subset of the pure-motion LM-Ms and (ii) the subset of the mixed-motion LM-Ms. The first subset collects all the LM-Ms whose end effector exhibits motions that belong to only one out of the ten motion subgroups of $\{D\}$, whereas the second one collects all the other LM-Ms.

The pure-motion LM-Ms can be further spread into ten pure-motion subsets: one for each pure motion identified by the ten subgroups of $\{D\}$. In (Hervé 1978, 1999), a kinematic chain is called mechanical bond when it connects one rigid body (end effector) to another (frame) so that the relative motion between end effector and frame is constrained. A mechanical bond is called mechanical generator when all the allowed relative motions between end effector and frame belong to only one of the ten subgroups of $\{D\}$.

The nature of an LM-M can be identified by analysing its workspace, $\{W\}$ (the workspace is the connected set of poses (positions and orientations) the end effector can assume without disassembling the LM-M). In fact, if any couple of poses belonging to $\{W\}$ defines an end-effector motion that belongs to the same motion subgroup of $\{D\}$, then the LM-M is a pure-motion LM-M, otherwise it is a mixed-motion LM-M. Hereafter, if a set of motions, $\{M\}$, only collects the motions identified by all the couples of poses that belong to the same connected set of poses, $\{P\}$, then it will be said that "$\{P\}$ corresponds to $\{M\}$ and vice versa" (it is worth noting that different set of poses may correspond to the same set of motions).

When serial manipulators with lower mobility (LM-SMs) are considered, the end-effector motion is obtained by composing the motions of all the manipulator's joints (Hervé 1978). Therefore, a pure-motion LM-SM can be obtained only by using joints whose motions belong to the same motion subgroup. Moreover, the sum of the degrees of freedom (dofs) of the joints must be equal to the dimension of that motion subgroup.

When a parallel manipulator with lower mobility (LM-PM) is considered, the identification of the set of motions, $\{L\}$, the end effector can perform is a bit more complex. From a structural point of view, a parallel manipulator is a machine where the end effector is connected to the frame through a number, n, of kinematic chains (limbs) acting in parallel. Therefore, in an LM-PM with n limbs, both $\{L\}$ and $\{W\}$ are subsets of the common intersection of the n sets, respectively, of motions, $\{L_j\}$, j=1,…,n, and of poses, $\{W_j\}$, j=1,…,n, the j-th limb would allow to the end effector if it were the only kinematic chain joining end effector and frame. If all the $\{W_j\}$ are restricted to the end effector poses that can be reached without disassembling the LM-PM and all the corresponding $\{L_j\}$ are accordingly restricted (hereafter, if it is not differently specified, this restriction will be implicitly assumed), then the following relationships hold:

$${W} = \bigcap_{j=1,n} {W_j} \tag{1}$$

$${L} = \bigcap_{j=1,n} {L_j} \tag{2}$$

This fact discloses a wide scenario of possible machine architectures even for pure-motion LM-PMs since conditions (1) and (2), where {L} is a subset of a motion subgroup, can be satisfied by using, in the limbs, joints of any type (not necessarily belonging to the same motion subgroup), and, as limbs, kinematic chains with number of dof (limb's connectivity) greater than the manipulator's dofs.

Each subset of LM-PMs can be further separated into two classes: the class of the overconstrained manipulators and the class of the non-overconstrained manipulators. Overconstrained manipulators are machines whose dof number is higher than the one computed as if the constraints due to the joints were independent. An overconstrained LM-PM can be obtained by using, as limbs, a number, n, of serial kinematic chains with the same number of dofs as the LM-PM, provided that n be equal to the LM-PM's dofs, and the limbs be arranged so that they warranty a non-empty intersection, {W}, among the n sets, {W_j}, j=1,...,n. This principle has guided the design of many overconstrained LM-PMs among which the most known is certainly the 3RRR wrist (Gosselin & Angeles 1989) which uses, as limbs, three serial wrists of type RRR with the same spherical-motion center (see Fig. 1). The advantage of a pure-motion LM-PM obtained by using this principle, with respect to the serial manipulator that have the same topology as the one of the LM-PM's limbs, is that the LM-PM has all the actuators located on the frame, which allows fast manipulators to be manufactured. This advantage is paid with a reduced workspace and with an architecture that need an high manufacturing precision to avoid jamming of the joints and high internal loads in the links.
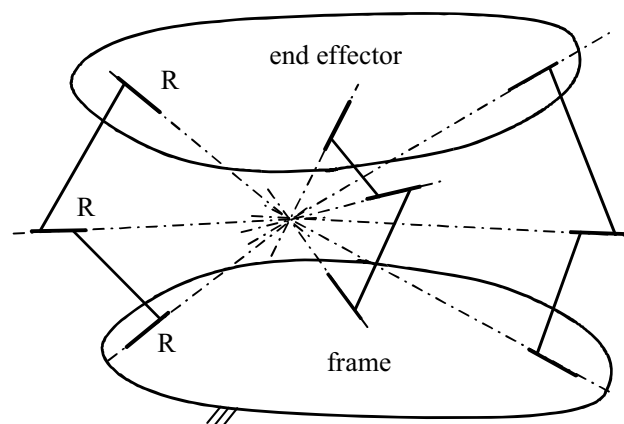
Figure 1. 3RRR wrist (Gosselin & Angeles 1989)

3. Determination of Limbs' Topologies for an LM-PM

Conditions (1) and (2) guide the determination of limbs' topologies suitable for making the end effector of an LM-PM perform a given type of motion. In the literature, the analyses that bring to identify such topologies have been mainly addressed through three different approaches: (i) group theory (Hervé 1978, 1995, 1999, 2004; Hervé & Sparacino 1991; Karouia & Hervé 2000, 2002; Huynh & Hervé 2005; Lee & Hervé 2006; Rico et al. 2006), (ii) screw theory (Tsai 1999; Fang & Tsai 2002; Frisoli et al. 2000; Kong & Gosselin 2002, 2004a, 2004b, 2005; Huang & Li 2002, 2003; Carricato 2005) and (iii) velocity-loop equations (Di Gregorio & Parenti-Castelli 1998; Di Gregorio 2001a, 2001b, 2002, 2003; Carricato & Parenti-Castelli 2002, 2003).

The first approach (group theory) determines the generic $\{L_j\}$ by composing the set of motions generated by each joint of the kinematic chain that is candidate to be a limb, and, then, searches for the geometric conditions the potential limb must satisfy in order to make a subset of an assigned motion subgroup of $\{D\}$ be a subset of $\{L_j\}$. The result of this type of analysis is the determination (topology plus geometric conditions) of all the generators of a given motion subgroup. Each generator of a given motion subgroup of $\{D\}$ can be used as limb in an LM-PM that must bound the end-effector motions to a subset of that motion subgroup.

The second approach (screw theory) determine the screw (twist), $\$_j$, which represents the generic element of $\{L_j\}$, as a linear combination of the twists of all the joints of the kinematic chain that is candidate to be a limb. Then, the screw (wrench), ξ_j, reciprocal to $\$_j$, which represents the system of forces exerted on the end effector by the j-th limb, is computed. Finally, the wrench system obtained as linear combination of the ξ_j is considered, and the geometric conditions that make it coincide with the wrench system reciprocal to all the elements of the motion subgroup, which $\{L\}$ must be a subset of, are deduced.

The third approach (velocity-loop equations) consists in writing n times both the end-effector angular velocity, ω, and the velocity, $\dot{P}$, of an end-effector point by exploiting the kinematic properties of the n limbs of the generic LM-PM topology under analysis. So doing n expressions of the couple of vectors (ω, $\dot{P}$) are obtained where the j-th expression, j=1,…,n, is a linear combination of the joint rates of the j-th limb. The analysis of these (ω, $\dot{P}$) expressions is sufficient to determine the geometric conditions that each limb has to satisfy in order to make (I) all the n expressions compatible, and (II) the end-effector's motion characteristics (ω, $\dot{P}$) respect the conditions which warranty that all the end effector motions belong to an assigned motion subgroup of $\{D\}$. Since this approach deduces geometric conditions by analysing the instantaneous end-effector motion, the characteristics of the finite end-effector motion are stated by demonstrating that those conditions are sufficient to warranty an infinite

sequence of instantaneous motion of the same type provided that no singular configuration is encountered.

The first and the third approaches rely on purely kinematic considerations, whereas the second one takes into account both kinematic and static considerations which is typical of approaches based on screw theory. The three approaches are all able to find the singular configurations of any LM-PM architecture, even though the second and the third ones directly give the singularity conditions as a result of the analysis that identifies the limb topologies, which make them more appropriate for the singularity analysis.

4. Singularity Analysis

Singularities are manipulator configurations where the input-output instantaneous relationship fails (Gosselin & Angeles 1990; Ma & Angeles 1991; Zlatanov et al. 1995). If the input-output instantaneous relationship is considered (Gosselin & Angeles 1990), they are of three types: (I) singularities of the inverse kinematic problem, (II) singularities of the direct kinematic problem, and (III) singularities both of the inverse and of the direct kinematic problems.

Type-(I) singularities occur when at least one out of the input-variable rates (actuated-joint rates) are undetermined even though all the output-variable rates (end-effector's motion characteristics (ω, $\dot{P}$)) are assigned. All the manipuator configurations where the end effector reaches the border of the workspace are type-(I) singularities, and finding type-(I) singularities is one way to determine the workspace border. From a static point of view, in type-(I) singularities, at least one component of output torque (force), applied to the end effector, is equilibrated by the manipulator structure without applying any input torque (force) in the actuated joints.

Type-(II) singularities occur when at least one component of end-effector's motion characteristics, (ω, $\dot{P}$), is undetermined even though all the actuated-joint rates are assigned. These singularities may be present only in the PMs and fall inside the workspace. From a static point of view, in type-(II) singularities, a (finite or infinitesimal) output torque (force), applied to the end effector, needs at least one infinite input torque (force) in the actuated joints to be equilibrated. Since, long before the input torque (force) becomes infinite, the manipulator breaks down, type-(II) singularities must be found during design and avoided during operation.

This singularity classification has been extended in (Zlatanov et al. 1995) by taking into account the rates of the non-actuated joints.

In the literature (Di Gregorio & Parenti-Castelli 2002; Di Gregorio 2001a, 2001b, 2002a, 2003, 2004a, 2004c; Zatlanov et al. 2001, 2002), the possibility of changing the type of motion, the end effector performs, in correspondence of par-

ticular type-(II) singularities, named constraint singularities, has been high-lighted. Constraint singularities affect only LM-PMs where the limbs' connectivity is greater than the manipulator's dofs.

Conditions (1) and (2) can explain why constraint singularities may occur in LM-PMs where the limbs' connectivity is greater than the manipulator's dofs. If m_j and m with $m<m_j\leq6$ are the connectivity of the j-th limb and the LM-PM's dofs respectively, then the $\{W_j\}$ and the $\{L_j\}$ sets have dimension m_j whereas $\{W\}$ and $\{L\}$ have dimension m. A continuous subset with dimension m of a continuous set with dimension m_j (>m) can be generated in ∞^{m_j-m} ways; hence, it can happen that $\{L_j\}$ have m-dimensional subsets, $\{L_{kj}\}$, $k=1,\ldots,s_j$, of different motion subgroups of $\{D\}$ and of mixed motions among its m-dimensional subsets, and that the corresponding m-dimensional subsets, $\{W_{kj}\}$, $k=1,\ldots,s_j$, of $\{W_j\}$ have a non-empty intersection $\{C_j\}$ (i.e. they constitute a connected set). When this condition occurs, the j-th limb can move the end effector from a pose of $\{C_j\}$ by making it perform motions that belong to different m-dimensional motion subgroups of $\{D\}$ (that belong either to a m-dimensional motion subgroup of $\{D\}$ or to a mixed-motion subsets of $\{W_j\}$). Since, according to condition (1) the set $\{C\}$, defined as follows

$$\{C\} = \bigcap_{j=1,n} \{C_j\} , \qquad\qquad (3)$$

must be a subset of $\{W\}$, if $\{C\}$ is a non-empty set and $\{W\}$ contains subsets that belong to different m-dimensional motion subgroups (or to m-dimensional subsets of mixed motions together with subsets of a m-dimensional motion subgroup), then, there is a non-empty subset, $\{S\}$, of $\{C\}$ whose elements are end-effector poses from which the LM-PM can move the end effector by making it perform motions that belong to different m-dimensional motion subgroups of $\{D\}$ (that belong to either a m-dimensional motion subgroup of $\{D\}$ or a mixed-motion subsets of $\{W\}$). The end-effector's possibility of leaving a pose by performing motions that belong to disjoint m-dimensional subsets of $\{D\}$ implies that the end effector locally has at least one additional dof (i.e. m+h dofs with $h\geq1$) when the end effector assumes that pose. Therefore, when the end effector assumes that pose, the end effector's motion characteristics, ($\boldsymbol{\omega}$, $\dot{\mathbf{P}}$), are not determined even though the m actuated-joint rates are assigned (i.e. the LM-PM's configuration with the end effector located at that pose is a particular type-(II) singularity).

In (Zatlanov et al. 2001), it has been presented a three-dof LM-PM with topology 3URU (i.e. with three limbs of type URU (U stands for universal joint)), named DYMO (Fig. 2), that, by crossing constraint singularities, can become either a TPM, or an SPM, or a PPM, or a three-dof mixed-motion LM-PM.
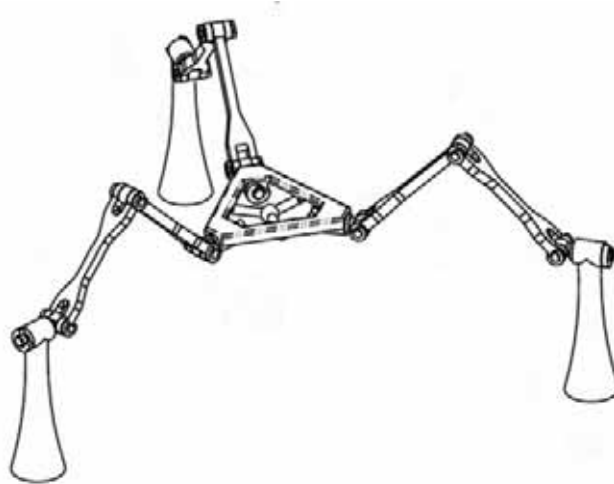
Figure 2. 3URU DYMO (Zatlanov et al. 2001)

A method to avoid the presence of constraint singularities is to choose the limb topologies and to assemble the limbs so that the subset {S} is an empty set. This idea guided the proposal of the 3RRS wrist (Di Gregorio 2004b), that has three limbs of type RRS, where the axes of the six (two per limb) revolute pairs pass through the center of the spherical motion (see Fig. 3). The three RRS limbs are assembled so that {L} contains only motions belonging to the spherical sub-group which make {S} empty even though {C} is not empty.

All type-(II) singularities must be individuated during the LM-PM design and, when possible, eliminated by suitably choosing the manipulator geometry. Moreover, the end effector must be kept as far as possible from these singularities during operation.

From an analytic point of view, the research of the type-(II) singularities can be implemented either through a static analysis (Di Gregorio 2004a) or through a kinematic analysis (Di Gregorio 2003). The static analysis studies the relationship between the system of external loads applied to the end effector and the set of the generalised torques applied in the actuated joints to equilibrate those loads. The kinematic analysis studies the relationship between the end-effector's motion characteristics (ω, $\dot{\mathbf{P}}$) and the m-dimensional vector (m (m<6) is the dof number of the LM-PM), $\dot{\mathbf{q}}$, that collects all the actuated-joint rates, $\dot{q}_p$, p=1,...,m.
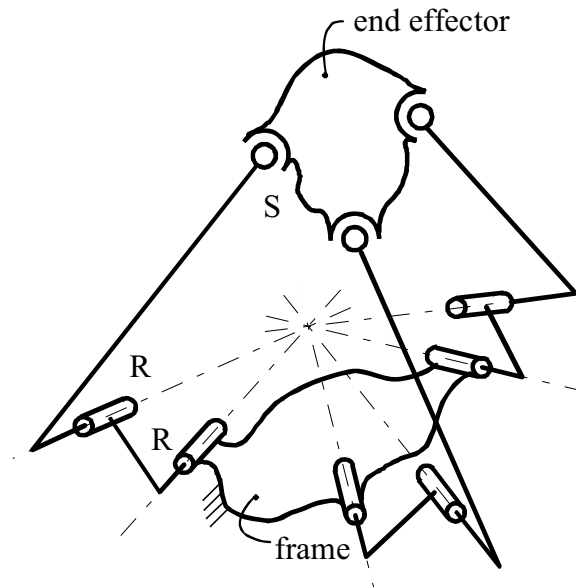
Figure 3. 3RRS wrist (Di Gregorio 2004b)

By following the method based on the kinematic analysis, the relationship to be studied is

$$\mathbf{A} \begin{Bmatrix} \dot{\mathbf{P}} \\ \boldsymbol{\omega} \end{Bmatrix} = \mathbf{B}\dot{\mathbf{q}} \qquad (4)$$

where **A** and **B** are a 6×6 matrix and a 6×m matrix respectively, and both, in general, depend on the m-dimensional vector **q** which collects the m actuated-joint variables, q_p, p=1,…,m, (i.e. they depend on the manipulator configuration). Since the LM-PM has m dofs, 6–m equations of system (4) simply state that $\boldsymbol{\omega}$ and $\dot{\mathbf{P}}$ cannot be arbitrarily chosen.

A non-singular configuration is characterised by rank(**A**)=6 and rank(**B**)=m. A type-(I) singularity is characterised by rank(**A**)=6 and rank(**B**)<m. A type-(II) singularity is characterised by rank(**A**)<6 (i.e. det(**A**)=0) and rank(**B**)=m. A type-(III) singularity is characterised by rank(**A**)<6 and rank(**B**)<m.

In order to find the type-(II) singularities the values of **q** that solve the equation

$$\det(\mathbf{A})=0 \qquad (5)$$

must be determined. Moreover, the condition number of **A** evaluated for an assigned value of **q** (i.e. an assigned configuration) can be used to judge how far is the configuration individuated by that value of **q** from type-(II) singular-

ity conditions (Gosselin & Angeles 1991): the nearer to one the condition number is, the farther from type-(II) singularity conditions that configuration is (the condition number ranges from 1 to infinity). The configurations where the condition number of $\mathbf{A}$ is equal to one are the farthest from type-(II) singularity conditions. Such configurations are called isotropic configurations. In an isotropic configuration, the matrix $\mathbf{A}^T\mathbf{A}$ is proportional to the 6×6 identity matrix, $\mathbf{I}_6$, or, which is the same, the singular values of $\mathbf{A}$ are all equal.

In an LM-PM has a matrix $\mathbf{A}$ that is constant (i.e. does not depend on $\mathbf{q}$) and non singular, then all the manipulator configurations have the same condition number and are not singular. Such a manipulator will be called constant-isotropy LM-PM. In addition, if, in a constant-isotropy LM-PM, the constant value of the condition number of $\mathbf{A}$ is one, then all the manipulator configurations are isotropic and the manipulator is called fully isotropic.

The appealing properties of constant-isotropy or fully isotropic LM-PMs pushed researchers to determine their topologies (Kong & Gosselin 2002a; Carricato & Parenti-Castelli 2002; Carricato 2005; Gogu 2004). Among all the proposed fully-isotropic architecture, the Cartesian 3PRRR (Kong & Gosselin 2002b; Di Gregorio 2002b; Kim & Tsai 2003) is certainly the most known. With reference to Fig. 4, the Cartesian 3PRRR has three limbs of type PRRR where the prismatic pair is actuated. In the j-th limb, j=1, 2, 3, of type PRRR, the three revolute-pair axes and the prismatic-pair sliding direction are all parallel. Finally, the sliding directions of the three prismatic pairs are mutually orthogonal.
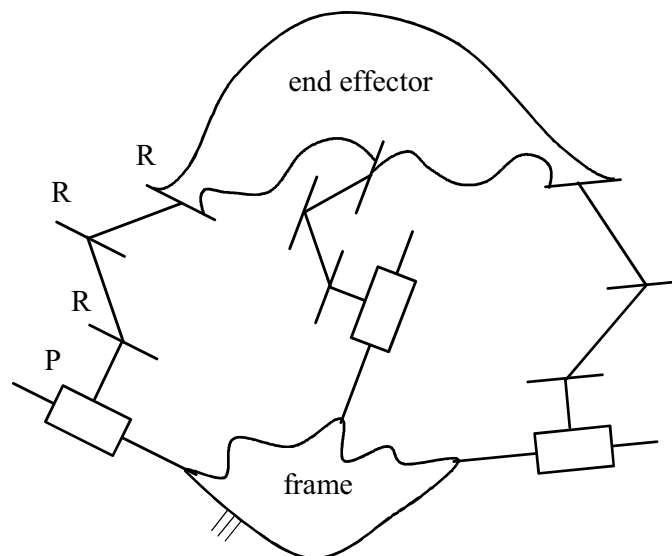
Figure 4. Cartesian 3PRRR (Kong & Gosselin 2002b; Di Gregorio 2002b; Kim & Tsai 2003)

5. Conclusions

The functional (kinetostatic) design of a LM-PM starts from the analysis of the manipulation task to accomplish, continues with the identification of the limb topologies and finishes with the determination of the machine architecture passing through the singularity analysis.

In the literature, the methodologies for the identification of the limb topologies suitable for a given manipulation task has been well described. How to combine the limbs in order to avoid singularities has been diffusely discussed at least for the most popular architectures. Nevertheless, comparison criteria among different architectures that perform the same manipulation task are not well established, yet. So that, even though is quite easy to find long lists of limb's topologies that are suitable for a given manipulation task (the works reported in the references are just a sample of the vast literature on this subject), stating which is the best one still is an open problem.

Some authors (Tsai & Joshi 2001) proposed the use of the "global condition number" (defined as the average value, on the workspace, of the inverse of the condition number) as index for evaluating or optimising a machine, but the same authors had to recognise that the comparison among different LM-PMs must take into account also the inertia properties of the machines. In general, it can be said that machines which exhibit good kinetostatic properties do not necessarily provide good dynamic performances.

6. References

Carricato, M. (2005). Fully isotropic four-degrees-of-freedom parallel mechanisms for Schoenflies motion. *The International Journal of Robotics Research*, Vol. 24, No. 5, (2005), pp. 397-414

Carricato, M. & Parenti-Castelli, V. (2002). Singularity-free fully-isotropic translational parallel mechanisms. *The International Journal of Robotics Research*, Vol. 21, No. 2, (2002), pp. 161-174

Carricato, M. & Parenti-Castelli, V. (2003). A family of 3-DOF translational parallel manipulators. *ASME Journal of Mechanical design*, Vol. 125, No. 2, (2003), pp. 302-307

Clavel, R. (1988). DELTA: A fast robot with parallel geometry, *Proceedings of the 18th International Symposium on Industrial Robots*, pp. 91-100, Sydney (Australia), April 1988

R. Di Gregorio, R. (2001a). Kinematics of a new spherical parallel manipulator with three equal legs: the 3-URC wrist. *Journal of Robotic Systems*, Vol. 18, No. 5, (2001), pp. 213-219

R. Di Gregorio, R. (2001b). A new parallel wrist employing only revolute pairs: the 3-RUU wrist. *ROBOTICA*, Vol. 19, No. 3, (2001), pp. 305-309

R. Di Gregorio, R. (2002a). A new family of spherical parallel manipulators. *ROBOTICA*, Vol. 20, No. 4, (2002), pp. 353-358

R. Di Gregorio, R., (2002b). Kinematics of a new translational parallel manipulator, *Proceedings of the 11th International Workshop on Robotics in Alpe-Adria-Danube Region*, RAAD 2002, pp. 249-254, Balatonfured (Hungary), June 30- July 2, 2002

R. Di Gregorio, R. (2003). Kinematics of the 3-UPU wrist. *Mechanism and Machine Theory*, Vol. 38, No. 3, (2003), pp. 253-263

R. Di Gregorio, R. (2004a). Statics and singularity loci of the 3-UPU wrist. *IEEE Transactions on Robotics*, Vol. 20, No. 4, (2004), pp. 630-635; (also in: *Proceedings of 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, AIM'01, Como, Italy, July 8-11, 2001)

R. Di Gregorio, R. (2004b). The 3-RRS wrist: a new, simple and non-overconstrained spherical parallel manipulator. *ASME Journal of Mechanical Design*, Vol. 126, No. 5, (2004), pp. 850-855 (also in: *Proceedings of ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DETC/CIE 2002, Montreal, Canada, September 29-October 2, 2002, Paper No. DETC2002/MECH-34344)

R. Di Gregorio, R., (2004c). Determination of singularities in DELTA-like manipulators. *The International Journal of Robotics Research*, Vol. 23, No. 1, (2004), pp. 89-96

R. Di Gregorio, R. & Parenti-Castelli, V. (1998). A translational 3-DOF parallel manipulators, In: *Advances in Robot Kinematics*, Lenarcic, J. & Husty, M.L., (Ed.), pp. 49-58, Kluwer Academic Publishers, Dordrecht (Netherlands)

R. Di Gregorio, R. & Parenti-Castelli, V. (2002). Mobility Analysis of the 3-UPU Parallel Mechanism Assembled for a Pure Translational Motion. *ASME Journal of Mechanical Design*, Vol. 124, No.2, (2002), pp. 259-264 (also in: *Proceedings of the 1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, AIM'99, Atlanta, Georgia, September 19-23, 1999, pp 520-525)

Fang, Y. & Tsai, L.-W., (2002). Structure synthesis of a class of 4-dof and 5-dof parallel manipulators with identical limb structures. *The International Journal of Robotics Research*, Vol. 21, No. 9, (2002), pp. 799-810

Frisoli, A.; Checcacci, D.; Salsedo, F. & Bergamasco, M. (2000). Synthesis by screw algebra of translating in-parallel actuated mechanisms, In: *Advances in Robot Kinematics*, Lenarcic, J. & Stanisic, M.M., (Ed.), pp. 433-440, Kluwer Academic Publishers, Dordrecht (Netherlands)

Gogu, G., (2004). Fully-isotropic T3R1-type parallel manipulators, In: *Advances in Robot Kinematics*, Lenarcic, J. & Galletti, C., (Ed.), pp. 265-274, Kluwer Academic Publishers, Dordrecht (Netherlands)

Gosselin, C.M. & Angeles, J., (1989). The optimum kinematic design of a spherical three-degree-of-freedom parallel manipulator. *ASME Journal*

of Mechanisms, Transmission and Automation in Design, Vol. 111, No. 2, (1989), pp. 202-207

Gosselin, C.M. & Angeles, J., (1990). Singularity analysis of closed-loop kinematic chains. *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 3, (1990), pp. 281-290.

Gosselin, C.M. & Angeles, J., (1991). A global performance index for the kinematic optimization of robotic manipulators. *ASME Journal of Mechanical Design*, Vol. 113, No. 2, (1991), pp. 220-226.

Hervé, J.M. (1978). Analyse structurale des mécanismes par groupe des déplacements. *Mechanism and Machine Theory,* Vol. 13, No. 4, (1978), pp. 437-450, ISSN 0094-114X/78/0801-0437

Hervé, J.M. (1995). Design of parallel manipulators via the displacement group, *Proceedings of the 9th World Congress on the Theory of Machines and Mechanisms*, Vol. 3, pp. 2079-2082, Milan (Italy), 1995

Hervé, J.M. (1999). The Lie group of rigid body displacements, a fundamental tool for mechanism design. *Mechanism and Machine Theory,* Vol. 34, No. 6, (1999), pp. 719-730, ISSN 0094-114X/98/00051-2

Hervé, J.M. & Sparacino, F. (1991). Structural synthesis of "parallel" robots generating spatial translation, *Proceedings of the IEEE 1991 International Conference on Automation and Robotics,* pp. 808-813, ISSN 7803-0078/91/0600-0808, Pisa (Italy), June 1991

Hervé, J.M. (2004). Parallel mechanisms with pseudo-planar motion generators, In: *Advances in Robot Kinematics*, Lenarcic, J. & Galletti, C., (Ed.), pp. 431-440, Kluwer Academic Publishers, Dordrecht (Netherlands)

Huang, Z. & Li, Q.C. (2002). General methodology for type synthesis of symmetrical lower-mobility parallel manipulators and several novel manipulators. *The International Journal of Robotics Research*, Vol. 21, No. 2, (2002), pp. 131-145

Huang, Z. & Li, Q.C. (2003). Type synthesis of symmetrical lower-mobility parallel mechanisms using the constraint-synthesis method. *The International Journal of Robotics Research*, Vol. 22, No. 1, (2003), pp. 59-79

Huynh, P. & Hervè, J.M. (2005). Equivalent kinematic chains of three degree-of-freedom tripod mechanisms with planar-spherical bonds. *ASME Journal of Mechanical Design*, Vol. 127, No. 1, (2005), pp. 95-102

Karouia, M. & Hervé, J.M. (2000). A three-dof tripod for generating spherical rotation, In: *Advances in Robot Kinematics*, Lenarcic, J. & Stanisic, M.M., (Ed.), pp. 395-402, Kluwer Academic Publishers, Dordrecht (Netherlands)

Karouia, M. & Hervé, J.M. (2002). A family of novel orientational 3-dof parallel robots, *Proceedings of the 14th CISM-IFToMM Symposium on Robots and Manipulators (RoManSy'14)*, pp. 359-368, Udine (Italy), July 2002, Springer (2003)

Kim, H.S. & Tsai, L.-W., (2003). Design optimization of a Cartesian parallel manipulator. *ASME Journal of Mechanical Design*, Vol. 125, No. 1, (2003), pp. 43-51

Kong, X. & Gosselin, C.M. (2002a). Type synthesis of linear translational parallel manipulators, In: *Advances in Robot Kinematics*, Lenarcic, J. & Thomas, F., (Ed.), pp. 411-420, Kluwer Academic Publishers, Dordrecht (Netherlands)

Kong, X. & Gosselin, C.M. (2002b). Kinematics and singularity analysis of a novel type of 3-CRR 3-dof translational parallel manipulator. *The International Journal of Robotics Research*, Vol. 21, No. 9, (2002), pp. 791-798

Kong, X. & Gosselin, C.M. (2004a). Type synthesis of 3-DOF translational parallel manipulators based on screw theory. *ASME Journal of Mechanical Design*, Vol. 126, No. 1, (2004), pp. 83-92

Kong, X. & Gosselin, C.M. (2004b). Type synthesis of 3-DOF spherical parallel manipulators based on screw theory. *ASME Journal of Mechanical Design*, Vol. 126, No. 1, (2004), pp. 101-108

Kong, X. & Gosselin, C.M. (2005). Type synthesis of 3-DOF PPR-equivalent parallel manipulators based on screw theory and the concept of virtual chain. *ASME Journal of Mechanical Design*, Vol. 127, No. 6, (2005), pp. 1113-1121

Lee, C.-C. & Hervé, J.M. (2006). Pseudo-planar motion generators, In: *Advances in Robot Kinematics*, Lenarcic, J. & Roth, B., (Ed.), pp. 435-444, Kluwer Academic Publishers, Dordrecht (Netherlands)

Ma, O., & Angeles, J. (1991). Architecture singularities of platform manipulators, *Proceedings of the 1991 IEEE International Confonference on Robotics and Automation*, pp. 1542-1547, Sacramento (CA, USA), April 1991

Rico, J.M.; Cervantes-Sancez, J.J.; Tadeo-Chavez, A.; Perez-Soto, G.I. & Rocha-Chavaria, J. (2006). A comprehensive theory of type synthesis of fully parallel platforms, *Proceedings of the ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, paper No DETC2006-99070, Philadelphia (USA), September 2006

Tsai, L.-W., (1999). The enumeration of a class of three-dof parallel manipulators, *Proceedings of the 10th World Congress on the Theory of Machine and Mechanisms*, pp. 1121-1126, Oulu (Finland), Gune 1999

Tsai, L.-W. & Joshi, S. (2001). Comparison study of architectures of four 3 degree-of-freedom translational parallel manipulators, *Proceedings of the 2001 IEEE International Confonference on Robotics and Automation*, pp. 1283-1286, Seoul (Korea), May 2001

Zlatanov, D.; Fenton, R.G. & Benhabib, B. (1995). A unifying framework for classification and interpretation of mechanism singularities. *ASME Journal of Mechanical Design*, Vol. 117, No. 4, (1995), pp. 566-572.

Zatlanov, D.; Bonev, I. A. & Gosselin, C.M. (2001). Constraint singularities as configuration space singularities. online paper: www.parallemic.org/Reviews/Review008.html

Zatlanov, D.; Bonev, I. A. & Gosselin, C.M. (2002). Constraint singularities of parallel mechanisms, *Proceedings of the 2002 IEEE International Confernce on Robotics & Automation*, pp.496-502, Washington (DC, USA), May 2002

Error Modeling and Accuracy
of Parallel Industrial Robots

Hongliang Cui and Zhenqi Zhu

1. Introduction

Most industrial robots are open-chain mechanisms constructed of consecutive links connected by rotational or prismatic joints of one degree of freedom. These serial manipulators have large workspace, high dexterity and good maneuverability. However, due to their serial structure they exhibit low stiffness and poor positioning accuracy. As a result, their use in applications that require large loads (e.g. machining) and high accuracy, is limited. In the case of a parallel manipulator, the end-effector is attached to a moveable plate which is supported in-parallel by a number of actuated links. As a result, these parallel manipulators are anticipated to possess the following advantages, compared with serial manipulators: 1) high force/torque capacity since the load is distributed to several in-parallel actuators; 2) high structural rigidity; and 3) better accuracy due to less cumulative joint errors.

A large number of publications dealing with the accuracy of the serial manipulators appeared in the past. These include topics on error modeling effects of manufacturing tolerance on pose accuracy and numerous calibration strategies. However, very few publications dealing with the same issue as related to parallel manipulators can be found. Since high accuracy is generally believed to be one of their advantages compared to that of serial manipulators, it is important to address this issue. The purpose of this research is to establish the kinematic and error models for evaluating the effects of manufacturing tolerances, installation errors and stiffness effect on the accuracy of a parallel robotic system.

In order to evaluate the accuracy of parallel robotic system, it is necessary to develop a kinematic model which will accommodate the above errors. Based on this model, algorithms for forward, inverse kinematics and error modeling of the parallel robot are presented.   These algorithms with a set of typical tolerances were used to compute the pose errors which include three translational and three angular errors.

Manufacturing tolerances, **installation errors** and **link offsets** cause deviations with respect to the nominal kinematic parameters of the robot system. As a result, if the nominal values of these parameters are used within the robot system control software, the resulting pose of the system will be inaccurate. Accuracy of a robot is the closeness with which its actual pose matches the pose predicted by its controller. A robot normally designed for repeated work such as spray painting, pick and place, etc., has high repeatability but low accuracy. An accurate robot is required in applications where off-line programming is involved. To a large extent, robot inaccuracy is induced by the propagation of *geometric errors*, *compliance errors* and *time-variant thermal errors.* The geometric errors of a robot come from manufacturing imperfections, misalignments or joint wear. Compliance errors are due to the flexibility of robot joints and link deflection under self-gravity and external load. The compliance errors also depend on the robot's changing position. Thermal errors result from thermal distortion and expansions of robot components due to internal and external heat sources such as motors, bearings and ambient temperature change.

Link and joint flexibility has a significant impact on robot performance and stability. Link gravity and external payload cause the deflection of links and flexible joints, and therefore degrade the robot performance. Link compliance effects are represented by six differential component changes: three translational and three rotational changes. This paper presents a systematic methodology for estimating the compliance characteristics of serial and parallel manipulators due to external concentrated load/deflection. In related experiments, special measurement tools and sensors are necessary to identify the stiffness of driving joints.

Also in this paper a general methodology is presented to calibrate and compensate for robot compliance errors and thermal errors in addition to geometric errors, An error synthesis model based on the Denavit-Hartenberg (D-H) convention is derived for simultaneously considering geometric errors, compliance errors and thermal errors. Based on this model a general methodology is developed to calibrate geometric errors, compliance errors and thermal errors. Experimental results show that the accuracy of the robot is improved by an order of magnitude after the compensation.

1.1 Serial and Parallel Robots

Robots are representative of mechanics devices which integrate aspects of manipulation, sensing, control, and communication. Rarely have so many technologies and scientific disciplines focused on the functionality and performance of a system as they have done in the fields of robot development and application. Robotics integrates the state of the art of many front-running technologies. Large efforts have been made to define an industrial robot and to

classify its application by industrial branches so that remarkably precise data and monitoring are available today.

The task of an industrial robot in general is to move a body (workpiece) with six maximal Cartesian spatial DOF (three translations, three rotations) to another point and orientation within a workspace. The complexity of the task determines the required kinematic configuration. The number of DOFs determines how many independently driven and controlled axes are needed to move a body in a defined way. Industrial robots normally have up to four principal arm axes and three wrist axes. While many exciting robot structures use serial kinematic chains, some parallel kinematic structures have been adopted for a variety of tasks. Typical configurations of industrial robots are shown in Figure 7. Most closed-loop kinematics is based on the so-called hexapod principle (Stewart platform, 1965), which represents a mechanically simple and efficient design. The structure is relatively stiff and enables relatively high positioning accuracy and high speeds, but workspace or working volume is limited.

Parallel or closed-chain linkages and serial or open-loop kinematic chains have been substantially investigated over last several decades. A closed-chain linkage, which usually has a limited number of degrees of freedom, is not applicable as a general-purpose robot kinematic configuration. A serial kinematic chain can provide a large workspace, but with less rigidity and load-carrying capacity compared with a parallel kinematic chain. The fully parallel-driven manipulators such as Stewart-platform have been investigated by many researchers. In general, the workspace of a robot arm consisting of only parallel kinematic chains is relatively small. Currently, there has been an increasing interest in the design of hybrid or serial-parallel robot manipulators which can provide salient features of both serial and parallel kinematic chains. An appropriately designed hybrid robotic manipulator will have a large load-carrying capacity and workspace, and yet be comparatively small and lightweight.

The TAU parallel configuration (Figure 1) is rooted in a series of inventions and was masterminded by Torgny Brogardh, 2000; 2001; 2002. The configuration of the robot simulates the shape of "τ" like the name of the Delta robot named after the "∇" shape configuration of the parallel robot. As shown in Fig. 1.1, the basic TAU configuration consists of three driving axes, three arms, six linkages, 12 joints and a moving (tool) plate. There are six chains connecting the main column to the end-effector in the TAU configuration. The TAU robot is a typical 3/2/1 configuration, which configuration is shown in Figure 11 of Section 2. There are three parallel and identical links and another two parallel and identical links. Six chains will be used to derive all kinematic equations. Table 1 highlights the features of the TAU configuration.

On the subject of D-H modeling, Denavit J. & Hartenberg, H, 1955, Tasi, L. 1999, Raghavan, M. 1993, Abderrahim M. & Whittaker, A. R. 2000 have ap-

plied the method and studied the limitations of various modeling methods. On the subject of **forward kinematics**, focus has been on finding *closed form solutions* based on various robotic configurations, and *numerical solutions* for difficult configurations of robots. It can be found in the work done by Dhingra A. K. et al. 1998, 2000, Shi, X. & Fenton, R. G. 1994, Didrit, O. et al, 1998, Zhang, X. & Song, S. 1991, Nanua, P. et al, 1990, Sreenivasan, S. V. et al, 1994, Griffis, M. & Duffy, J. 1989, and Lin, W. et al, 1992. On the subject of error analysis, Wang, J. & Masory, O. 1993, Gong, C. et al, 2000, Patel, A. J. & Ehmann, F. E. 2000 used *forward kinematic solutions* to obtain errors. **Jacobian matrix** was also used in obtaining errors. On the subject of the variation of parallel configurations, based on the work done by Dhingra, A. K. et al, 1999, 2000, Geng, Z. & Haynes, L. S. 1994, the influence of the configurations on the methods of finding closed form solutions can be found.

In this paper, the D-H model (Figure 2) is used to define the TAU robot configuration, a complete set of parameters is included in the modeling process. Kinematic model and error model are established for including all types of errors using **Jacobian matrix method** for the TAU robot. Meanwhile, a very effective **Jacobian Approximation Method** is introduced to calculate the forward kinematic problem instead of the Newton-Raphson method. It denotes that a closed form solution can be obtained instead of a numerical solution. A full size Jacobian matrix is used in carrying out error analysis, error budget, and model parameter estimation and identification. Simulation results indicate that both Jacobian matrix and Jacobian Approximation Method are correct and have an accuracy of micrometers. ADAMS simulation results are used in verifying the established models.

A six-degrees-of-freedom precision measuring system is introduced in this study as an application of all methods mentioned above. The methods are also applied to explore new robotic applications such as grinding and machining. These new developments also revive the interest in robotic performance evaluation. Given the mechanical configurations of industrial robots with their popular six degrees of freedom, industrial robots have to be evaluated with metrology device or system of 3 or more degrees of freedom. Evaluation methods and equipment are needed to measure the spatial pose of robot efficiently with low cost.

Several methods are available for characterizing robot performance in accordance with ISO 9283 "Manipulating Industrial Robots Performance Criteria and Related Test Methods". Eight major performance measuring methods and techniques are introduced in the technical report ISO TR 13309, including the accurate, easy-to-use but costly laser tracking technique. The pros and cons of existing multi-degrees of freedom measuring systems, including laser tracker, straight edges, multi-probes at certain check points, image and scanning techniques etc, are well documented [Lau and Hocken, 1984; Van Brussel, 1990; Ji-

ang et al, 1988]. Pose measurement of robotic end-effecter has been the focus [Ziegert and Datseries, 1990; Zhu and Cui, 2001, 2003].

Precision booster (Figure 3) a 6-DOF piezoelectric ultraprecision positioning drive is developed to provide industrial robots with 6-DOF fine positioning capability. It is designed to mount at the end of the forearm of a robot before its end-effector. With the added fine positioning capability, the accuracy of industrial robots can be greatly enhanced. Working with more accurate feedback sensors or calibration processes, the booster enables industrial robot to reach micrometer accuracy – one or two orders of magnitude higher than those of conventional serial robots. The accuracy of the precision booster can be designed in the range of sub-micrometer or micrometer over a range of millimeters enough to cover the sub-millimeter positioning resolution offered by existing industrial robots. The booster features monolithic flexure construction and the flexure structure functions as a spatial motion mechanism. This monolithic motion mechanism is backlash free and stick-slip free. High strength and high stiffness piezoelectric actuators are used to power the booster to perform fine positioning.

	TAU robot
Work area	1.5 m * 3 m
Repeatability	15 μm
Path accuracy	30 μm
Acceleration	5 g
Maximum positioning speed	180m / min
Excitation frequency	> 40 Hz
Cost	< 250 KUSD

Table 1. Specifications of the TAU Robot Based on Certain Applications
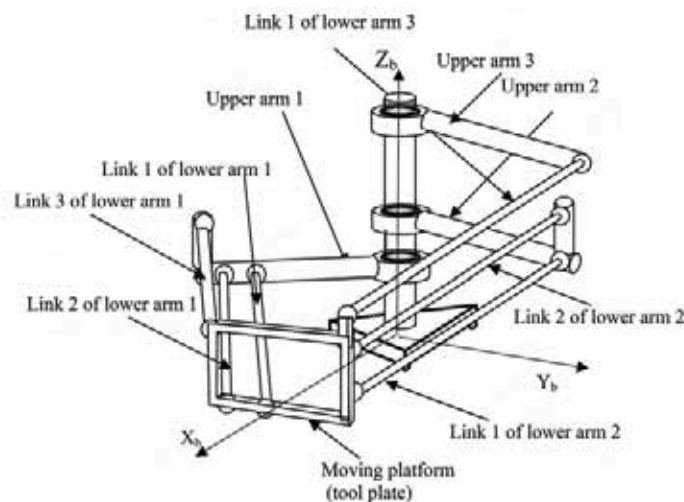
Figure 1. One of the TAU Robot Configurations
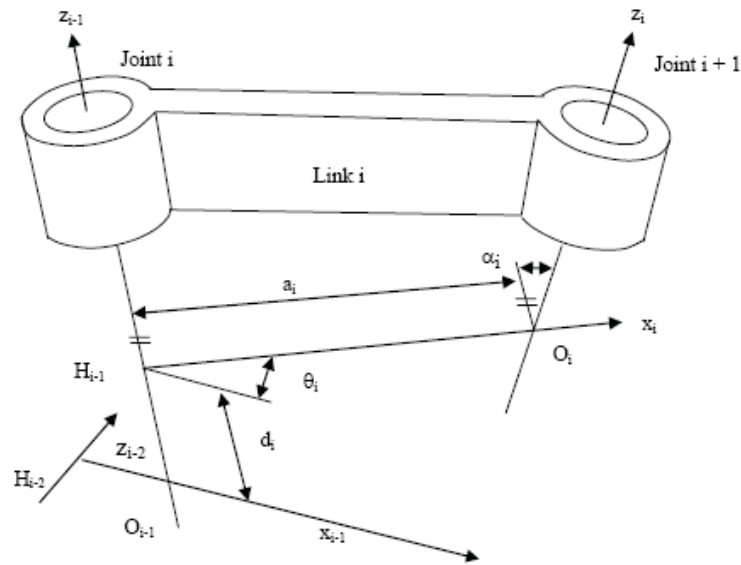
Figure 2. Definition of the Parameters in D-H Model

1.2 Kinematic Configurations of Parallel Robots

Gough-Stewart parallel robot, or so-called 'hexapod' shown in Figure 3 (Gough 1957 and Stewart 1965), is an assembly consisting of a fixed base with universal joints connecting the base to six linear-actuated limbs that support a moving platform through six ball-and-socket joints. This configuration allows the platform to move with six degrees-of-freedom employing the fewest number of actuators while maintaining stiffness by using only two-force-members. It is a closed-loop kinematic system with parallel links and is considered to be far more rigid than that of its serial counterparts of the same size and weight. Its force-output-to-manipulator-weight-ratio is generally an order of magnitude bigger than that of most industrial robots (Liu, 1993). *The same closed-loop kinematic configuration that gives its rigidity also complicates the solution of the forward kinematics in such a way that no closed-loop solution for this problem has been found* (Lacaze, Tasoluk and Meystel, 1997).

Tricept robot, shown in Figure 4, logically derived from the Tetrabot (Thornton, 1988), has a 3-DOF (degree of freedom) configuration of the parallel type to execute translational motions and a 3-DOF spherical wrist to execute rotational motions (Neumann and Neos Robotics, 1998). Its workspace is to be considered relatively large compared to the size of the robot. In order to further enlarge the size of the workspace, the addition of a revolute joint at the fixed base has been envisaged, introducing kinematic redundancy into the robotic manipulator. Its translational part can be thought as a reduced Stewart

platform with only three limbs. Like the Stewart platform, its kinematics has not been completely obtained: the inverse kinematics problem admits an analytical solution whereas the direct kinematics problem may require the use of iterative algorithms (Siciliano, 1999).

Delta robot, patented in U.S. in 1990 (Clavel, 1990), is shown in Figure 5. The basic idea behind the Delta parallel robotic design is the use of parallelograms. A parallelogram allows an output link to remain at a fixed orientation with respect to an input link. The use of three such parallelograms restrains completely the orientation of the mobile platform, which remains only three purely translational degrees of freedom. The input links of the three parallelograms are mounted on rotating levers via revolute joints. The revolute joints of the rotating levers are actuated in two different ways: with rotational (DC or AC servo) motors or with linear actuators. Finally, a fourth leg is used to transmit rotary motion from the base to an end-effector mounted on the mobile platform.

The use of base-mounted actuators and low-mass links allows the mobile platform to achieve accelerations of up to 50-G in experimental environment and 12 G in industrial applications. This makes the Delta robot a perfect candidate for pick and place operations of light objects. The Delta design has been applied to industry robot for several years. Its kinematics and dynamics also have been developed (Hunt 1973 and Codourey 1998).
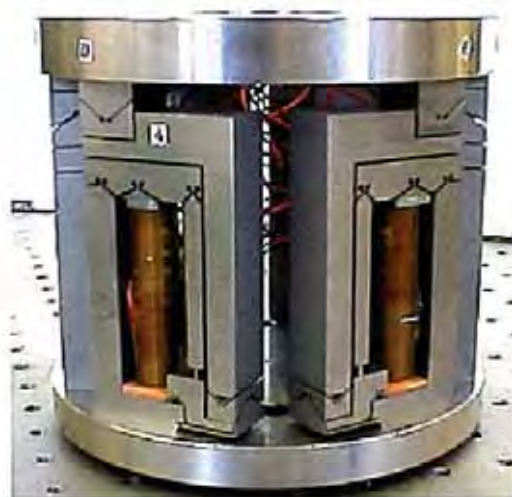
Figure 3.Piezo Driven Flexure Based Hexapod (Zhu and Cui, 2001)

Figure 4. Tricept Robot (Neumann and Neos Robotics, 1998)
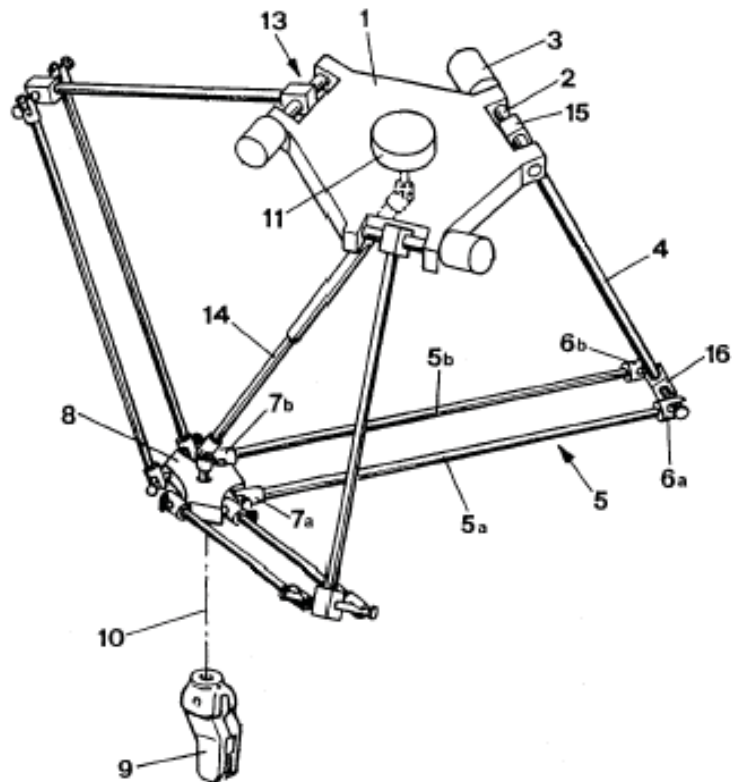
Figure 5. Delta Robot from US patent No. 4,976,582
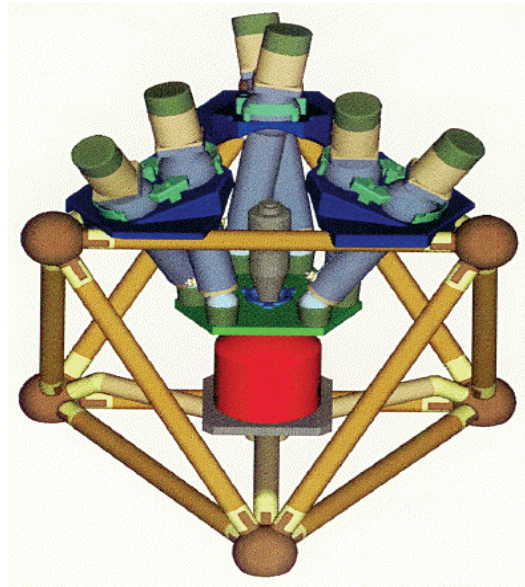
Figure 6. Octahedral Hexapod (NIST)

Octahedral Hexapod as shown in Fig. 1.6 is a demonstration machining center with six DOFs. It is a small, portable machine based on an octahedral framework. Machine motion is achieved by a Stewart Platform style actuation system. The framework and machining system can achieve high overall stiffness due to the fact that the structural members are generally in tension or compression with a minimum amount of bending stress. This structure allows the machine's capabilities to be independent of its foundation. Six identical struts with spherical pivots are mounted to the framework to drive the machining spindle, providing six-axis machining capability. The machine has a work volume of approximately 5" diameter X 3.5" high. The assembled machine will fit in a 24" X 24" X 25" volume. The machine completely disassembles and stores in a case approximately 24" X 16" X 10".
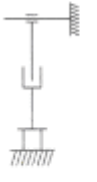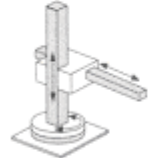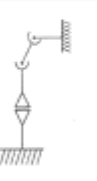
Figure 7 Typical Arms and Wrist Configuration of Industrial Robots (Handbook of Industrial Robotics)

2. Tau Configuration Design

Hybrid manipulators are parallel-serial connection robots that give rise to a multitude of highly articulate robotic manipulators. The robotic manipulators have a strength-to-weight ratio many times larger than the value currently available with industrial or research manipulators. This is due to the fact that these hybrid manipulators are stress compensated and ultralight in weight, yet are extremely stiff due to the fact that the force distribution in their structures is mostly axial.

Serially connected robot manipulators in the form of an open-loop kinematic chain with computer-controlled joint actuation have been utilized extensively in robot industry. For parallel manipulators, a classic example is the Stewart platform, which has been kinematically and, to some extent, dynamically investigated by many researchers.

The major advantages of existing parallel robots, compared with serial robots, are smaller mass and higher stiffness of the arm system. This is very important to achieve a shorter cycle time with lower actuator power together with more accurate movement.

The disadvantage is a relative small workspace in relation to the volume of the arm system. In the process of improving the robotic performance by diminishing the disadvantages, the basic features in design should include the following:

1. All the actuators are mounted on a fixed platform, which minimizes the mass of the moving arm system.
2. The links connected to the actuated platform are two-force members transmitting only compression and tensile forces and do not carry bending and twisting loads, which makes it easy to achieve a moving arm system of high in stiffness and low in mass.
3. The joints can be implemented as ball and socket bearings, which makes it possible to obtain high precision in addition to high stiffness and low mass for the joint arrangement.
4. The actuated platform is positioned with 3 translational DOFs in a parallel fashion without angular displacement.

2.1 The Link Clustering Design Approach

Systematic clustering of the links connected to the actuated robot platform has been studied. Based on this design approach new parallel arm structures have been identified and some new robot concepts have been found (Brogardh, T, 2000).

Figure 8 shows schematically the basic components needed to achieve the

Delta parallel arm robot with the kinematic features listed above. The actuated platform is connected to 6 links of type A by means of ball and socket joints that each has 3 DOFs. Type A means that the links are designed to be stiff only for forces along their axial direction in the structure. This force loading characteristic in the links of type A is guaranteed since a ball and socket joint cannot transmit bending moment or twisting torque to the link it is connected to.

The actuators in Figure 8 are mounted on the fixed platform and the moving part of the actuators is connected to the links of type A via links of type B. The type B links are designed to be stiff against also bending moment and twisting torque. All the links of type B do not need to be connected to actuators, but 3 of them must, otherwise the actuated platform cannot be manipulated in 3 DOFs.
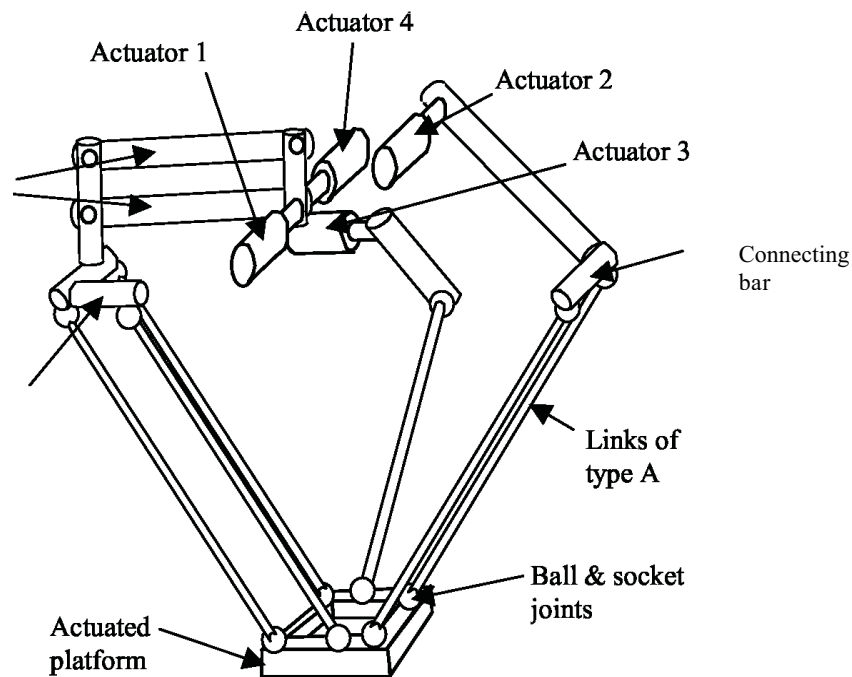
Figure 8. Components for the Design of Structures with the Same Features as the Delta Robot. (Courtesy of Brogardh, T, 2000).

Each of the links of type B (Figure 8) can be connected to one or more of the links of type A. One could say that each link of type B can be connected to a cluster of links of type A and it is possible to introduce a simple clustering scheme, where for example 2/2/2 means that the links of type A are clustered with 2 links to each of the 3 links of type B. To achieve parallel movements of the actuated platform (to preserve the tilt angles), type A links belonging to the same cluster must be parallel and have the same length. Moreover, to avoid a

collapsing parallel arm structure because of kinematic singularities, the placement of the type A link joints on the actuated platform must be optimized as well as the relative directions between the type A links of the different clusters. The 6 links of type A can be clustered in 3 ways: 2/2/2, 3/2/1 and 4/1/1. The 4/1/1 clustering will not fulfill the kinematic demands for a controllable structure and can be omitted. However, the 2/2/2 and the 3/2/1 clustering according to Figure 10 are kinematically useful. Using the 2/2/2 clustering scheme for the design will end up with the Delta structure. The optimized link placement in this case is achieved when the lines between the joints of each cluster on the actuated platform have an angle of 120 degrees between each other. The arm structure will collapse if the angle between two joint lines is 0 (180) degrees instead of 120 degrees. If instead the 3/2/1 clustering is used for the design of a parallel arm robot, the placement of the joints of the type A links on the platform surface is not critical. The only demand is that the 3 lower joints of cluster 1 are not allowed to be on a straight line on the platform. The optimum is achieved when these 3 joints of cluster 1 form a triangle with equal side length. This robustness with respect to the link placement on the actuated platform opens up new possibilities for the robot design.

In Figure 9 the actuated platform is considered to have a plane design, which means that the links of type A connect to the platform surface in a plane. However, the actuated platform could also be designed as a 3-D framework as depicted in Figure 10. This framework does not need to be a cube as in the figure, but the cube drawing makes it easier to see the configurations of the links. As in the case with a plane platform design, there are also in this case 2 useful clustering possibilities for an actuated 3-D platform: 2/2/2 and 3/2/1.

3/2/1 clustering   2/2/2 clustering
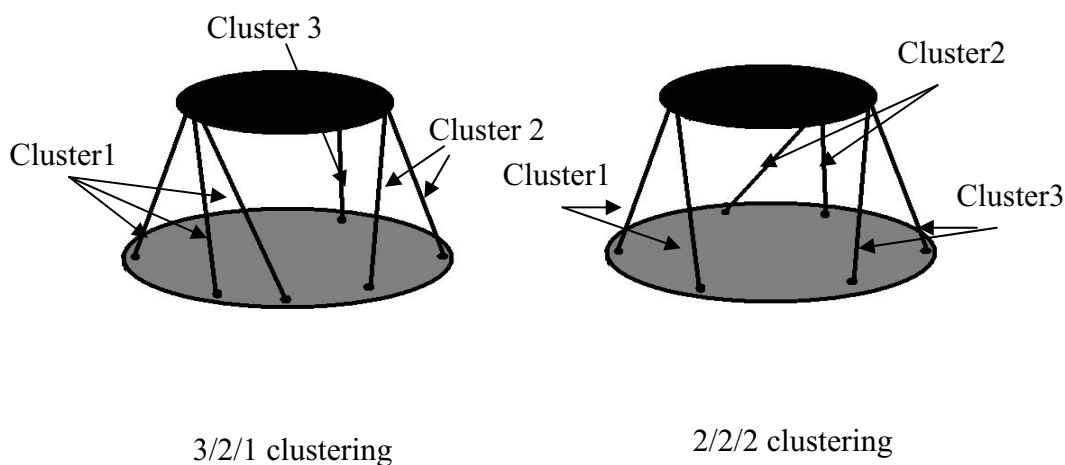
Figure 9. Useful Clustering Strategies When the Links of Type A Are Attached to the Actuator Platform in a 2-D Pattern.
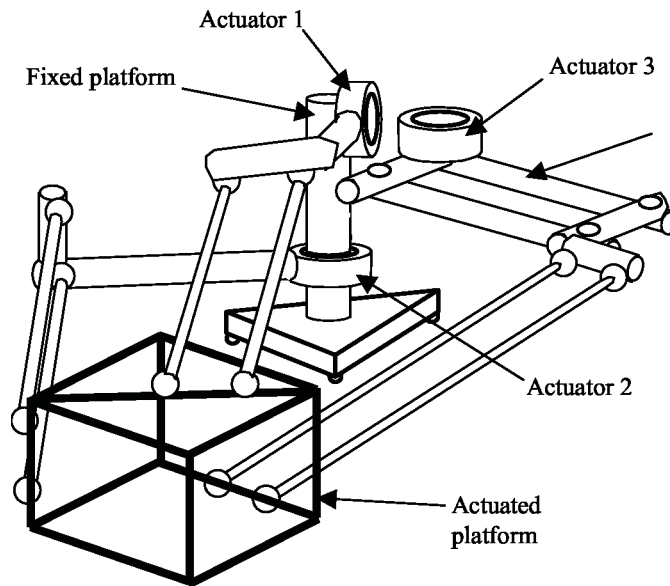
Figure 10. Useful 2/2/2 Clustering Strategies when the Links of Type A are Attached to the Actuator Platform in a 3-D pattern. (Courtesy of Brogardh, T, 2000).

2.2 TAU 3/2/1 Configuration

A new class of parallel robot, namely, TAU robot, has been created based on the 3/2/1 configuration. It combines the performance advantages of parallel arm mechanism (e.g., high stiffness, high accuracy) with the large workspace of serial robot.

As shown in Figure 11, the primary design of the TAU prototype robot has three actuators mounted on the base fixture and arranged in a line, which is called an I-configuration TAU. From bottom to top, actuators and upper arms (type B link) are numbered as 1, 2 and 3, and connected with 3, 2 and 1 lower arm(s) (type A link) respectively. This configuration basically performs a 3-DOF motion in its workspace. The 3-DOF parallel robot has a small footprint but with an enhanced stiffness.

The six links (lower arms) connected to the tool plate are driven by the three upper arms rotating around Z-axis. This structure has 3 DOFs in its work-space. With its geometric constraint, the DOF of a TAU robot is equal to (Tsai, 1999)

$$\text{DOF} = \lambda(n - j - 1) + \sum_{i=1} f_i$$

λ: degree of freedom of the space in which a mechanism is intended to function

n: number of links in a mechanism, including the fixed link

j: number of joints in a mechanism, assuming that all the joints are binary.

f_i: degree of relative motion permitted by joint i.

Joints between fixture and upper arms are 1-DOF rotational joints. Joints connecting upper and lower arms are 2-DOF universal joints. 3-DOF sphericajoints connect lower arms and moving plate.
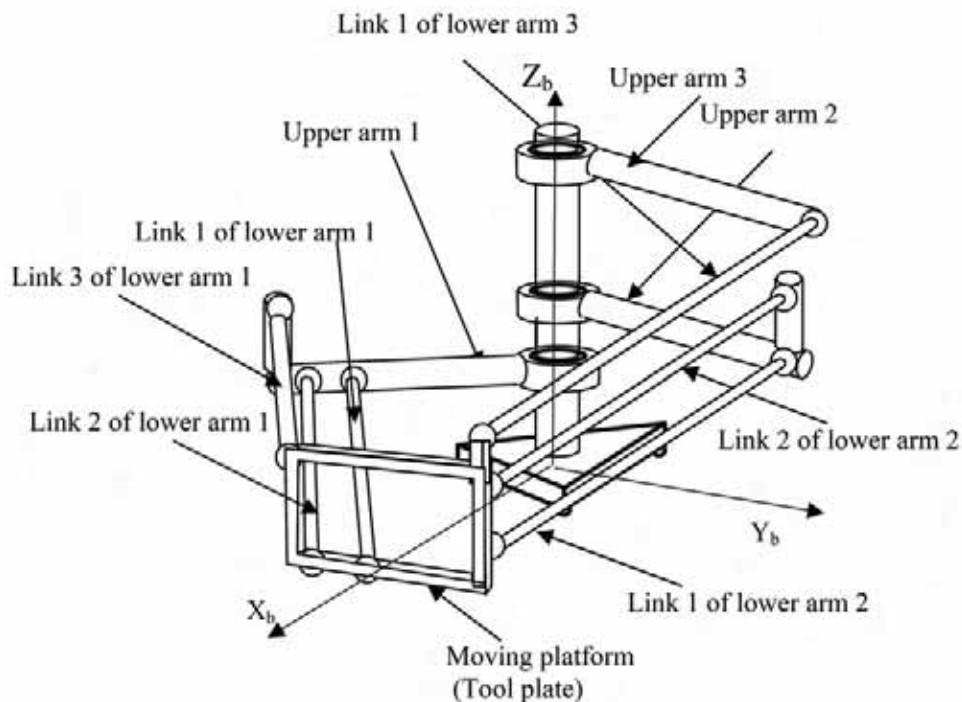
Figure 10.TAU Robot Based on Clustering Design Approach

2.3 Features of the TAU Configuration

The parallel robotic configuration for translational motion has a higher stiffness compared to the serial robotic configuration. It also has the following features: Large workspace, 360 degree around its base axis like a serial robot, analytic kinematic solution and analytic rigid-body dynamic solution.

Applications and Design Requirements

With these new features, the robot has the possibility to work with several conveyors and feeders placed around the robot. This is just one example of how a SCARA like parallel arm robot could be used to increase the productivity in an existing production line just by replacing conventional SCARA robots used today with its parallel arm cousins.

Typical Applications

Spot welding and painting are among the earliest application for industrial robot. Their payload is usually less than 50 kg. Repeatability requirement is in the range of 100 μm.

Pick and place and packaging have relatively low requirement on repeatability and stiffness. Payload varies from 1 to 500 kg. High speed is preferred for high productivity.

Machining or material removal including deburring, grinding, milling and sawing, requires high stiffness. Stiffness and accuracy of the robot decide the quality of the machined product.

Potential Applications

Laser cutting or welding, as a non-contact process requires an accuracy/repeatability less than 100 μm. Payload, which is the laser gun and accessories, is usually less than 50 kg. Speed required is not high in such applications.

Coordinate measuring function is typically performed by a CMM. It has a strict accuracy requirement of less than 50 μm for both static and path following at a low speed.

Fine material removal is as precision machining application now dominated by CNC machines. It requires the highest stiffness and system accuracy.

Design Objectives

The mechanism design is application orientated. Three typical future applications were selected and studied in the design phase: 2-D laser cutting, CMM for automobile vehicle and material removal applications. Each of them represents a typical application with certain requirements. Accuracy is a dominating factor reflecting the level of performance of any measurement system. The accuracy is low for current articulated robot arms. Material removal application requires high stiffness. Current serial configured CNC machines or parallel-configured machines have a limited workspace.

	TAU robot	Linear motor gantry
Work area	1.5 m * 3 m	1.5 m * 3 m
Repeatability	15 μm	17 μm
Path accuracy	30 μm	50 μm
Acceleration	5 g	2 - 4 g
Maximum positioning speed	180 m / min	100 - 200 m / min
Excitation frequency	> 40 Hz	13 Hz
Cost	< 250 KUSD	250 KUSD

Table 2. Performance Comparison with 2-D Laser Cutting Gantry Robot

Table 2 shows the performance comparison between the TAU robot and the gantry robot currently used in laser cutting application, which indicates the potential applications instead of using linear gantry robot. The performance of TAU covers all advantages of the Linear Motor Gantry.

Displacement sen-

Carbon fiber composite arm

Direct drive ac-
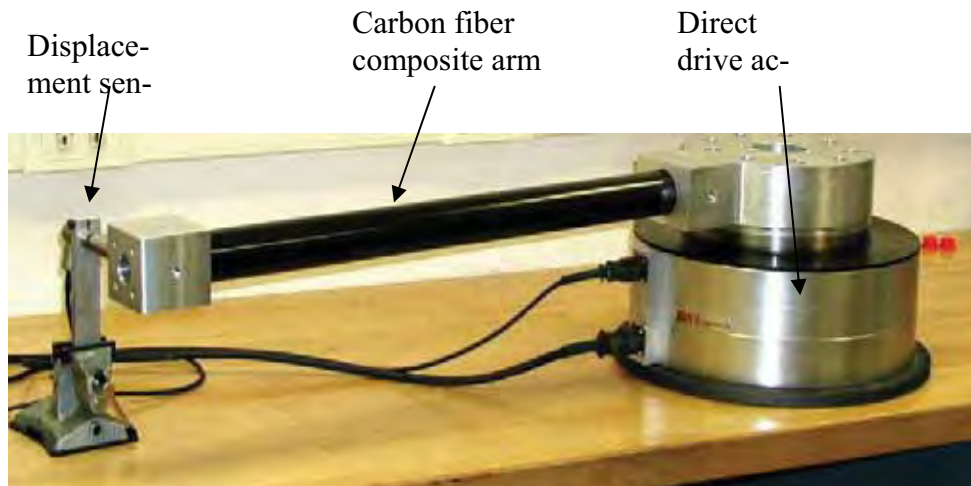
Figure 12. Single Arm Test Platform for Drive Motor Error Analysis
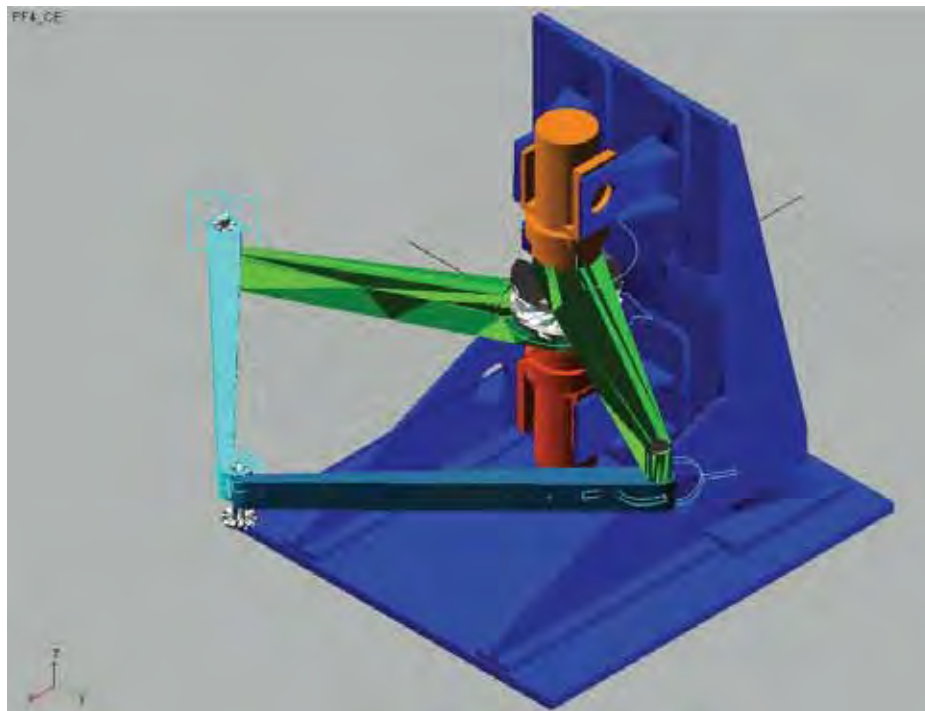
Figure 13. ADAMS Simulation Model for Two-Arm Test Platform

Figure 14. Two-Arm Test Platform (double SCARA structure)
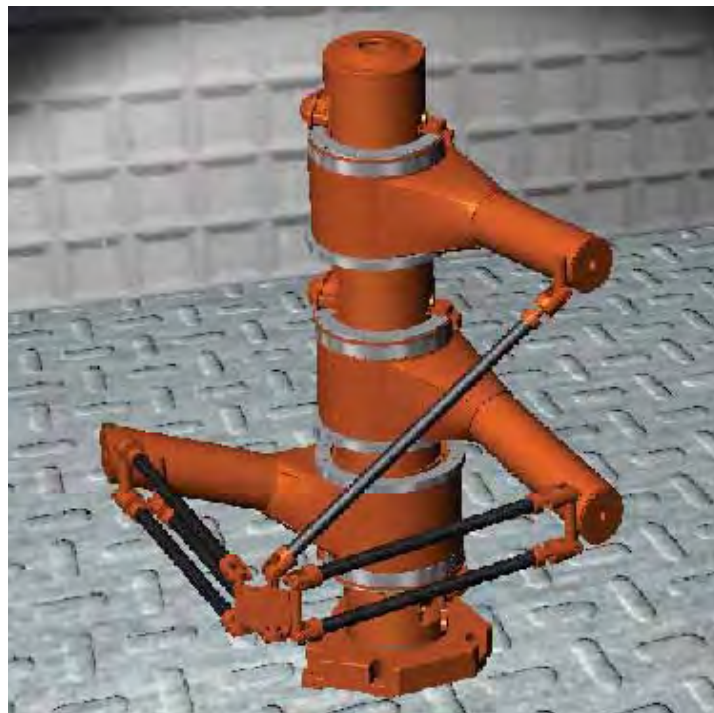
Figure 15. TAU Prototype Design

3. Kinematics of Tau configuration

This chapter gives the nominal (no error) kinematics of the TAU robot. It is a general solution for this type of 3-DOF parallel-serial robots. For the two-arm test platform, a simple kinematic solution can be obtained based on its double SCARA configuration and it is not included in this chapter. The two-arm test platform kinematics was used in friction model identification and kinematic error calibration of the two-arm test platform. It will be introduced as needed in the related chapters.

To solve the kinematics of this 3-DOF TAU robot, three independent equations are needed. The three lower arm links, connected between the moving plate and upper arm 1, are designed to be parallel to each other and have the same length. Similarly, the two lower arms of upper arm 2 are also parallel and equal in length, which gives another length equation. The third equation comes from the lower arm 3.

Formulating these three equations all starts from point P in Figure 3.1, where three kinematic chains meet. Three basic equations for the kinematic problem are:
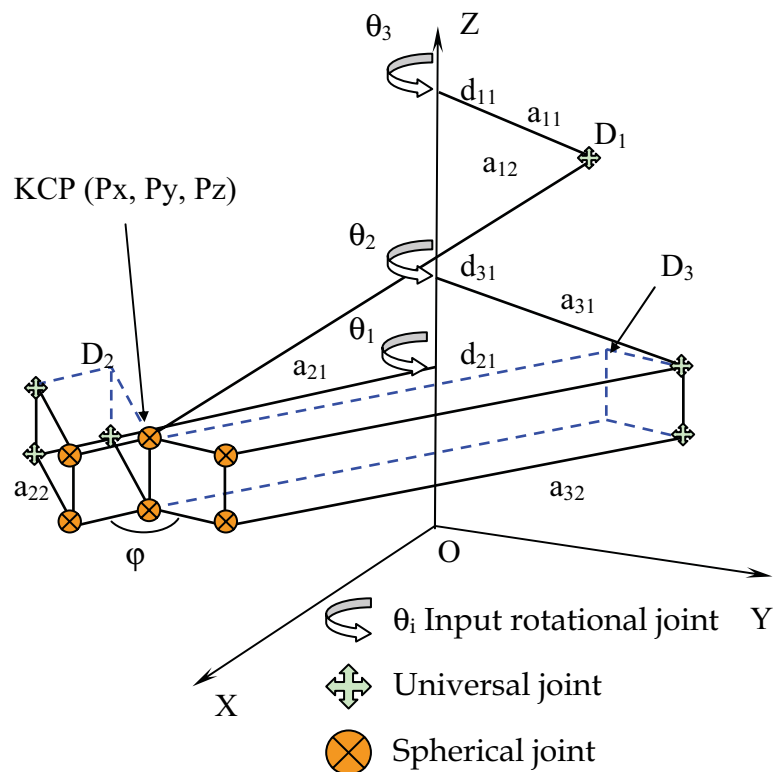
Figure 16.TAU Robot Kinematic Representation

For Point D_1:

$$D_{1x} = a_{11}\cos(\theta_3) - a_{33}\cos(120 + \theta_1)$$

$$D_{1y} = a_{11}\sin(\theta_3) - a_{33}\sin(120 + \theta_1)$$

$$D_{1z} = d_{11} - d4$$

$$dist(D_1 - P) = a_{12}$$

For Point D_2:

$$D_{2x} = a_{21}\cos(\theta_1)$$

$$D_{2y} = a_{21}\sin(\theta_1)$$

$$D_{2z} = d_{21} + d_{23}$$

$$dist(D_2 - P) = a_{22}$$

For Point D_3:

$$D_{3x} = a_{31}\cos(\theta_2) - a_{33}\cos(120 + \theta_1)$$

$$D_{3y} = a_{31}\sin(\theta_2) - a_{33}\sin(120 + \theta_1)$$

$$D_{3z} = d_{31}$$

$$dist(D_3 - P) = a_{32}$$

Basic equations

$$a_{12}^2 = (D_{1x} - Px)^2 + (D_{1y} - py)^2 + (D_{1z} - Pz)^2 \tag{1}$$

$$a_{22}^2 = (D_{2x} - Px)^2 + (D_{2y} - Py)^2 + (D_{2z} - Pz)^2 \tag{2}$$

$$a_{32}^2 = (D_{3x} - Px)^2 + (D_{3y} - Py)^2 + (D_{3z} - Pz)^2 \tag{3}$$

3.1 Inverse Kinematics

In an inverse kinematic problem, the Cartesian positioning information (Px, Py, Pz) is known. The unknowns are joint space position of active drive angles: θ_1, θ_2 and θ_3.
Substitute point D2 into Equation (2):

$$2a_{21}(Px\cos\theta_1 + Py\sin\theta_1) = a_{21}^2 - a_{22}^2 + Px^2 + Py^2 + (D_{2z} - Pz)^2$$

Therefore, the first angle is obtained as:

$$\theta_1 = \cos^{-1}\frac{a_{21}^2 - a_{22}^2 + Px^2 + Py^2 + (D_{2z} - Pz)^2}{2a_{21}\sqrt{Px^2 + Py^2}} + \tan^{-1}\frac{Py}{Px} \tag{4}$$

Substitute point D3 into Equation (3):

$$a_{32}^2 = (a_{31} \cos\theta_2 - a_{33} \cos(120+\theta_1) - Px)^2$$
$$+ (a_{31} \sin\theta_2 - a_{33} \sin(120+\theta_1) - Py)^2 + (D_{3z} - Pz)^2$$

Therefore:

$$\theta_2 = \cos^{-1}\frac{a_{31}^2 - a_{32}^2 + C_{3x}^{\,2} + C_{3y}^{\,2} + (D_{3z} - C_{3z})^2}{2a_{31}\sqrt{C_{3x}^{\,2} + C_{3y}^{\,2}}} + \tan^{-1}\frac{C_{3y}}{C_{3x}} \tag{5}$$

Where,

$$C_{3x} = Px + a_{33} \cos(120+\theta_1)$$
$$C_{3y} = Py + a_{33} \sin(120+\theta_1)$$
$$C_{3z} = Pz$$

Substitute point D1 into Equation (1)

$$2a_{11}(C_{3x} \cos\theta_3 + C_{3y} \sin\theta_3) = a_{11}^2 - a_{12}^2 + C_{3x}^{\,2} + C_{3y}^{\,2} + (D_{1z} - C_{3z})^2$$

Therefore:

$$\theta_3 = \cos^{-1}\frac{a_{11}^2 - a_{12}^2 + C_{3x}^{\,2} + C_{3y}^{\,2} + (D_{1z} - C_{3z})^2}{2a_{11}\sqrt{C_{3x}^{\,2} + C_{3y}^{\,2}}} + \tan^{-1}\frac{C_{3y}}{C_{3x}} \tag{6}$$

Equations (4), (5) and (6), therefore, are the inverse kinematics ended at point P on the moving platform. Noticed that point P is the kinematic calculation point, and additional inverse kinematic is needed to transfer TCP (Tool Center Point) to point P, when tool or wrist assembly is attached to the moving platform.

3.2 Forward Kinematics

The forward kinematic problem of a parallel configuration in general is more difficult than the inverse problem, for certain configurations there is no analytical solution admitted. For this TAU robot, the analytical forward kinematics is achievable. The Cartesian positioning information (Px, Py, Pz) is unknown in this case. The known are joint space position of active drive angles: θ_1, θ_2 and θ_3.

Change the format of Equations (1), (2) and (3) into:

$$D_{1x}^2 - 2D_{1x}Px + Px^2 + D_{1y}^2 - 2D_{1y}Py + Py^2 + D_{1z}^2 - 2D_{1z}Pz + Pz^2 = a_{12}^2 \qquad (7)$$

$$D_{2x}^2 - 2D_{2x}Px + Px^2 + D_{2y}^2 - 2D_{2y}Py + Py^2 + D_{2z}^2 - 2D_{2z}Pz + Pz^2 = a_{22}^2 \qquad (8)$$

$$D_{3x}^2 - 2D_{3x}Px + Px^2 + D_{3y}^2 - 2D_{3y}Py + Py^2 + D_{3z}^2 - 2D_{3z}Pz + Pz^2 = a_{32}^2 \qquad (9)$$

Equation (7) – Equation (8)

$$(D_{1x}^2 + D_{1y}^2 + D_{1z}^2) - (D_{2x}^2 + D_{2y}^2 + D_{2z}^2) - 2(D_{1x} - D_{2x})Px - 2(D_{1y} - D_{2y})Py$$
$$- 2(D_{1z} - D_{2z})Pz = a_{12}^2 - a_{22}^2 \qquad (10)$$

Equation (7) – Equation (9)

$$(D_{1x}^2 + D_{1y}^2 + D_{1z}^2) - (D_{3x}^2 + D_{3y}^2 + D_{3z}^2) - 2(D_{1x} - D_{3x})Px - 2(D_{1y} - D_{3y})Py$$
$$- 2(D_{1z} - D_{3z})Pz = a_{12}^2 - a_{32}^2 \qquad (11)$$

Thus, define:

$$d_1 = D_{1x}^2 + D_{1y}^2 + D_{1z}^2$$

$$d_2 = D_{2x}^2 + D_{2y}^2 + D_{2z}^2$$

$$d_3 = D_{3x}^2 + D_{3y}^2 + D_{3z}^2$$

Equation (10) becomes

$$(D_{1x} - D_{2x})Px + (D_{1y} - D_{2y})Py + (D_{1z} - D_{2z})Pz = (a_{22}^2 - a_{12}^2 + d_1 - d_2)/2$$

Let
$$a_1 = D_{1x} - D_{2x}$$
$$b_1 = D_{1y} - D_{2y}$$
$$c_1 = D_{1z} - D_{2z}$$
$$e_1 = (a_{22}^2 - a_{12}^2 + d_1 - d_2)/2$$

Substitutes into Equation (10):

$$a_1 \cdot Px + b_1 \cdot Py + c_1 \cdot Pz = e_1$$

Similarly define

$$a_2 = D_{1x} - D_{3x}$$
$$b_2 = D_{1y} - D_{3y}$$
$$c_2 = D_{1z} - D_{3z}$$
$$e_2 = (a_{32}^2 - a_{12}^2 + d_1 - d_3)/2$$

Then Equation (11) as

$$a_2 \cdot Px + b_2 \cdot Py + c_2 \cdot Pz = e_2$$
$$a_1 \cdot Px + b_1 \cdot Py = e_1 - c_1 \cdot Pz$$
$$a_2 \cdot Px + b_2 \cdot Py = e_2 - c_2 \cdot Pz$$

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} Px \\ Py \end{bmatrix} = \begin{bmatrix} e_1 - c_1 \cdot Pz \\ e_2 - c_2 \cdot Pz \end{bmatrix}$$

Define $\Delta = a_1 b_2 - a_2 b_1$

For case 1, $\Delta \neq 0$:

$$\Delta x = (e_1 - c_1 \cdot Pz)b_2 - (e_2 - c_2 \cdot Pz)b_1$$
$$= (b_2 e_1 - b_1 e_2) + (b_1 c_2 - b_2 c_1) \cdot Pz$$
$$\Delta y = (e_2 - c_2 \cdot Pz)a_1 - (e_1 - c_1 \cdot Pz)a_2$$
$$= (a_1 e_2 - a_2 e_1) + (a_2 c_1 - a_1 c_2) \cdot Pz$$
$$Px = \frac{\Delta x}{\Delta} = \frac{b_2 e_1 - b_1 e_2}{\Delta} + \frac{b_1 c_2 - b_2 c_1}{\Delta} \cdot Pz$$
$$Py = \frac{\Delta y}{\Delta} = \frac{a_1 e_2 - a_2 e_1}{\Delta} + \frac{a_2 c_1 - a_1 c_2}{\Delta} \cdot Pz$$

Define:

$$f_1 = \frac{b_2 e_1 - b_1 e_2}{\Delta}$$

$$f_2 = \frac{a_1 e_2 - a_2 e_1}{\Delta}$$

$$f_x = \frac{b_1 c_2 - b_2 c_1}{\Delta}$$

$$f_y = \frac{a_2 c_1 - a_1 c_2}{\Delta}$$

Thus:

$$Px = f_1 + f_x Pz$$
$$Py = f_2 + f_y Pz \tag{12}$$

Substitute Px, Py into Equation (3):

$$a_{32}^2 = (D_{3x} - f_1 - f_x Pz)^2 + (D_{3y} - f_2 - f_y Py)^2 + (D_{3z} - Pz)^2$$

Resort Equations above:

$$(1 + f_x^2 + f_y^2)Pz^2 + 2(f_{11}f_x + f_{22}f_y - D_{3z})Pz + (f_{11}^2 + f_{22}^2 + D_{3z}^2 - a_{32}^2) = 0$$

Where,

$$f_{11} = f_1 - D_{3x}$$
$$f_{22} = f_2 - D_{3y}$$

Then, let
$$A = 1 + f_x^2 + f_y^2$$
$$B = 2(f_{11}f_x + f_{22}f_y - D_{3z})$$
$$C = f_{11}^2 + f_{22}^2 + D_{3z}^2 - a_{32}^2$$

The solution of Equation $A \cdot Pz^2 + B \cdot Pz + C = 0$ is well known as:

$$Pz = \frac{-B \pm \sqrt{B^2 - 4 \cdot A \cdot C}}{2 \cdot A}$$
.... (13)

From Equation (12)

$$Px = f_1 + f_x Pz$$
.... (14)
$$Py = f_2 + f_y Pz$$
.... (15)

For case 1, Δ = 0, i.e. a2 = b2 = 0

$$\Delta = a_1 b_2 - a_2 b_1$$

In this case, $Pz = D_{3z}$ and only one Equation is available,

$$a_1 \cdot Px + b_1 \cdot Py = e_1 - c_1 \cdot Pz$$
i.e.

$$Py = (e_1 - c_1 \cdot Pz - a_1 \cdot Px) / b_1$$
$$= f_y Px + f_1 Pz + f_2$$

Where:

$$f_1 = -c_1 / b_1$$
$$f_2 = +e_1 / b_1$$
$$f_x = -a_1 / b_1$$

Substitute Px, Py into Equation (3) and resort the equation above:

$$(1 + f_x^2)Px^2 + 2(f_{11}f_x - D_{3x})Px + (f_{11}^2 + D_{3x}^2 - a_{32}^2) = 0$$

Where, $f_{11} = f_1Pz + f_2 - D_{3y}$
Then, let

$$A = 1 + f_x^2$$
$$B = 2(f_{11}f_x - D_{3x})$$
$$C = f_{11}^2 + D_{3x}^2 - a_{32}^2$$

The solution of Equation $A \cdot Px^2 + B \cdot Px + C = 0$ is well known as:

$$Px = \frac{-B \pm \sqrt{B^2 - 4 \cdot A \cdot C}}{2 \cdot A}$$

From Equation (12)

$$Py = f_x Px + f_1 Pz + f_2$$
$$Pz = D_{3z}$$

For case b1, b2 =0
Equation $a_1 \cdot Px + b_1 \cdot Py = e_1 - c_1 \cdot Pz$ becomes

$$Px = (e_1 - c_1 \cdot Pz) / a_1$$

Py can be solved by one of those basic Equations, for example for Equation (1).

$$Py = D_{1y} \pm \sqrt{a_{12}^2 - (D_{1x} - Px)^2 - (D_{1z} - Pz)^2}$$

The sign is the same as d1y.

For case a1, a2 =0

Equation becomes

$$a_1 \cdot Px + b_1 \cdot Py = e_1 - c_1 \cdot Pz$$
$$Py = (e_1 - c_1 \cdot Pz) / b_1$$

Px can be solved by one of those basic Equations as

$$Px = D_{1x} \pm \sqrt{a_{12}^2 - (D_{1y} - Py)^2 - (D_{1z} - Pz)^2}$$

The sign is the same as D_{1x}.

Thus forward kinematic is solved based on geometry constrains. Like the inverse kinematics, additional mathematic work is needed for the kinematic chain from point P to final TCP depending on the configuration details of the tool or wrist.

3.3 Discriminant Analysis of Kinematic Solution

Mathematically neither forward nor inverse kinematics gives single solution. Forward kinematics usually has two solutions, because the passive joint angles formed between upper arm and lower arm are not determined by kinematic equations. When only arm 1 and arm 2 chains are considered, upper arm 1, lower arm1, upper arm 2 and lower arm 2 form a quadrilateral geometry. These two solutions form one convex and one concave quadrilateral and one and only one of them is allowed by mechanical constrains. The discriminating condition is the angle between arm 1 and arm 2. For inverse kinematics, the mathematic equations can give out up to 8 solutions for the same position input. Still the physical constrains limits the left arm can be only placed on the left side of right arm, together with the convex and concave condition, there is only one solution is reasonable for arm 1 and arm 2. However including arm 3 into consideration, if it can rotate freely around its axis, there are two solutions for the drive angle of arm 3 except for singularity point. However since the elbow joint of arm 3 physically limits arm 3 so that arm 3 can only move within one side. Therefore, combining mathematics and physical constrains together, within the reachable workspace, TAU robot kinematics gives single solution on each input for both forward and inverse routine.

4. Error Modeling and Jacobian Matrix with all variables

The purpose of error analysis is to minimize the error of robot system through assembly based on the comprehensive system error model. The reason is based

on the fact that all error source will either have a negative or positive influence on the system error, which is then possible to arrange them in a way that cancellation or at least error reduction will happen. The methodology is described as:

- Identifying the error effect of individual component using the established system error model.
- Identifying the dimensional ranges allowed in an assembly for each connection.
- Using the system error model to identify the negative or positive direction that a connection should be made within the ranges allowed.
- Predicting and minimizing system error using the model.
- Using proper error budget approach to minimize the system error.

4.1 Error Modeling

The assembly process is a process of error identification and more importantly, a process of error assignment in the way towards minimizing system error. During the process, error budget is completed, and more importantly, an accurate kinematic model should be established. The process is geared directly towards error control and compensation when a robot is in service. The process is also a redesign process for improved performance
Next attentions should be paid to:

- The direction and degree of influence of an error source on system error varies in the whole workspace.
- Random errors can not be dealt effectively.
- Effective fixture and measuring are important.
- The methodology reduces robotic system error and opens the door for more accurate error compensation.

For the TAU-robot, an important thing needed is the error analysis. One needs to assign an error limit or range to all components in order to obtain a given robotic system accuracy. The procedure is so called Error Budget.
Before the error budget, an important thing to accomplish is to establish and analyze the Jacobian Matrix. It is necessary to know Jacobian Matrix for all components before assigning error to all components. On the other hand one can also obtain the final accuracy with knowing Jacobian Matrix. Besides one can know which components are more important than others based on the Jacobian Matrix. Table 3 lists all the design variables for the TAU robot.

NO.	DESCRIPTION	MAME	NO.	DESCRIPTION	MAME
			42		x5
1	drive 1	Joint 1	43	joint_link22_arm2	y5
2	drive 2	Joint 2	44		z5
3	drive 3	Joint 3	45		x6
17	joint 1 and arm 1	a1	46	joint_link13_arm3	y6
24		d1	47		z6
4		sit1	48	joint_link11_platform	x11
10		afa1	49		y11
18	joint_link11_arm 1	a2	50		z11
19	short arm 1	a3	51	joint_link31_platform	x22
25		d3	52		y22
5		sit3	53		z22
11		afa3	54	joint_link21_platform	x33
20	joint 2 and arm 2	a4	55		y33
26		d4	56		z33
6		sit4	57	joint_link12_platform	x44
12		afa4	58		y44
21	short arm 2	a5	59		z44
27		d5	60	joint_link22_platform	x55
7		sit5	61		y55
13		afa5	62		z55
22	joint 3	a6	63	joint-link13_platform	x66
28		d6	64		y66
8		sit6	65		z66
14		afa6	16	height of the TCP	a
23	arm 3	a7	66	link 13	L0
29		d7	67	link 11	L1
9		sit7	68	link 31	L2
15		afa7	69	link 21	L3
30	joint_link11_arm1	x1	70	link 22	L4
31		y1	71	link 12	L5
32		z1			
33	joint_link21_arm1	x2			
34		y2			
35		z2			
36	joint_link31_arm1	x3			
37		y3			
38		z3			
39	joint_link12_arm2	x4			
40		y4			
41		z4			

Table 3. Design Variables of TAU Robot

There are six kinematic chains from the base to the end-effector as:

Transfer Matrix: M1   Base->Joint1->Joint_link11_arm1
Transfer Matrix: M1*M3   Base->Joint1->Joint_link21_arm1
Transfer Matrix: M2   Base->Joint1->Joint_link31_arm1
Transfer Matrix: M4*M5   Base->Joint2->Joint_link12_arm2
Transfer Matrix: M4   Base->Joint2->Joint_link22_arm2
Transfer Matrix: M6*M7   Base->Joint3->Joint->Joint13_link_arm3

Where,

$$M_1 \rightarrow \begin{bmatrix} \cos(\text{joint}_1 + \Delta\theta_1) & -\sin(\text{joint}_1 + \Delta\theta_1)\cdot\cos(\Delta\alpha_1) & \sin(\text{joint}_1 + \Delta\theta_1)\cdot\sin(\Delta\alpha_1) & (700 + \Delta a_1)\cdot\cos(\text{joint}_1 + \Delta\theta_1) \\ \sin(\text{joint}_1 + \Delta\theta_1) & \cos(\text{joint}_1 + \Delta\theta_1)\cdot\cos(\Delta\alpha_1) & -\cos(\text{joint}_1 + \Delta\theta_1)\cdot\sin(\Delta\alpha_1) & (700 + \Delta a_1)\cdot\sin(\text{joint}_1 + \Delta\theta_1) \\ 0 & \sin(\Delta\alpha_1) & \cos(\Delta\alpha_1) & 750 + \Delta d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_2 \rightarrow \begin{bmatrix} \cos(\text{joint}_1 + \Delta\theta_2) & -\sin(\text{joint}_1 + \Delta\theta_2)\cdot\cos(\Delta\alpha_2) & \sin(\text{joint}_1 + \Delta\theta_2)\cdot\sin(\Delta\alpha_2) & (900 + \Delta a_2)\cdot\cos(\text{joint}_1 + \Delta\theta_2) \\ \sin(\text{joint}_1 + \Delta\theta_2) & \cos(\text{joint}_1 + \Delta\theta_2)\cdot\cos(\Delta\alpha_2) & -\cos(\text{joint}_1 + \Delta\theta_2)\cdot\sin(\Delta\alpha_2) & (900 + \Delta a_2)\cdot\sin(\text{joint}_1 + \Delta\theta_2) \\ 0 & \sin(\Delta\alpha_2) & \cos(\Delta\alpha_2) & 750 + \Delta d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_3 \rightarrow \begin{pmatrix} \cos(\Delta\theta_3) & -\sin(\Delta\theta_3)\cdot\cos(\Delta\alpha_3) & \sin(\Delta\theta_3)\cdot\sin(\Delta\alpha_3) & \Delta a_3\cdot\cos(\Delta\theta_3) \\ \sin(\Delta\theta_3) & \cos(\Delta\theta_3)\cdot\cos(\Delta\alpha_3) & -\cos(\Delta\theta_3)\cdot\sin(\Delta\alpha_3) & \Delta a_3\cdot\sin(\Delta\theta_3) \\ 0 & \sin(\Delta\alpha_3) & \cos(\Delta\alpha_3) & 200 + \Delta d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_4 \rightarrow \begin{bmatrix} \cos(\text{joint}_2 + \Delta\theta_4) & -\sin(\text{joint}_2 + \Delta\theta_4)\cdot\cos(\Delta\alpha_4) & \sin(\text{joint}_2 + \Delta\theta_4)\cdot\sin(\Delta\alpha_4) & (900 + \Delta a_4)\cdot\cos(\text{joint}_2 + \Delta\theta_4) \\ \sin(\text{joint}_2 + \Delta\theta_4) & \cos(\text{joint}_2 + \Delta\theta_4)\cdot\cos(\Delta\alpha_4) & -\cos(\text{joint}_2 + \Delta\theta_4)\cdot\sin(\Delta\alpha_4) & (900 + \Delta a_4)\cdot\sin(\text{joint}_2 + \Delta\theta_4) \\ 0 & \sin(\Delta\alpha_4) & \cos(\Delta\alpha_4) & 950 + \Delta d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_5 \rightarrow \begin{pmatrix} \cos(\Delta\theta_5) & -\sin(\Delta\theta_5)\cdot\cos(\Delta\alpha_5) & \sin(\Delta\theta_5)\cdot\sin(\Delta\alpha_5) & \Delta a_5\cdot\cos(\Delta\theta_5) \\ \sin(\Delta\theta_5) & \cos(\Delta\theta_5)\cdot\cos(\Delta\alpha_5) & -\cos(\Delta\theta_5)\cdot\sin(\Delta\alpha_5) & \Delta a_5\cdot\sin(\Delta\theta_5) \\ 0 & \sin(\Delta\alpha_5) & \cos(\Delta\alpha_5) & -200 + \Delta d_5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_6 \rightarrow \begin{pmatrix} \cos\left(\frac{1}{2}\cdot\text{joint}_1 + \frac{1}{2}\cdot\text{joint}_2 + \Delta\theta_6\right) & -\sin\left(\frac{1}{2}\cdot\text{joint}_1 + \frac{1}{2}\cdot\text{joint}_2 + \Delta\theta_6\right)\cdot\cos(-90 + \Delta\alpha_6) & \sin\left(\frac{1}{2}\cdot\text{joint}_1 + \frac{1}{2}\cdot\text{joint}_2 + \Delta\theta_6\right)\cdot\sin(-90 + \Delta\alpha_6) & \Delta a_6\cdot\cos\left(\frac{1}{2}\cdot\text{joint}_1 + \frac{1}{2}\cdot\text{joint}_2 + \Delta\theta_6\right) \\ \sin\left(\frac{1}{2}\cdot\text{joint}_1 + \frac{1}{2}\cdot\text{joint}_2 + \Delta\theta_6\right) & \cos\left(\frac{1}{2}\cdot\text{joint}_1 + \frac{1}{2}\cdot\text{joint}_2 + \Delta\theta_6\right)\cdot\cos(-90 + \Delta\alpha_6) & -\cos\left(\frac{1}{2}\cdot\text{joint}_1 + \frac{1}{2}\cdot\text{joint}_2 + \Delta\theta_6\right)\cdot\sin(-90 + \Delta\alpha_6) & \Delta a_6\cdot\sin\left(\frac{1}{2}\cdot\text{joint}_1 + \frac{1}{2}\cdot\text{joint}_2 + \Delta\theta_6\right) \\ 0 & \sin(-90 + \Delta\alpha_6) & \cos(-90 + \Delta\alpha_6) & 1700 + \Delta d_6 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_7 \rightarrow \begin{bmatrix} \cos(\text{joint}_3 + \Delta\theta_7) & -\sin(\text{joint}_3 + \Delta\theta_7)\cdot\cos(\Delta\alpha_7) & \sin(\text{joint}_3 + \Delta\theta_7)\cdot\sin(\Delta\alpha_7) & (900 + \Delta a_7)\cdot\cos(\text{joint}_3 + \Delta\theta_7) \\ \sin(\text{joint}_3 + \Delta\theta_7) & \cos(\text{joint}_3 + \Delta\theta_7)\cdot\cos(\Delta\alpha_7) & -\cos(\text{joint}_3 + \Delta\theta_7)\cdot\sin(\Delta\alpha_7) & (900 + \Delta a_7)\cdot\sin(\text{joint}_3 + \Delta\theta_7) \\ 0 & \sin(\Delta\alpha_7) & \cos(\Delta\alpha_7) & \Delta d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

So, the six length equations can be obtained from matrices above.

4.2 Jacobian Matrix of TAU Robot with All Error Parameters

In error analysis, error sensitivity is represented by the Jacobian matrix. Derivations of the Jacobian matrix can be carried out after all the D-H models are established. For the TAU robot, the 3-DOF kinematic problem will become a 6-DOF kinematic problem. The kinematic problem becomes more complicated. In fact, the error sensitivity is formulated through $\dfrac{\partial x}{\partial g_i}$, $\dfrac{\partial y}{\partial g_i}$, $\dfrac{\partial z}{\partial g_i}$ where x, y, z represent the position of the tool plate and dg_i is the error source for each component. So the following equations can be obtained:

$$dx = \sum_1^N \frac{\partial x}{\partial l_i} dg_i \tag{16}$$

$$dy = \sum_1^N \frac{\partial y}{\partial l_i} dg_i \tag{17}$$

$$dz = \sum_1^N \frac{\partial z}{\partial l_i} dg_i \tag{18}$$

The error model is actually a 6-DOF model since all error sources have been considered. It includes both the position variables X, Y, Z and also rotational angles α, β, γ. From the six kinematic chains, the equations established based on D-H models are

$$f_1 = f_1(x, y, z, \alpha, \beta, \gamma, g) = 0$$
$$f_2 = f_2(x, y, z, \alpha, \beta, \gamma, g) = 0$$
$$..$$
$$f_6 = f_6(x, y, z, \alpha, \beta, \gamma, g) = 0$$

Differentiating all the equations against all the variables $x, y, z, \alpha, \beta, \gamma$ and g, where g is a vector including all geometric design parameters:

$$\frac{\partial f_i}{\partial x} \cdot dx + \frac{\partial f_i}{\partial y} \cdot dy + \frac{\partial f_i}{\partial z} \cdot dz + \frac{\partial f_i}{\partial \alpha} \cdot d\alpha + \frac{\partial f_i}{\partial \beta} \cdot d\beta + \frac{\partial f_i}{\partial \gamma} \cdot d\gamma + \sum_j \frac{\partial f_i}{\partial g_j} \cdot dg_j = 0 \tag{19}$$

Rewrite it in matrix as

$$
\begin{bmatrix}
\frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} & \frac{\partial f_1}{\partial \alpha} & \frac{\partial f_1}{\partial \beta} & \frac{\partial f_1}{\partial \gamma} \\
\frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} & \frac{\partial f_2}{\partial \alpha} & \frac{\partial f_2}{\partial \beta} & \frac{\partial f_2}{\partial \gamma} \\
\frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} & \frac{\partial f_3}{\partial \alpha} & \frac{\partial f_3}{\partial \beta} & \frac{\partial f_3}{\partial \gamma} \\
\frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial z} & \frac{\partial f_4}{\partial \alpha} & \frac{\partial f_4}{\partial \beta} & \frac{\partial f_4}{\partial \gamma} \\
\frac{\partial f_5}{\partial x} & \frac{\partial f_5}{\partial y} & \frac{\partial f_5}{\partial z} & \frac{\partial f_5}{\partial \alpha} & \frac{\partial f_5}{\partial \beta} & \frac{\partial f_5}{\partial \gamma} \\
\frac{\partial f_6}{\partial x} & \frac{\partial f_6}{\partial y} & \frac{\partial f_6}{\partial z} & \frac{\partial f_6}{\partial \alpha} & \frac{\partial f_6}{\partial \beta} & \frac{\partial f_6}{\partial \gamma}
\end{bmatrix}
\bullet
\begin{bmatrix} dx \\ dy \\ dz \\ d\alpha \\ d\beta \\ d\gamma \end{bmatrix}
=
\begin{bmatrix}
\sum_j \frac{-\partial f_1}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_2}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_3}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_4}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_5}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_6}{\partial g_j} dg_j
\end{bmatrix}
\tag{20}
$$

In a compact form, it becomes

$$J_1 dX = dG \tag{21}$$

Where

$$
dG =
\begin{bmatrix}
\sum_j \frac{-\partial f_1}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_2}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_3}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_4}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_5}{\partial g_j} dg_j \\
\sum_j \frac{-\partial f_6}{\partial g_j} dg_j
\end{bmatrix}
= -
\begin{bmatrix}
\frac{\partial f_1}{\partial g_1} & \frac{\partial f_1}{\partial g_2} & \cdots & \frac{\partial f_1}{\partial g_N} \\
\cdot & \cdot & \cdots & \cdot \\
\cdot & \cdot & \cdots & \cdot \\
\frac{\partial f_6}{\partial g_1} & \frac{\partial f_6}{\partial g_2} & \cdots & \frac{\partial f_6}{\partial g_N}
\end{bmatrix}_{6 \times N}
\bullet
\begin{bmatrix} dg_1 \\ dg_2 \\ \cdot \\ \cdot \\ \cdot \\ dg_N \end{bmatrix}_{N \times 1}
\tag{22}
$$

From Equation (22) above, we have,

$$dG = J_2 dg \tag{23}$$

Substitute Equation (21) into Equation (23) to obtain

$$J_1 dX = J_2 dg \tag{24}$$

$$dX = (J_1^{-1} J_2) dg \tag{25}$$

The Jacobian matrix is obtained as $J_1^{-1} \cdot J_2$

$$J = J_1^{-1} \cdot J_2 = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} & \frac{\partial f_1}{\partial \alpha} & \frac{\partial f_1}{\partial \beta} & \frac{\partial f_1}{\partial \gamma} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} & \frac{\partial f_2}{\partial \alpha} & \frac{\partial f_2}{\partial \beta} & \frac{\partial f_2}{\partial \gamma} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} & \frac{\partial f_3}{\partial \alpha} & \frac{\partial f_3}{\partial \beta} & \frac{\partial f_3}{\partial \gamma} \\ \frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial z} & \frac{\partial f_4}{\partial \alpha} & \frac{\partial f_4}{\partial \beta} & \frac{\partial f_4}{\partial \gamma} \\ \frac{\partial f_5}{\partial x} & \frac{\partial f_5}{\partial y} & \frac{\partial f_5}{\partial z} & \frac{\partial f_5}{\partial \alpha} & \frac{\partial f_5}{\partial \beta} & \frac{\partial f_5}{\partial \gamma} \\ \frac{\partial f_6}{\partial x} & \frac{\partial f_6}{\partial y} & \frac{\partial f_6}{\partial z} & \frac{\partial f_6}{\partial \alpha} & \frac{\partial f_6}{\partial \beta} & \frac{\partial f_6}{\partial \gamma} \end{bmatrix}^{-1} \bullet \begin{bmatrix} -\frac{\partial f_1}{\partial g_1} & -\frac{\partial f_1}{\partial g_2} & \cdots & -\frac{\partial f_1}{\partial g_N} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ -\frac{\partial f_6}{\partial g_1} & -\frac{\partial f_6}{\partial g_2} & \cdots & -\frac{\partial f_6}{\partial g_N} \end{bmatrix} \qquad (26)$$

For a prototype of the TAU robotic design, the dimension of the Jacobian matrix is 6 by 71. An analytical solution can be obtained and is used in error analysis.

4.3 Newton-Raphson Numerical Method

Because of the number of parameters involved as well as the number of error sources involved, the kinematic problem becomes very complicated. No analytical solution can be obtained but numerical solution. The TAU configuration, however, as a hybrid or a special case of parallel robots, its forward kinematic problem is, therefore, very complicated. The Newton-Raphson method as an effective numerical method can be applied to calculate the forward problem of the TAU robot, with an accurate Jacobian matrix obtained. The Newton-Raphson method is represented by

$$X_{n+1} = X_n - [F'(X_n)]^{-1} \cdot F(X_n) \qquad (27)$$

With the six chain equations obtained before, the following can be obtained

$$[F'(X_n)]^{-1} = \mathrm{Inv} \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} & \frac{\partial f_1}{\partial \alpha} & \frac{\partial f_1}{\partial \beta} & \frac{\partial f_1}{\partial \gamma} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} & \frac{\partial f_2}{\partial \alpha} & \frac{\partial f_2}{\partial \beta} & \frac{\partial f_2}{\partial \gamma} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} & \frac{\partial f_3}{\partial \alpha} & \frac{\partial f_3}{\partial \beta} & \frac{\partial f_3}{\partial \gamma} \\ \frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial z} & \frac{\partial f_4}{\partial \alpha} & \frac{\partial f_4}{\partial \beta} & \frac{\partial f_4}{\partial \gamma} \\ \frac{\partial f_5}{\partial x} & \frac{\partial f_5}{\partial y} & \frac{\partial f_5}{\partial z} & \frac{\partial f_5}{\partial \alpha} & \frac{\partial f_5}{\partial \beta} & \frac{\partial f_5}{\partial \gamma} \\ \frac{\partial f_6}{\partial x} & \frac{\partial f_6}{\partial y} & \frac{\partial f_6}{\partial z} & \frac{\partial f_6}{\partial \alpha} & \frac{\partial f_6}{\partial \beta} & \frac{\partial f_6}{\partial \gamma} \end{bmatrix} \qquad (28)$$

This equation is used later to calculate the forward kinematic problem, and it is also compared with the method described in the next section.

4.4 Jacobian Approximation Method

A quick and efficient analytical solution is still necessary even though an accurate result has been obtained by the N-R method. The N-R result is produced based on iteration of numerical calculation, instead of from an analytical closed form solution. The N-R method is too slow in calculation to be used in on-line real time control. No certain solution is guaranteed in the N-R method. So the Jacobian approximation method is established. Using this method, error analysis, calibration, compensation, and on-line control model can be in turn established. As the TAU robot is based on a 3-DOF configuration, instead of a general Stewart platform, the Jacobian approximate modification can be obtained based the 3-DOF analytical solution without any errors. The mathematical description of the Jacobian approximation method can be described as follows.

For forward kinematics,

$$X = F(\theta, \varepsilon)$$
$$X = F(\theta, 0) + J_{FORWARD} \cdot d\varepsilon$$

(29)

Where $J_{FORWARD} = F^{'}(\theta, \varepsilon)$ and ε represents error. Thus, the analytical solution $F(\theta, 0)$ and $F(X, 0)$, is obtained. Therefore, the Jacobian Approximation as an analytical solution is obtained and is used to solve nonlinear equations instead of using N-R method.

4.5 Jacobian Matrix with a probe

A real tool should be attached on the wrist of robots as robots are used for any application. Here a probe means a real tool.
From the six kinematic chains, the equations established based on D-H models are

$$f_1 = f_1(x, y, z, \alpha, \beta, \gamma, g) = 0$$
$$f_2 = f_2(x, y, z, \alpha, \beta, \gamma, g) = 0$$
$$..$$
$$f_6 = f_6(x, y, z, \alpha, \beta, \gamma, g) = 0$$

(30)

Differentiating all the equations against all the variables $x, y, z, \alpha, \beta, \gamma$ and g, where g is a vector including all geometric design parameters:

$$\frac{\partial f_i}{\partial x} \cdot dx + \frac{\partial f_i}{\partial y} \cdot dy + \frac{\partial f_i}{\partial z} \cdot dz + \frac{\partial f_i}{\partial \alpha} \cdot d\alpha + \frac{\partial f_i}{\partial \beta} \cdot d\beta + \frac{\partial f_i}{\partial \gamma} \cdot d\gamma + \sum_j \frac{\partial f_i}{\partial g_j} \cdot dg_j = 0 \qquad (19)$$

Rewrite it in matrix as

$$\begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_1}{\partial z} & \dfrac{\partial f_1}{\partial \alpha} & \dfrac{\partial f_1}{\partial \beta} & \dfrac{\partial f_1}{\partial \gamma} \\[2mm] \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} & \dfrac{\partial f_2}{\partial z} & \dfrac{\partial f_2}{\partial \alpha} & \dfrac{\partial f_2}{\partial \beta} & \dfrac{\partial f_2}{\partial \gamma} \\[2mm] \dfrac{\partial f_3}{\partial x} & \dfrac{\partial f_3}{\partial y} & \dfrac{\partial f_3}{\partial z} & \dfrac{\partial f_3}{\partial \alpha} & \dfrac{\partial f_3}{\partial \beta} & \dfrac{\partial f_3}{\partial \gamma} \\[2mm] \dfrac{\partial f_4}{\partial x} & \dfrac{\partial f_4}{\partial y} & \dfrac{\partial f_4}{\partial z} & \dfrac{\partial f_4}{\partial \alpha} & \dfrac{\partial f_4}{\partial \beta} & \dfrac{\partial f_4}{\partial \gamma} \\[2mm] \dfrac{\partial f_5}{\partial x} & \dfrac{\partial f_5}{\partial y} & \dfrac{\partial f_5}{\partial z} & \dfrac{\partial f_5}{\partial \alpha} & \dfrac{\partial f_5}{\partial \beta} & \dfrac{\partial f_5}{\partial \gamma} \\[2mm] \dfrac{\partial f_6}{\partial x} & \dfrac{\partial f_6}{\partial y} & \dfrac{\partial f_6}{\partial z} & \dfrac{\partial f_6}{\partial \alpha} & \dfrac{\partial f_6}{\partial \beta} & \dfrac{\partial f_6}{\partial \gamma} \end{bmatrix} \bullet \begin{bmatrix} dx \\ dy \\ dz \\ d\alpha \\ d\beta \\ d\gamma \end{bmatrix} = \begin{bmatrix} \sum_j \dfrac{-\partial f_1}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_2}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_3}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_4}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_5}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_6}{\partial g_j} dg_j \end{bmatrix} \qquad (20)$$

In a compact form, it becomes

$$J_1 dX = dG \qquad (21)$$

Where

$$dG = \begin{bmatrix} \sum_j \dfrac{-\partial f_1}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_2}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_3}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_4}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_5}{\partial g_j} dg_j \\[2mm] \sum_j \dfrac{-\partial f_6}{\partial g_j} dg_j \end{bmatrix} = - \begin{bmatrix} \dfrac{\partial f_1}{\partial g_1} & \dfrac{\partial f_1}{\partial g_2} & \cdots & \dfrac{\partial f_1}{\partial g_N} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \dfrac{\partial f_6}{\partial g_1} & \dfrac{\partial f_6}{\partial g_2} & \cdots & \dfrac{\partial f_6}{\partial g_N} \end{bmatrix}_{6 \times N} \bullet \begin{bmatrix} dg_1 \\ dg_2 \\ \cdot \\ \cdot \\ dg_N \end{bmatrix}_{N \times 1} \qquad (22)$$

From Equation (22) above, we have

$$dG = J_2 dg \tag{23}$$

Substitute Equation (21) into Equation (23) to obtain

$$J_1 dX = J_2 dg \tag{24}$$

$$dX = (J_1^{-1} J_2) dg \tag{25}$$

The Jacobian matrix is obtained as $J_1^{-1} \cdot J_2$

$$
J = J_1^{-1} \cdot J_2 = \begin{bmatrix}
\dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_1}{\partial z} & \dfrac{\partial f_1}{\partial \alpha} & \dfrac{\partial f_1}{\partial \beta} & \dfrac{\partial f_1}{\partial \gamma} \\
\dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} & \dfrac{\partial f_2}{\partial z} & \dfrac{\partial f_2}{\partial \alpha} & \dfrac{\partial f_2}{\partial \beta} & \dfrac{\partial f_2}{\partial \gamma} \\
\dfrac{\partial f_3}{\partial x} & \dfrac{\partial f_3}{\partial y} & \dfrac{\partial f_3}{\partial z} & \dfrac{\partial f_3}{\partial \alpha} & \dfrac{\partial f_3}{\partial \beta} & \dfrac{\partial f_3}{\partial \gamma} \\
\dfrac{\partial f_4}{\partial x} & \dfrac{\partial f_4}{\partial y} & \dfrac{\partial f_4}{\partial z} & \dfrac{\partial f_4}{\partial \alpha} & \dfrac{\partial f_4}{\partial \beta} & \dfrac{\partial f_4}{\partial \gamma} \\
\dfrac{\partial f_5}{\partial x} & \dfrac{\partial f_5}{\partial y} & \dfrac{\partial f_5}{\partial z} & \dfrac{\partial f_5}{\partial \alpha} & \dfrac{\partial f_5}{\partial \beta} & \dfrac{\partial f_5}{\partial \gamma} \\
\dfrac{\partial f_6}{\partial x} & \dfrac{\partial f_6}{\partial y} & \dfrac{\partial f_6}{\partial z} & \dfrac{\partial f_6}{\partial \alpha} & \dfrac{\partial f_6}{\partial \beta} & \dfrac{\partial f_6}{\partial \gamma}
\end{bmatrix}^{-1} \bullet
$$

$$\tag{26}$$

$$
\begin{bmatrix}
-\dfrac{\partial f_1}{\partial g_1} & -\dfrac{\partial f_1}{\partial g_2} & \cdots & -\dfrac{\partial f_1}{\partial g_N} \\
\vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \cdots & \vdots \\
-\dfrac{\partial f_6}{\partial g_1} & -\dfrac{\partial f_6}{\partial g_2} & \cdots & -\dfrac{\partial f_6}{\partial g_N}
\end{bmatrix}
$$

In the case with a probe on the end effecter:

From the Jacobian matrix , $dX = (J_1^{-1} J_2) dL$, transfer the coordinate of TCP

into the probe coordinates Xp, Yp and Zp as

$$
\begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & x \\ R_{21} & R_{22} & R_{23} & y \\ R_{31} & R_{32} & R_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix} \tag{27}
$$

Differentiating Equation (20), one can obtain:

$$
\begin{bmatrix} dX_P \\ dY_P \\ dZ_P \end{bmatrix} = DR \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix} + R \cdot \begin{bmatrix} dx_L \\ dy_L \\ dz_L \end{bmatrix} + \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix}
$$
(28)

Where

$$
DR_{i,j} = R_{i,j}^{\alpha} \cdot d\alpha + R_{i,j}^{\beta} \cdot d\beta + R_{i,j}^{\gamma} \cdot d\gamma
$$

$$
R_{i,j}^{\alpha} = \frac{dR_{i,j}}{d\alpha} \qquad R_{i,j}^{\beta} = \frac{dR_{i,j}}{d\beta} \qquad R_{i,j}^{\gamma} = \frac{dR_{i,j}}{d\gamma}
$$
(31)

Rewrite the equation into following forms,

$$
\begin{bmatrix} dX_P \\ dY_P \\ dZ_P \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & M_{11} & M_{12} & M_{13} \\ 0 & 1 & 0 & M_{21} & M_{22} & M_{23} \\ 0 & 0 & 1 & M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \\ d\alpha \\ d\beta \\ d\gamma \end{bmatrix} + R \cdot \begin{bmatrix} dx_L \\ dy_L \\ dz_L \end{bmatrix}
$$
(32)

Where $M_{ij} = DR \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix}$ then substitute $dX = (J_1^{-1} J_2) dL$ into Equation (32)

Finally

$$
\begin{bmatrix} dX_P \\ dY_P \\ dZ_P \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 & M_{11} & M_{12} & M_{13} \\ 0 & 1 & 0 & M_{21} & M_{22} & M_{23} \\ 0 & 0 & 1 & M_{31} & M_{32} & M_{33} \end{bmatrix} \cdot J & R \end{bmatrix} \cdot \begin{bmatrix} dL_1 \\ dL_2 \\ . \\ . \\ dL_N \\ dx_L \\ dy_L \\ dz_L \end{bmatrix}
$$

The final Jacobian matrix with a probe is $\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 & M_{11} & M_{12} & M_{13} \\ 0 & 1 & 0 & M_{21} & M_{22} & M_{23} \\ 0 & 0 & 1 & M_{31} & M_{32} & M_{33} \end{bmatrix} \cdot J & R \end{bmatrix}$

4.6 Inverse Jacobian Matrix with a Probe

From 6 link length equations below:

$$f_1 = f_1(\theta_1, \theta_2, \theta_3, x, y, z, \alpha, \beta, \gamma, g_1) = 0$$
$$f_2 = f_2(\theta_1, \theta_2, \theta_3, x, y, z, \alpha, \beta, \gamma, g_1) = 0 \tag{33}$$
$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$
$$f_6 = f_6(\theta_1, \theta_2, \theta_3, x, y, z, \alpha, \beta, \gamma, g_1) = 0$$

where $\theta_1, \theta_2, \theta_3$, are drive angles from actuators or motors and $x, y, z, \alpha, \beta, \gamma$ is the pose of TCP.

with the probe, one can obtain the next three equations from Euler transformation.

$$\begin{aligned} f_7 &= f_7(x, y, z, \alpha, \beta, \gamma, g_2) = 0 \\ f_8 &= f_8(x, y, z, \alpha, \beta, \gamma, g_2) = 0 \\ f_9 &= f_9(x, y, z, \alpha, \beta, \gamma, g_2) = 0 \end{aligned} \quad \text{from} \quad \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} & & & x \\ R & & & y \\ & & & z \\ 0 & & & 1 \end{bmatrix} \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix} \tag{34}$$

Differentiate with respect to all the variables $\theta_1, \theta_2, \theta_3, x, y, z, \alpha, \beta, \gamma$ for Equation (30), where g_i is a vector including all design variables:

$$df_i = 0 \quad \text{or}$$

$$\frac{\partial f_i}{\partial \theta_1} \cdot d\theta_1 + \frac{\partial f_i}{\partial \theta_2} \cdot d\theta_2 + \frac{\partial f_i}{\partial \theta_3} \cdot d\theta_3 + \frac{\partial f_i}{\partial x} \cdot dx + \frac{\partial f_i}{\partial y} \cdot dy + \frac{\partial f_i}{\partial z} \cdot dz + \frac{\partial f_i}{\partial \alpha} \cdot d\alpha + \frac{\partial f_i}{\partial \beta} \cdot d\beta + \frac{\partial f_i}{\partial \gamma} \cdot d\gamma + \frac{\partial f_i}{\partial g_1} \cdot dg_1 = 0 \tag{35}$$

Rewrite it in a matrix form

$$J_1 d\theta + J_2 dX = -\partial G_1 \tag{36}$$

$$\text{Where } J_1 = \begin{bmatrix} \dfrac{\partial f_1}{\partial \theta_1} & \dfrac{\partial f_1}{\partial \theta_2} & \dfrac{\partial f_1}{\partial \theta_3} \\[2ex] \dfrac{\partial \dot{f}_6}{\partial \theta_1} & \dfrac{\partial \dot{f}_6}{\partial \theta_2} & \dfrac{\partial \dot{f}_6}{\partial \theta_3} \end{bmatrix} \tag{37}$$

$$J_2 = \begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_1}{\partial z} & \dfrac{\partial f_1}{\partial \alpha} & \dfrac{\partial f_1}{\partial \beta} & \dfrac{\partial f_1}{\partial \gamma} \\[2ex] \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} & \dfrac{\partial f_2}{\partial z} & \dfrac{\partial f_2}{\partial \alpha} & \dfrac{\partial f_2}{\partial \beta} & \dfrac{\partial f_2}{\partial \gamma} \\[2ex] \dfrac{\partial f_3}{\partial x} & \dfrac{\partial f_3}{\partial y} & \dfrac{\partial f_3}{\partial z} & \dfrac{\partial f_3}{\partial \alpha} & \dfrac{\partial f_3}{\partial \beta} & \dfrac{\partial f_3}{\partial \gamma} \\[2ex] \dfrac{\partial f_4}{\partial x} & \dfrac{\partial f_4}{\partial y} & \dfrac{\partial f_4}{\partial z} & \dfrac{\partial f_4}{\partial \alpha} & \dfrac{\partial f_4}{\partial \beta} & \dfrac{\partial f_4}{\partial \gamma} \\[2ex] \dfrac{\partial f_5}{\partial x} & \dfrac{\partial f_5}{\partial y} & \dfrac{\partial f_5}{\partial z} & \dfrac{\partial f_5}{\partial \alpha} & \dfrac{\partial f_5}{\partial \beta} & \dfrac{\partial f_5}{\partial \gamma} \\[2ex] \dfrac{\partial f_6}{\partial x} & \dfrac{\partial f_6}{\partial y} & \dfrac{\partial f_6}{\partial z} & \dfrac{\partial f_6}{\partial \alpha} & \dfrac{\partial f_6}{\partial \beta} & \dfrac{\partial f_6}{\partial \gamma} \end{bmatrix} \tag{38}$$

$$-\partial G_1 = \begin{bmatrix} \dfrac{-\partial f_1}{\partial g_1} dg_1 \\[2ex] \dfrac{-\partial f_2}{\partial g_1} dg_1 \\[2ex] \dfrac{-\partial f_3}{\partial g_1} dg_1 \\[2ex] \dfrac{-\partial f_4}{\partial g_1} dg_1 \\[2ex] \dfrac{-\partial f_5}{\partial g_1} dg_1 \\[2ex] \dfrac{-\partial f_6}{\partial g_1} dg_1 \end{bmatrix} \tag{39}$$

From Equation (33), one can obtain next formulation.

$$J_3 dX = -\partial G_2 \tag{40}$$

$$\text{Where } J_3 = \begin{bmatrix} \dfrac{\partial f_7}{\partial x} & \dfrac{\partial f_7}{\partial y} & \dfrac{\partial f_7}{\partial z} & \dfrac{\partial f_7}{\partial \alpha} & \dfrac{\partial f_7}{\partial \beta} & \dfrac{\partial f_7}{\partial \gamma} \\[2ex] \dfrac{\partial f_8}{\partial x} & \dfrac{\partial f_8}{\partial y} & \dfrac{\partial f_8}{\partial z} & \dfrac{\partial f_8}{\partial \alpha} & \dfrac{\partial f_8}{\partial \beta} & \dfrac{\partial f_8}{\partial \gamma} \\[2ex] \dfrac{\partial f_9}{\partial x} & \dfrac{\partial f_9}{\partial y} & \dfrac{\partial f_9}{\partial z} & \dfrac{\partial f_9}{\partial \alpha} & \dfrac{\partial f_9}{\partial \beta} & \dfrac{\partial f_9}{\partial \gamma} \end{bmatrix} \tag{41}$$

$$-\partial G_2 = \begin{bmatrix} \dfrac{-\partial f_7}{\partial g_2} dg_2 \\ \dfrac{-\partial f_8}{\partial g_2} dg_2 \\ \dfrac{-\partial f_9}{\partial g_2} dg_2 \end{bmatrix} \tag{42}$$

From Equation (36)

$$dX = -J_2^{-1} \cdot \partial G_1 - J_2^{-1} \cdot J_1 d\theta \tag{43}$$

Substituting Equation (43) into the Equation (36)

$$J_3 \cdot J_2^{-1} \cdot J_1 \cdot d\theta = J_5 \cdot \begin{bmatrix} dx_L \\ dy_L \\ dz_L \end{bmatrix} - J_3 \cdot J_2^{-1} J_4 \cdot \begin{bmatrix} dL_1 \\ dL_2 \\ . \\ dL_N \end{bmatrix} \tag{44}$$

Where $J_4 = \begin{bmatrix} \dfrac{\partial f_1}{\partial L_1} & \cdots & \dfrac{\partial f_1}{\partial L_N} \\ \cdots & \cdots & \cdots \\ \dfrac{\partial f_6}{\partial L_1} & \cdots & \dfrac{\partial f_6}{\partial L_N} \end{bmatrix}$ (45)

and

$$J_5 = \begin{bmatrix} \dfrac{\partial f_7}{\partial x_L} & \dfrac{\partial f_7}{\partial y_L} & \dfrac{\partial f_7}{\partial z_L} \\ \dfrac{\partial f_8}{\partial x_L} & \dfrac{\partial f_8}{\partial y_L} & \dfrac{\partial f_8}{\partial z_L} \\ \dfrac{\partial f_9}{\partial x_L} & \dfrac{\partial f_9}{\partial y_L} & \dfrac{\partial f_9}{\partial z_L} \end{bmatrix} \tag{46}$$

Finally

$$d\theta = [J_3 \cdot J_2^{-1} \cdot J_1]^{-1} \cdot [-J_3 \cdot J_2^{-1} \cdot J_4 \,\vdots\, J_5] \cdot \begin{bmatrix} dL_1 \\ \cdot \\ \cdot \\ dL_N \\ dx_L \\ dy_L \\ dz_L \end{bmatrix} \tag{47}$$

$$J_{INVERSE} = [J_3 \cdot J_2^{-1} \cdot J_1]^{-1} \cdot [-J_3 \cdot J_2^{-1} \cdot J_4 \,\vdots\, J_5]$$

4.7 Determination of Independent Design Variables Using SVD Method

With the reality that all the parts of a robot have manufacturing errors and misalignment errors as well as thermal errors, errors should be considered for any of the components in order to accurately model the accuracy of the robotic system. Error budget is carried out in the study and error sensitivity of robot kinematics with respect to any of the parameters can be obtained based on error modeling. This is realized through the established Jacobian matrix.

To find those parameters in the error model that are linearly dependent and those parameters that are difficult to observe, the Jacobian matrix is analyzed. SVD method (Singular Value Decomposition) is used in such an analysis.

A methodical way of determining which parameters are redundant is to investigate the singular vectors. An investigation of the last column of the V vector will reveal that some elements are dominant in order of magnitude. This implies that corresponding columns in the Jacobian matrix are linearly dependent. The work of reducing the number of error parameters must continue until no singularities exist and the condition number has reached an acceptable value.

A total of 31 redundant design variables of the 71 design parameters are eliminated by observing the numerical Jacobian matrix obtained. Table 7 in Section 6 lists the remaining calibration parameters.

4.8 Error Budget

When the SVD is completed and a linearly independent set of error model parameters determined, the Error Budget can be determined. The mathematical description of the error budget is as follows:

$$J = U \bullet S \bullet V^T$$

$$dX = J \bullet dg = U \bullet S \bullet V^T \bullet dg \qquad (48)$$

$$U^T \bullet dX = S \bullet V^T \bullet dg$$

Assume $U^T \bullet dX = d\overline{X}$ and $V^T \bullet dg = d\overline{g}$. So we have $d\overline{g} = d\overline{X} / S_{ii}$, finally,

$$dg = (V \bullet U^T \bullet dX) / S_{ii} \qquad (49)$$

Thus if the dX is given as the accuracy of the TAU robot, the error budget dg can be determined.

Given the D-H parameters for all three upper arms and the main column, the locations of the joints located at each of the three upper arms can be known accurately. The six chain equations are created for the six link lengths, as follows:

$$F = \left\{ \begin{bmatrix} f1(upperarm\_points, TCP\_points) \\ f2(upperarm\_points, TCP\_points) \\ f3(upperarm\_points, TCP\_points) \\ f4(upperarm\_points, TCP\_points) \\ f5(upperarm\_points, TCP\_points) \\ f6(upperarm\_points, TCP\_points) \end{bmatrix} \right.$$

Where

$$TCP\_point = f(px, py, pz, \alpha, \beta, \gamma)$$

$$Upperarm\_point = f(\varepsilon)$$

and ε is a collection of all the design parameters. Thus,

$$F = \left\{ \begin{bmatrix} F1(\varepsilon, px, py, pz, \alpha, \beta, \gamma) \\ F2(\varepsilon, px, py, pz, \alpha, \beta, \gamma) \\ F3(\varepsilon, px, py, pz, \alpha, \beta, \gamma) \\ F4(\varepsilon, px, py, pz, \alpha, \beta, \gamma) \\ F5(\varepsilon, px, py, pz, \alpha, \beta, \gamma) \\ F6(\varepsilon, px, py, pz, \alpha, \beta, \gamma) \end{bmatrix} \right. \qquad (50)$$

An error model is developed based on the system of equations as described above. A total of 71 parameters are defined to represent the entire system. The 71 parameters include all the D-H parameters for the 3 upper arms, as well as the coordinates (x, y, z) of the 6 points at both ends of the 6 links, respectively. Table 8 in Section 6 presents the error budget.

4.9 Dexterity Analysis

From the inverse kinematics ,

$$S_i = RP_i^h \tag{51}$$

Where P_i^h denotes the position of the center on the end plate in local coordinate. R is the

transfer matrix of coordinate. So the link vector

$$L_i = P_h + S_i - P_i^b \tag{52}$$

P_h is the position coordinate of the center on the end plate. From the end plate velocities

to link velocities, We define the Jacobian matrix by

$$\dot{L} = J\dot{X} \tag{53}$$

Where $\dot{L}$ is the vector of link velocities and $\dot{X} = [\dot{P}_h^T, \omega^T]^T$ is the velocity vector
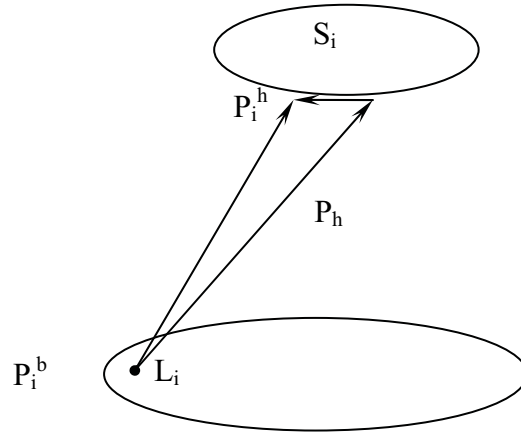
Figure 17. Vectors

Differentiating Equation (52) we can get

$$\dot{l}_i z_i + l_i \dot{z}_i = \dot{p}_h + (\omega \times S_i) \quad \text{(Zi is the unit vector of Li vector)} \tag{54}$$

Taking the inner product with Zi yields

$$\dot{l}_i = z_i \bullet \dot{p}_h + (S_i \times z_i) \bullet \omega \quad (\text{ from the } z_i \bullet (\omega \times s_i) = (s_i \times z_i) \bullet \omega \ ) \tag{55}$$

Thus we can obtain the Jacobian matrix as

$$J = \begin{bmatrix} z_1^T & (s_1 \times z_1)^T \\ .. & .. \\ z_6^T & (s_6 \times z_6)^T \end{bmatrix} \tag{56}$$

The dexterity is defined as

$$A = \frac{\lambda_{max}(J^{-1})}{\lambda_{min}(J^{-1})} \tag{57}$$

where λ is the eigen-value of the Jacobian matrix.

5. System Stiffness

The stiffness of the robot is a very important performance, which will have a significant influence on the robotic applications like cutting, milling, grinding

etc. In this chapter, general formulations for the stiffness of robotic system and the stiffness measurement result are presented, TCP stiffness is calculated based on theoretical analysis and modeling. In the stiffness analysis, the stiffness of individual component in related directions will be the output of stiffness model.

5.1 The Measurement of the Robot Stiffness

Based on the designed robot with certain component errors, Error modeling will be used to map the robot error over its working space. Thermal model will also be established. Deflection under load will be part of the modeling too. This comprehensive error model is the base for error analysis and robotic product design. It will also be used, or partly used for error compensation. For error compensation, however, suitable sensors will have to be used.

As measurement is concerned, it is important is to choose the suitable performance evaluation standard. The type of sensors will be selected based on the evaluation method. In selecting the sensors, resolution, repeatability, and accuracy under certain environments will be the key to consider. The factors of price and user-friendliness will also be weighted heavily. Measurement procedure will be carefully generated and measurement will be performed using certified metrology equipment only to ensure the results.

5.2 Formulations of the Robotic System Stiffness

A solution to the inverse kinematics problem is required for stiffness calculation. It is briefly described below. Referring to standard Stewart Platform the *i-th* leg length *li* is given by

$$li = gi(R,d) \tag{58}$$

where *d = [x,y,z]*, is the position vector of the platform coordinate system's origin in the base coordinate system, *li* is the length of the *i-th* leg and *gi* is only a function of *R* and *d* for constant geometric the *i-th* leg parameters.

$$R = \begin{bmatrix} \cos\phi vos\theta & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \sin\phi\sin\theta & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi \\ -\sin\phi & \cos\theta\sin\psi & \cos\theta\cos\psi \end{bmatrix} \tag{59}$$

Above is the rotation matrix relating the platform's coordinate system, to the base co-ordinate system, Here R is constructed using Roll-Pitch-Yaw (RPY) angle rotations, where R *(roll)* = ϕ around the z axis, P (pitch) = θ around the y axis, and Y (yaw) = ψ around the x axis.

 Thus, R is a rotation about the x axis of ψ, followed by θ, a rotation around y axis, and ending with a rotation of ϕ around z axis.

 Equation (58) represents the inverse kinematic solution. For some R and d, the i-th leg length (li) can be easily calculated.

If Equation (58) is expanded using Taylor series expansion, and the first order term considered only, the change in leg length, Δli, is obtained as a row vector Ji, multiplied by the column twist vector Δp as given below:

$$\Delta p = Ji\ \Delta li \tag{60}$$

where

$$J_i = \left[\frac{\partial x}{\partial g_i}, \frac{\partial y}{\partial g_i}, \frac{\partial z}{\partial g_i}, \frac{\partial \psi}{\partial g_i}, \frac{\partial \theta}{\partial g_i}, \frac{\partial \phi}{\partial g_i} \right] \tag{61}$$

And

$$\Delta p = [\Delta x, \Delta y, \Delta z, \Delta \psi, \Delta \theta, \Delta \phi]^T . \tag{62}$$

Assembling the equations for all the legs of the mechanism,

$$\Delta p = J \Delta q \tag{63}$$

where $\Delta q = [\Delta l_1, \Delta l_2, \Delta l_3, \Delta l_4, \Delta l_5, \Delta l_6]^T .$

From the principle of duality between the force/torque and velocity fields, or what is more commonly known as contragradience

$$f = J^T \tau \tag{64}$$

where

$$\tau = [F_x, F_y, F_z, M_x, M_y, M_z]^T$$

is the end effector wrench, and

$$f = [f_1, f_2, f_3, f_4, f_5, f_6]^T$$

is the vector of forces experienced by the legs, and J^T is the transpose of the Jacobian J, (described earlier).

As previously mentioned, the static stiffness (or rigidity) of the mechanism can be a primary consideration in the design of a parallel link manipulator for certain applications (specifically, those involving large forces and high accuracy).

The static stiffness of the PLM is a function of:

- The limbs' structure and material.
- The joints' stiffnesses.
- The platform and base stiffness.
- The geometry of the structure.
- The topology of the structure.
- The end-effector position and orientation.

To ensure meeting the stiffness specifications, it becomes important to estimate the stiffness, particularly the lowest stiffness value and the direction in which it is experienced, for the manipulator in a given posture or configuration. In the following analysis, this problem is addressed. Algebraic expressions for stiffness (both the engineering and the general, to be defined later) are developed. The fact that the minimum stiffness is experienced in the direction of the eigenvector that corresponds to the minimum eigenvalue of the 'stiffness matrix' of the manipulator is shown. A corresponding result can be obtained for the maximum stiffness of the manipulator. Finally, expressions are developed for the stiffness of the manipulator in any direction.

The basic assumption for the theory developed is:

- The joints are frictionless.
- The weights of the legs or arms are negligible.

The rigidity of the platform and the base is much greater than that of the legs and, therefore, can be considered as infinite (or in general, the manipulator's joints are the least stiff elements in the structure, and hence, dictate the manipulator stiffness). If k is the axial or arm stiffness, then for the i-th leg or arm

$$f_i = k_i \Delta l_i \tag{65}$$

where f_i is the force needed to cause a Δl_i change of the i-th leg length. Assembling the equations for all the legs, Equation (65) becomes

$$f_i = k \Delta q \tag{66}$$

Substituting for Δq from Equation (63)

$$f = k J^{-1} \Delta p \tag{67}$$

Multiplying both sides of Equation (67) with J^{-T} and substituting f with $J^T \tau$ from Equation (64) to obtain

$$\tau = J^{-T}kJ^{-1}\Delta p. \tag{68}$$

Equation (68) can be interpreted as τ is the wrench required to cause the platform to experience a twist of Δp. So the stiffness is obtained as

$$J^{-T}kJ^{-1} \tag{69}$$

5.3 Method for Measuring Joint Stiffness

From Equation (69), the stiffness if the robot can be obtained, including the component or joint stiffness K_i. In order to obtain the total stiffness of the robot, the joint stiffness has to be measured.
From Equation (68), the following Equation (70) can be obtained by finding the inverse of the matrix $J^{-T}kJ^{-1}$ as

$$\Delta p = JK_i^{-1}J^T\tau. \tag{70}$$

Equation (70) is very important for measuring the joint stiffness. Many different equations can be obtained by applying different force τ with different directions then measuring the deflections Δp. Least square method is applied to solve Equation (70). As variable $1/K_i$ is the unknown, one can simplify Equation (70) as linear equations since $K_i^{-1} = \left[1/k_i \right]$ is a diagonal matrix.

5.4 Results of the Stiffness Measurement

The instrument used in measuring includes:

* CMM ROMER 3000i Digitizer with an accuracy of 5μm
* Sphere with an accuracy of 0.02 mm
Pose measurement is carried out first as seen in Fig. 5.1. The conditions are

J1=84.7⁰, J2=-3.6⁰, J3=38.8⁰, J4=-0.3⁰, J5=50.6⁰ and J6=-110.2⁰.
Load F_x=-360N

And the measured deformation is

$\Delta x = -0.69$ mm, $\Delta y = 0.37$ mm, and $\Delta z = -0.13$ mm

Condition for Deflection Measurement:

- Measure robot translational deflections by the position of the center of the sphere, which is calculated based on the measurement result of the portable CMM ROMER.
- Motor servo is active during the measurement to take account of the controller stiffness.

Figure 18. Measurement Set-up
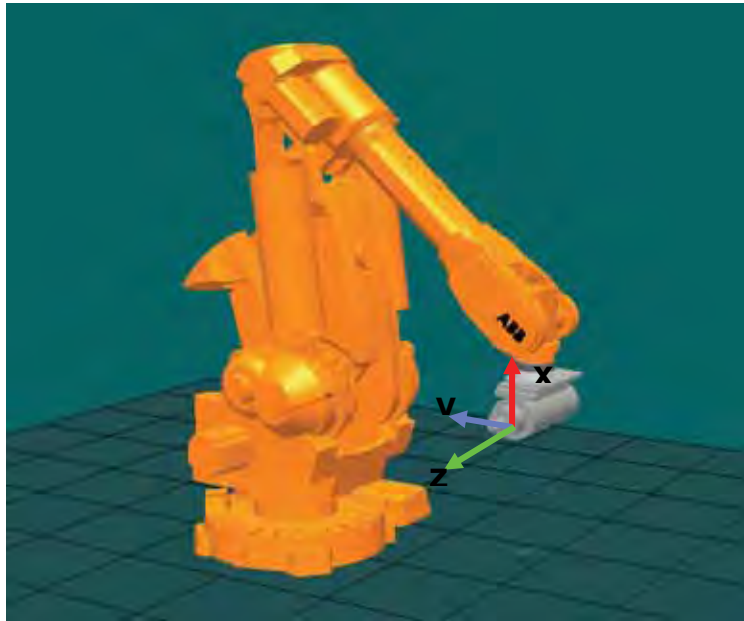
Figure 19 . Configuration of the IRB 4400 Robot
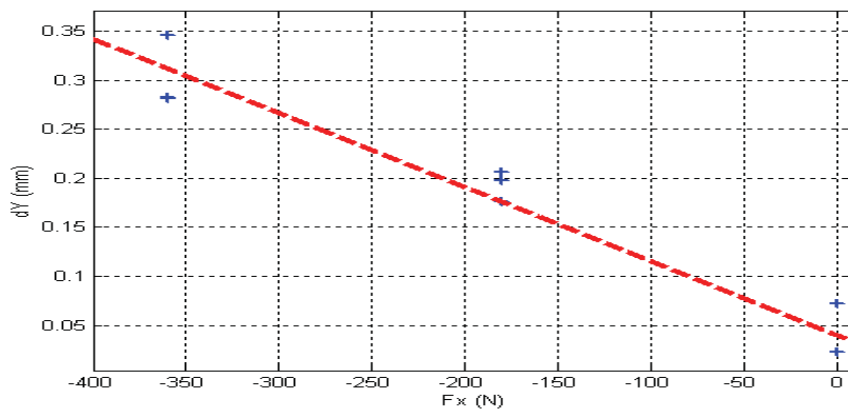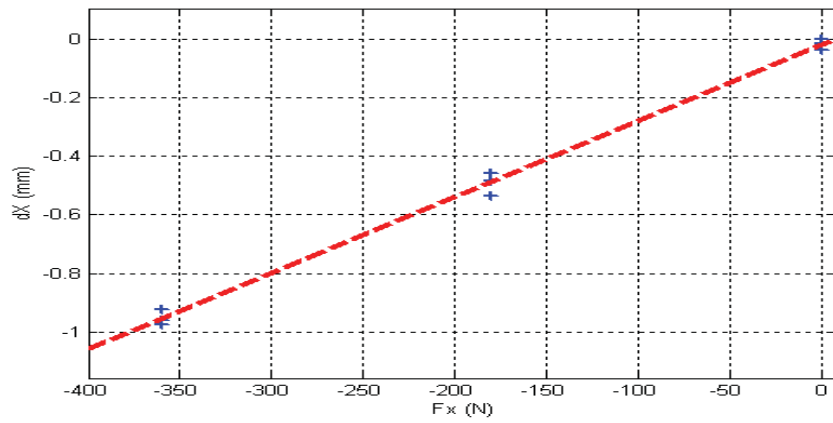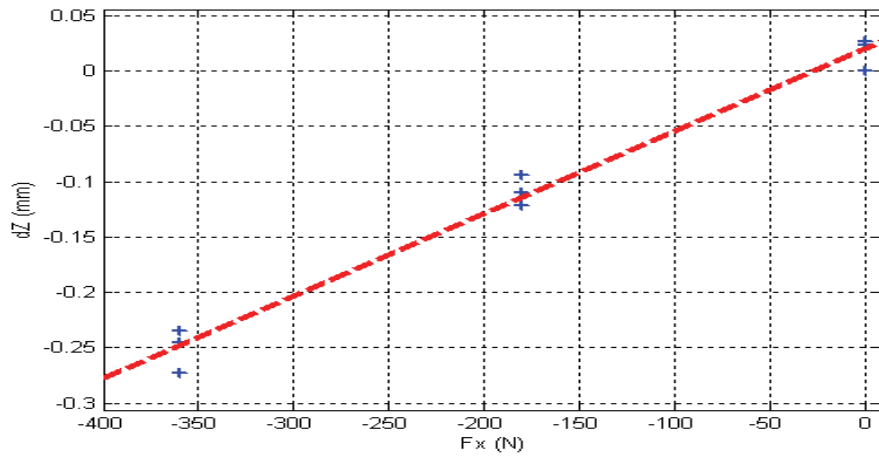
Figure 20. Measured Deflection/Deformation Dx, Dy, and Dz

Another measurement pose, as seen in Fig. 5.4 is J1=45.60, J2=-23.60, J3=37.20, J4=52.10, J5=52.10 and J6=-194.80. Load condition is
Load Fx=-360N
And the measured deformation is
Δx = -1.05 mm, Δy = -0.01 mm and Δz = -0.57 mm
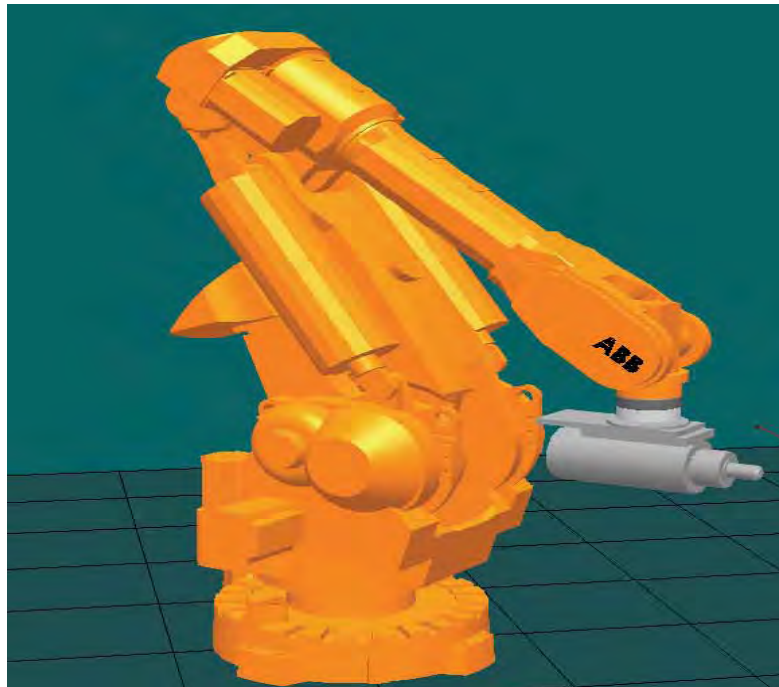
Figure 21. Configuration of the IRB 4400 Robot
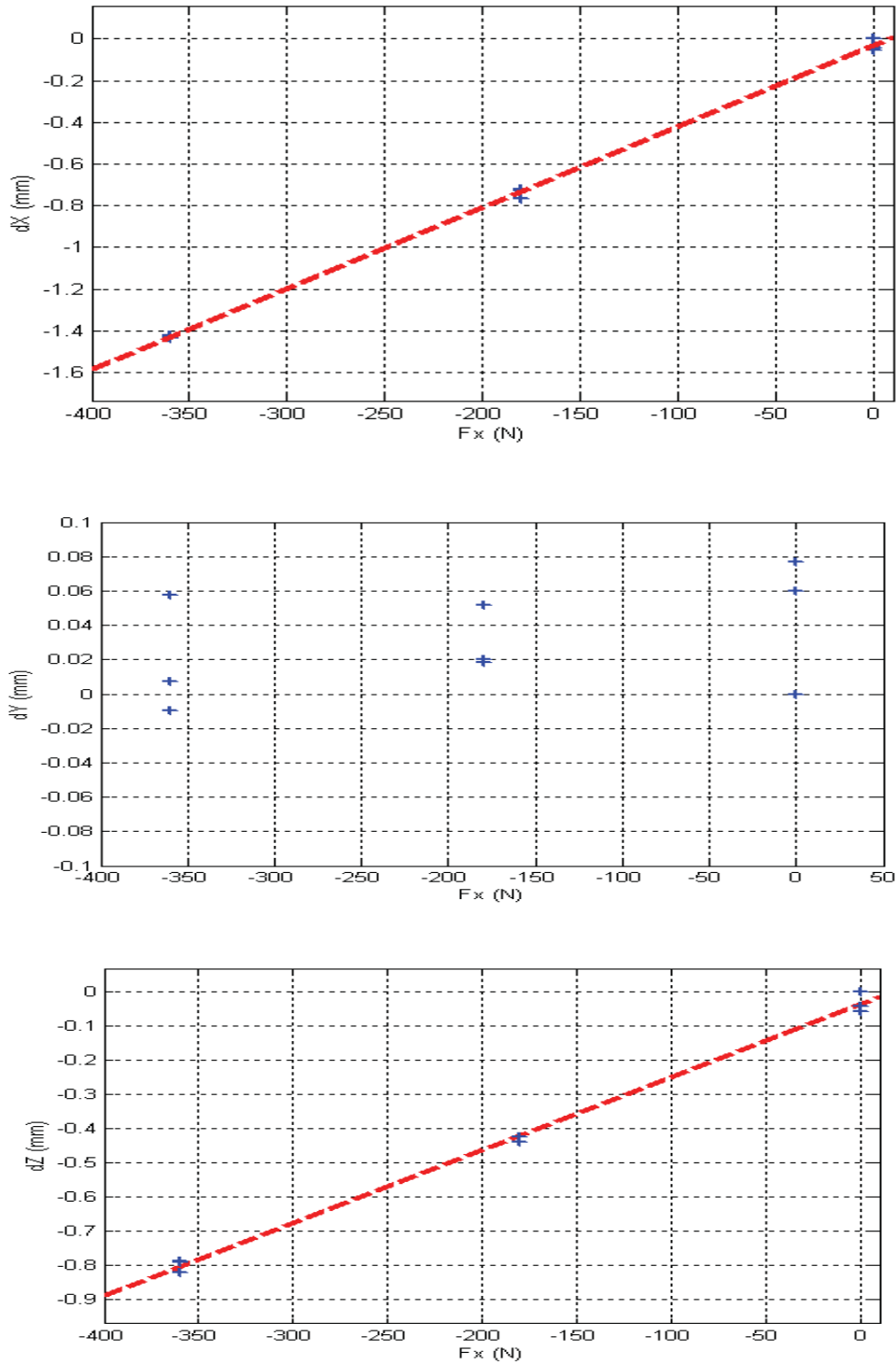
Figure 22. Measured Deflection/Deformation Dx, Dy, and Dz (Second pose)

Fx	Fy	Fz	dx	dy	dz
-180	0	0	-0.4561	0.1767	-0.1211
-360	0	0	-0.9232	0.2812	-0.2723
-360	0	0	-0.9604	0.2825	-0.2452
-180	0	0	-0.4822	0.1983	-0.0943
-180	0	0	-0.5359	0.2062	-0.1103
-360	0	0	-0.9775	0.3464	-0.2344
-180	0	0	-0.7276	0.0201	-0.4238
-360	0	0	-1.423	0.0073	-0.8206
-360	0	0	-1.4246	-0.0099	-0.7893
-180	0	0	-0.768	0.0184	-0.44
-180	0	0	-0.7194	0.0518	-0.4242
-360	0	0	-1.4357	0.0577	-0.7922
0	-275	25	0.0061	-0.8927	0.0336
0	-275	25	-0.0004	-0.9184	-0.0111
-40	-295	10	0.134	-1.1826	-0.0926
-40	-295	10	0.1308	-1.2146	-0.1407
-360	0	0	-0.9344	0.2758	-0.2987

Table 4. Measured Deformation Data

Then solve the K_q in Equation (70) $\Delta x = J K_q^{-1} J^T F$ with the least square method.

The final result is as follows:

	lsqr result	Nominal value
Axis 1:	19.03	22~80
Axis 2:	14.6	32~42
Axis 3:	45.83	25~39
Axis 4:	31.26	70
Axis 5:	15.16	50
Axis 6:	15	85

Table 5.Calculated Joint Stiffness

Fig. 5.6 also gives the standard deviation from the measurement data.

Based on the results, the measurement data can be trusted and the standard deviation of residual error is 0.042mm. Also, verification of solved stiffness agrees well. The stiffness model can provide a method for position compensation to reach a high level accuracy, with a force sensor measuring the process force in real time, the impact on position deformation can be estimated and compensated.
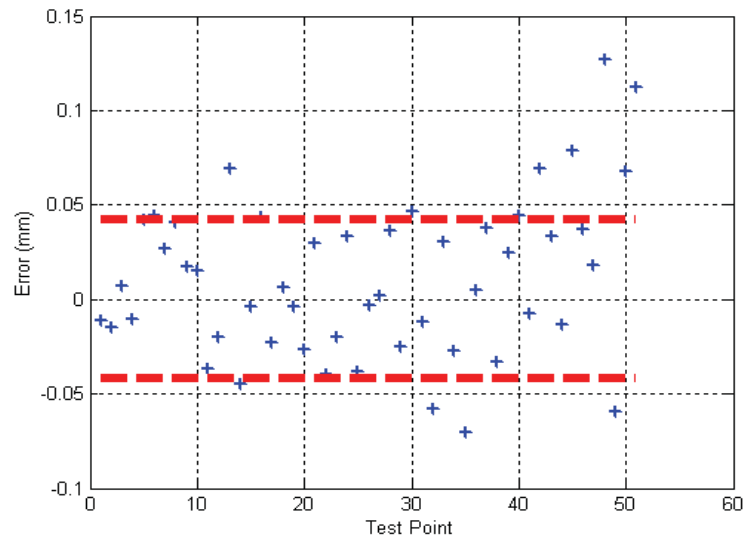
Figure 23. Residual Error Std = 0.042 mm

The same procedure can be applied to the TAU robot. The stiffness at TCP point was measured by applying a load at TCP and measuring the resulting displacements, see Figure 24. The results of the measurements are shown in Figure 25.

Figure 24. Setup of the TAU robot's Stiffness Measurement
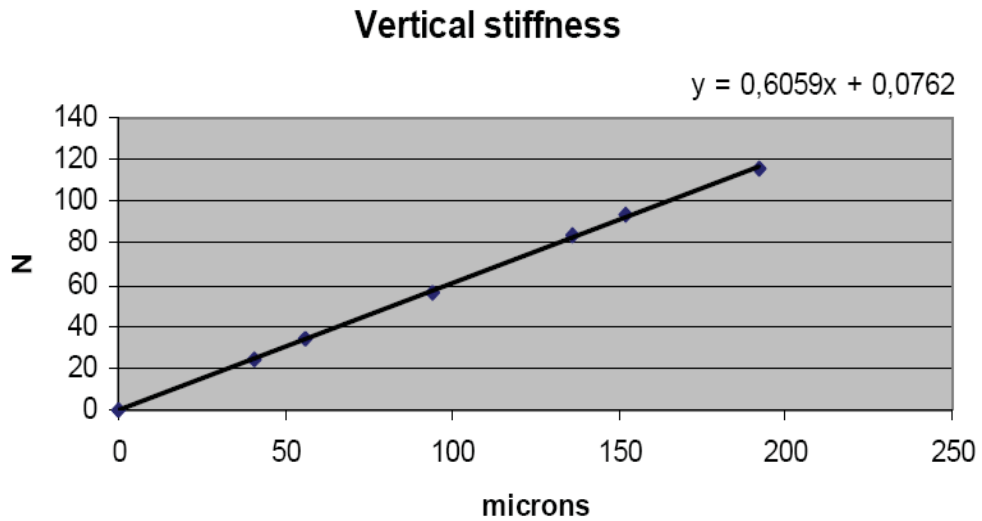
Figure 25. Measured Stiffness of the TAU robot

5.5 Application of the Robot Stiffness: Position Compensation

Position compensation can be made once the stiffness model is established. The application is to compensate the position error caused by the cutting force of milling processing.

Figure 26.Robotic Milling Setup

First, the surface quality of the aluminum block can be recorded as cutting without position compensation then cutting again to measure the surface quality with on line compensation. The surface will be measured via the laser. See Fig. 5.9 for the robotic milling setup. Based on the result shown in Figs. 5.10 and 5.11, the compensation procedure is effective reducing the error to less than 0.1 mm compared with the original error of 0.5 mm.

Conclusions:

Verification of solved stiffness agrees very well. The stiffness model can provide a method to model and test robot stiffness, with a force sensor measures the process force in real time, the impact on position deformation can be estimated and compensated.
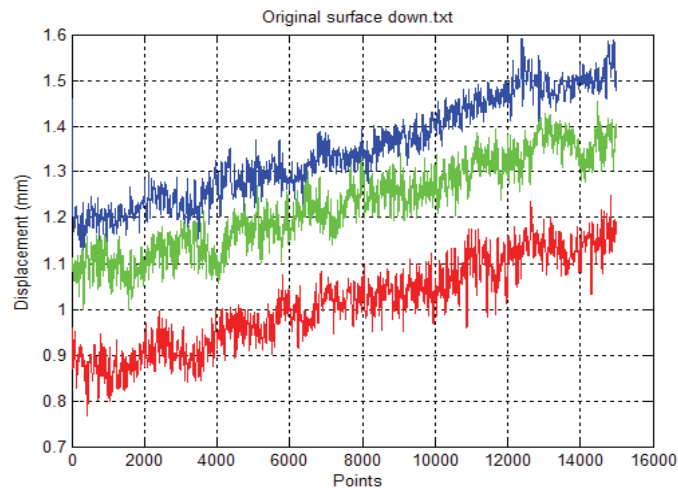
Figure 27. Surface Quality without Compensation
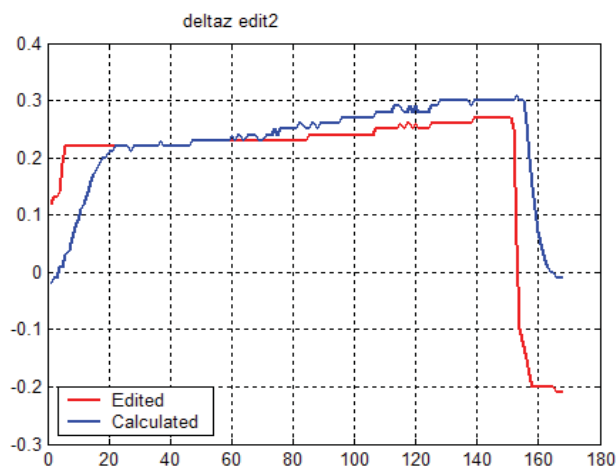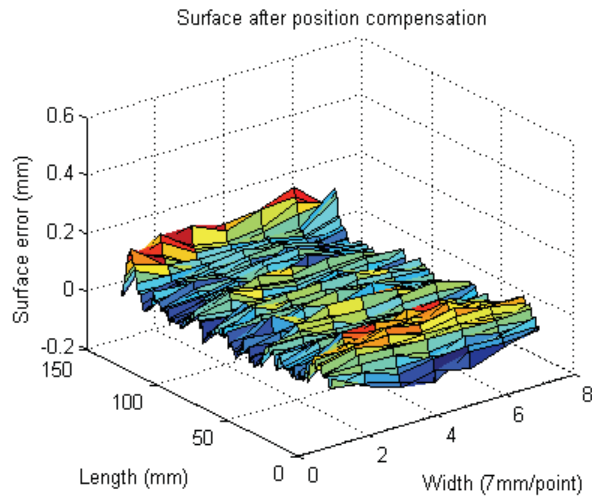
Figure 28. Surface Quality with Compensation, Mean Error < 0.1 mm

6 . Simulation and Experimental Results

The validation of the analytical model has been carried out, as well as the re-
alization of control scheme. Besides the analytical result and data, additional
results used in this chapter come from three sources:
Simulation results from ADAMS simulation software, see Figure 29 for details;
Test results from two-arm test platform, see Table 11;
Test results from the TAU prototype.

6.1 Validation of Jacobian Matrix and N-R Method

The Jacobian Matrix and N-R method need to be verified to guarantee their
correctness. These simulations are made by ADAMS (commercial simulation
software) see Figure 29.
The effect of the robot configurations were considered, all "verification points"
are located in the whole work-space and with total different configurations.
Figures 30, 6.3, and 6.4 show position differences between the N-R method and
the ADAMS simulation, which indicates that accurate results have been ob-
tained up to 0.06 um compared with ADAMS simulation results. These results
guarantee the correctness of Jacobian Matrix and N-R method. Based on the
simulation results, the N-R method with analytical Jacobian matrix can be
used in error modeling, error budget, offline calibration.
Like most of the methods in this thesis this method suffers from a drawback: it
can not be used in online position compensation and online control because it
is an iteration method even with an analytical, full size Jacobian Matrix. Next

section will focus on the Jacobian Approximation Method (JAM), which is able to deal with the online compensation and online control problems.

Figure 29. Using Adams to Verify the Analytical and Error Model
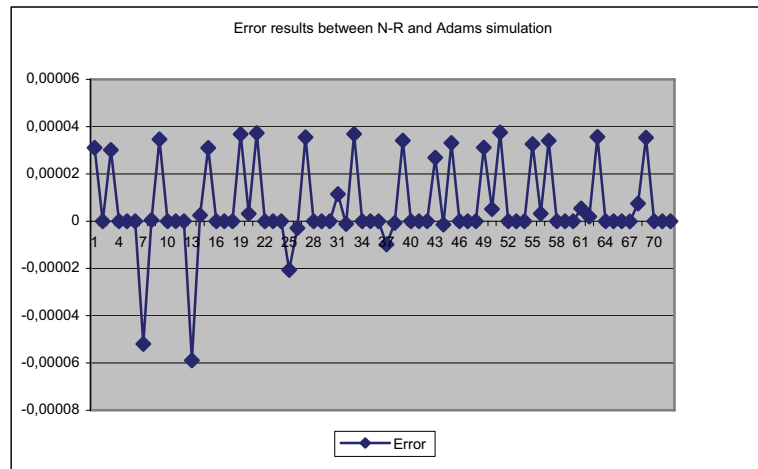
Figure 30. Position Error between the N-R Method and ADAMS Simulation

Comparison between N-R and Adams

Comparison between N-R and Adams

Comparison between N-R and Adams
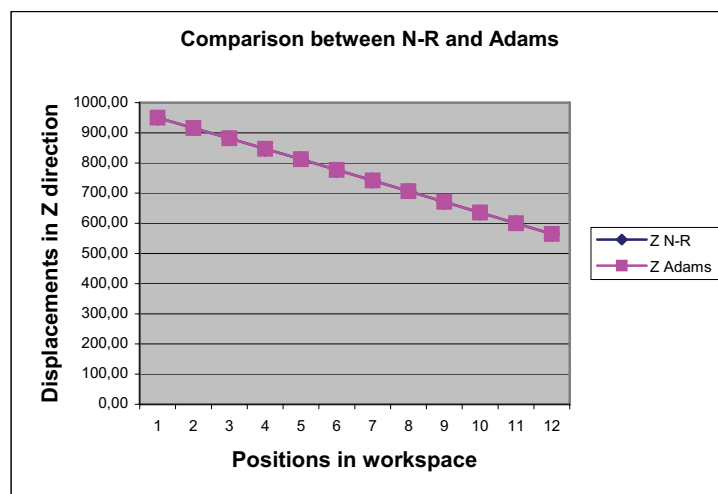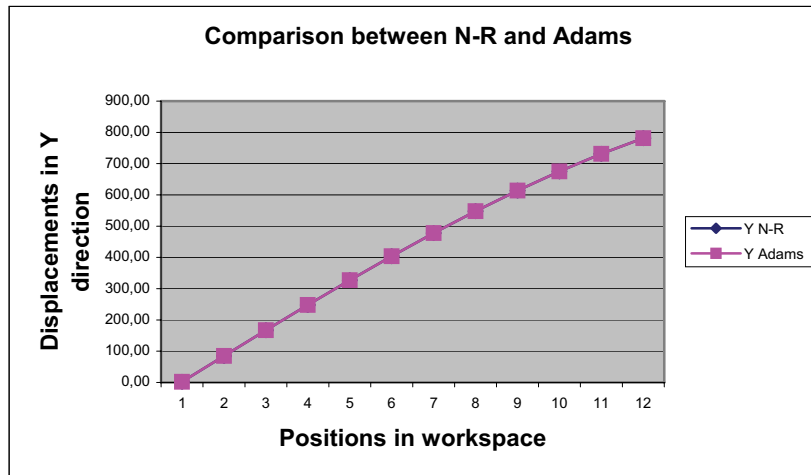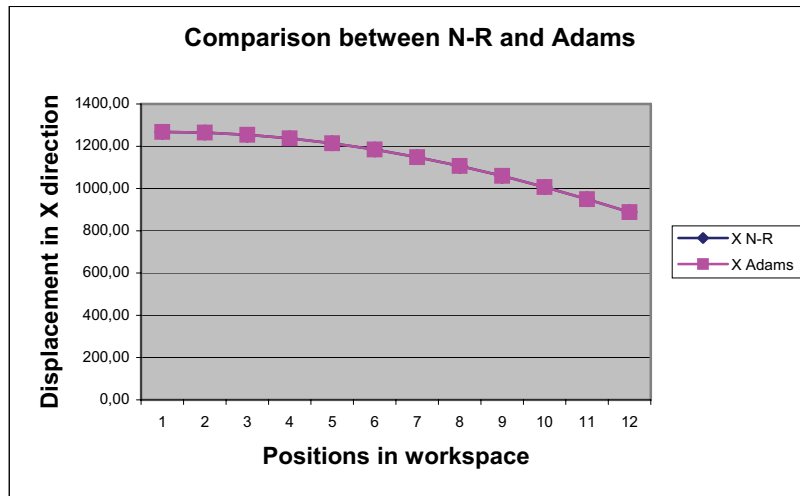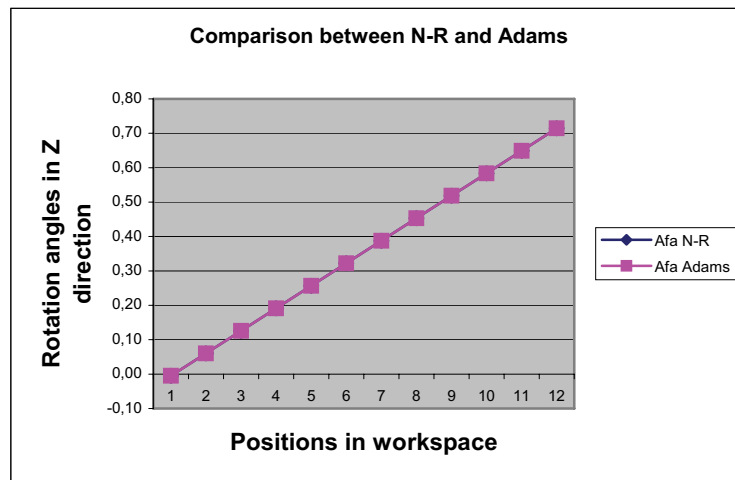
Figure 31. Results of N-R and ADAMS (Input Error Δ Link11=1mm)
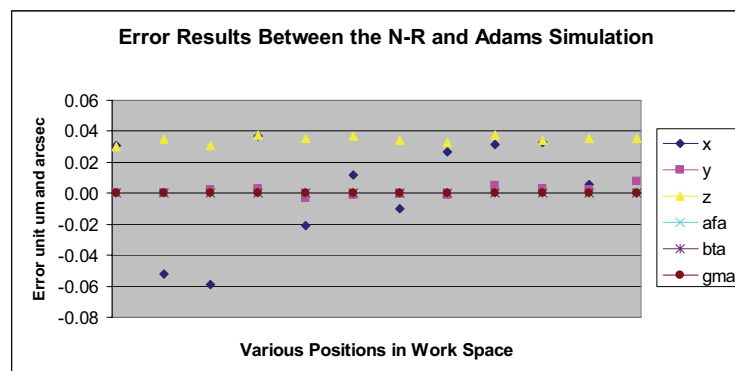
Figure 32. TCP Difference between ADAMS Simulation and N-R Method

6.2 Validation of Jacobian Approximation Method, Error Budget and Calibration

The Jacobian approximation method is verified by the following two different approaches:

    (1) 6-DOF forward kinematic analysis (Newton-Raphson method), and
    (2) ADAMS simulation results.

Based on the D-H model of TAU with all error parameters, inverse and forward kinematic models have been established. From the point of view of mathematics, the TAU kinematic problem is to solve 6 nonlinear equations using Newton-Raphson method with Jacobian matrix as the searching direction

and accurate results have been obtained up to 0.06 um compared with ADAMS simulation results.

It can be observed from the Figure 34, for data in detail, see Table 6, the JAM (Jacobian Approximation Method) is effective with an accuracy of 1.53 μm with an input error of 1 mm (Link 1 of lower arm 1). This was verified using ADAMS simulation results. Results from N-R method match very well with ADAMS simulation with a difference of only 0.06 μm.

The JAM can be used in on-line control and position compensation of the robot. For the TAU robot, a closed form solution of a forward kinematics problem is reached with a high accuracy instead of N-R numerical solution. The simulation results are almost perfect compared with that from ADAMS.

A series of results have been presented for error analysis. Figure 34 shows the results of SVD calibration. Which indicates the number of independent design variable is reduced from 71 to 31. A sudden drop can be observed from the Figure 34, which indicates other parameters behind variable #31 are not necessary and their effects on error model can be neglected. From Table 7, totally 40 redundant variables are removed also Table 10 gives the result of error budge.

Tables 8 and 9 give the actuator (driving motor) error and thermal error, which indicate the change of temperature should be controlled within $\pm 5^0$C otherwise the accuracy of system can not be reached to 50um. The resolution of drive motor should be at least < 10 arc second (1arc second=1/3600 degree).

SVD calibration is carried out for three parameters that contribute to the final position error, see Table 11 and 12. These parameters are Arm3 length, link13 length, and link12 length. Calibration process is completed for only1 iteration. Based on the Table 12 the accuracy of calibration is 4um for Link12 and others are below 1um, which indicates the calibration method and error model are correct.
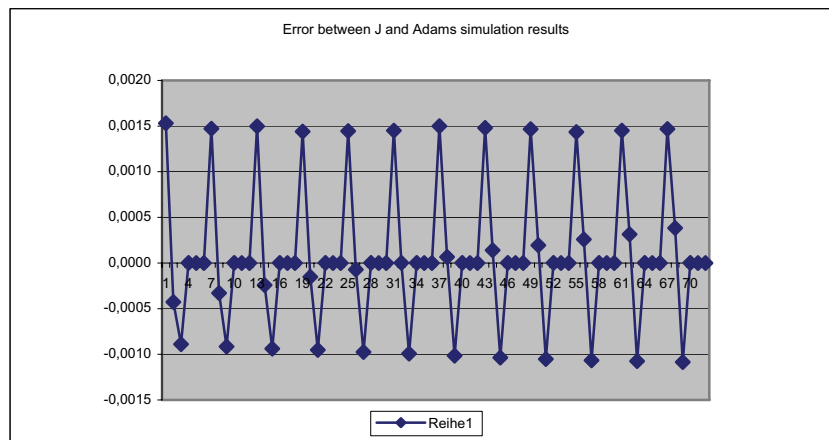
Figure 33. Position Error between Jacobian Approximation Method and ADAMS
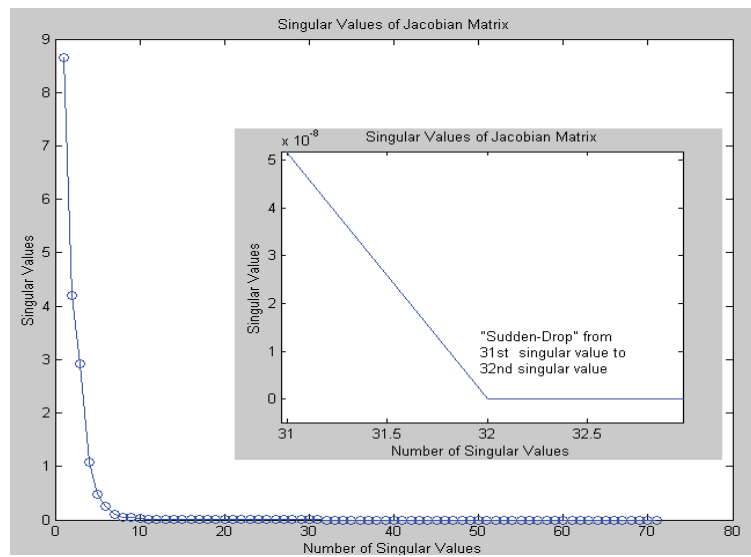
Figure 34. SVD Calibration of TAU Robot

Drive Angles	TCP Pose	Jacobian	Newton_raphson	Error between J and N
joint1=0 joint2=0 joint3=0	X	0,00E+00	1,53E-03	0,001531339
	Y	-1,81E+00	-1,81E+00	-0,0049559
	Z	-1,61E-16	-9,20E-04	-0,000919889
	afa	5,01E-03	5,01E-03	2,634E-07
	bta	-9,32E-19	-9,33E-19	-1,00679E-21
	gma	-9,32E-19	-9,32E-19	-1,5976E-22
joint1=3.75 joint2=3.75 joint3=2	X	1,19E-01	1,20E-01	0,00119916
	Y	-1,81E+00	-1,81E+00	-0,0009736
	Z	-2,09E-16	-9,45E-04	-0,000945048
	afa	5,01E-03	5,01E-03	2,7566E-06
	bta	0,00E+00	9,46E-16	9,45683E-16
	gma	0,00E+00	-4,84E-16	-4,84153E-16
joint1=7.5 joint2=7.5 joint3=4	X	2,37E-01	2,38E-01	0,00135537
	Y	-1,80E+00	-1,80E+00	0,0007562
	Z	-1,79E-16	-9,69E-04	-0,000968876
	afa	5,02E-03	5,02E-03	3,547E-07
	bta	0,00E+00	3,15E-16	3,14853E-16
	gma	0,00E+00	-4,82E-16	-4,82129E-16
joint1=11.25 joint2=11.25 joint3=6	X	3,54E-01	3,55E-01	0,00149511
	Y	-1,78E+00	-1,78E+00	0,0001837
	Z	-1,79E-16	-9,91E-04	-0,000991397
	afa	5,03E-03	5,03E-03	3,263E-06
	bta	0,00E+00	-3,10E-18	-3,10077E-18
	gma	-9,32E-19	1,15E-18	2,0782E-18
joint1=15 joint2=15 joint3=8	X	4,70E-01	4,71E-01	0,00111796
	Y	-1,75E+00	-1,75E+00	-0,0027737
	Z	-5,96E-17	-1,01E-03	-0,001012624
	afa	5,05E-03	5,05E-03	1,7286E-06
	bta	0,00E+00	0,00E+00	0
	gma	0,00E+00	0,00E+00	0
joint1=18.75 joint2=18.75 joint3=10	X	5,83E-01	5,85E-01	0,00173003
	Y	-1,72E+00	-1,72E+00	0,0017688
	Z	-5,96E-17	-1,03E-03	-0,001032565
	afa	5,07E-03	5,08E-03	6,0465E-06
	bta	4,66E-19	-6,39E-16	-6,39425E-16
	gma	-9,32E-19	9,59E-16	9,6015E-16
joint1=22.5 joint2=22.5 joint3=12	X	6,94E-01	6,96E-01	0,00184612
	Y	-1,68E+00	-1,68E+00	0,0036642
	Z	2,09E-16	-1,05E-03	-0,00105122
	afa	5,11E-03	5,11E-03	-3,4323E-06
	bta	0,00E+00	-8,47E-22	-8,47033E-22
	gma	0,00E+00	8,47E-22	8,47033E-22
joint1=26.25 joint2=26.25 joint3=14	X	8,03E-01	8,04E-01	0,00099179
	Y	-1,63E+00	-1,63E+00	0,002734
	Z	0,00E+00	-1,07E-03	-0,001068582
	afa	5,14E-03	5,14E-03	3,7091E-06
	bta	0,00E+00	3,26E-16	3,25672E-16
	gma	0,00E+00	-4,78E-16	-4,77901E-16
joint1=30 joint2=30 joint3=16	X	9,07E-01	9,09E-01	0,00170544
	Y	-1,57E+00	-1,57E+00	-0,0012306
	Z	-2,09E-16	-1,08E-03	-0,001084643
	afa	5,19E-03	5,19E-03	-2,0346E-06
	bta	0,00E+00	8,47E-22	8,47033E-22
	gma	0,00E+00	0,00E+00	0
joint1=33.75 joint2=33.75 joint3=18	X	1,01E+00	1,01E+00	-0,0004597
	Y	-1,51E+00	-1,51E+00	0,0015319
	Z	1,49E-16	-1,10E-03	-0,001099391
	afa	5,24E-03	5,24E-03	-7,54E-08
	bta	0,00E+00	-6,75E-16	-6,74923E-16
	gma	0,00E+00	4,55E-18	4,54772E-18
joint1=37.5 joint2=37.5 joint3=18	X	1,10E+00	1,11E+00	0,0060663
	Y	-1,44E+00	-1,44E+00	0,0007547
	Z	2,98E-17	-1,11E-03	-0,001112819
	afa	5,30E-03	5,30E-03	2,869E-07
	bta	0,00E+00	0,00E+00	0
	gma	0,00E+00	0,00E+00	0
joint1=41.25 joint2=41.25 joint3=22	X	1,20E+00	1,20E+00	-0,002128
	Y	-1,36E+00	-1,36E+00	-0,0038563
	Z	-2,98E-17	-1,12E-03	-0,001124931
	afa	5,37E-03	5,37E-03	-1,1E-07
	bta	0,00E+00	0,00E+00	0
	gma	0,00E+00	0,00E+00	0

Table 6. Comparison between the Results of JAM and N-R Method

Parameter Number	Parameter Definition	Parameter
16	height of the TCP	a
22	joint 3	a6
23	arm3	a7
24	joint 1 & arm 1	d1
25	short arm 1	d3
28	joint3	d6
31	joint_link11_arm1	y1
34	joint_link21_arm1	y2
37	joint_link31_arm1	y3
40	joint_link12_arm2	y4
43	joint_link22_arm2	y5
46	joint_link13_arm3	y6
48	joint_link11p	x11
49	joint_link11p	y11
51	joint_link31p	x22
52	joint_link31p	y22
54	joint_link21p	x33
55	joint_link21p	y33
56	joint_link21p	z33
57	joint_link12p	x44
58	joint_link12p	y44
59	joint_link12p	z44
60	joint_link22p	x55
61	joint_link22p	y55
62	joint_link22p	z55
63	joint_link13p	x66
64	joint_link13p	y66
67	link11	L1
68	link31	L2
69	link21	L3
70	link22	L4

Table 7. List of the Independent Design Variables

Actuator Error			X=1731mm	Y=0 mm	Z=1125mm
$\Delta\theta_1$	$\Delta\theta_2$	$\Delta\theta_3$	ΔX	ΔY	ΔZ
0	0	0	0	0	0
+/-100 arcsec	0	0	-0.1154 0.1149	0.2126 -0.2126	-0.7599 0.7598
0	+/-100 arcsec	0	0.3677 -0.3678	0.1605 -0.1605	0.2435 -0.2433
0	0	+/-100 arcsec	0 0	0.8392 -0.8392	0 0
+/-100 arcsec	+/-100 arcsec	0	0.2524 -0.2528	0.3732 -0.370	-0.5165 0.5164
0	+/-100 arcsec	+/-100 arcsec	0.3675 -0.3681	0.9999 -0.9995	0.2435 0.2433
+/-100 arcsec	+/-100 arcsec	+/-100 arcsec	0.2520 -0.2531	1.2125 -1.2121	-0.5165 0.5164

Table 8. Actuator Error

Temperature ΔT	X=1731mm ΔX	Y=0 mm ΔY	Z=1125mm ΔZ
+/-1°	0.0021	-0.0036	0.0121
	-0.0021	0.0036	0.0121
+/-3°	0.0063	-0.0107	0.0362
	-0.0063	0.0107	-0.0362
+/-5°	0.0106	-0.0178	0.0603
	-0.0106	0.0178	-0.0603

Table 9. Thermal Error

Error Budget			
Variable No.	Description	Name	Budget
1	drive 1	Joint 1	32 arcsec
2	drive 2	Joint 2	ar6 arcsec
3	drive 3	Joint 3	1.2 arcsec
17	joint 1 and arm 1	a1	1.62 um
24		d1	363 um
4		sit1	10.4 arcsec
10		afa1	110 arcsec
18	joint_link11_arm 1	a2	373 um
19	short arm 1	a3	174 um
25		d3	449 um
5		sit3	9.24 arcsec
11		afa3	9.45 arcsec
20	joint 2 and arm 2	a4	1.9 mm
26		d4	485 um
6		sit4	1.22 arcsec
12		afa4	38.5 arcsec
21	short arm 2	a5	430 um
27		d5	D
7		sit5	11.2 arcsec
13		afa5	D
22	joint 3	a6	0
28		d6	D
8		sit6	4.64 arcsec
14		afa6	D
23	arm 3	a7	0
29		d7	D
9		sit7	6.14 arcsec
15		afa7	D
30	joint_link11_arm1	x1	D
31		y1	43 um

32		z1	123 um
33		x2	D
34	joint_link21_arm1	y2	49.4 um
35		z2	D
36		x3	115 um
37	joint_link31_arm1	y3	108 um
38		z3	D
39		x4	D
40	joint_link12_arm2	y4	1.28 mm
41		z4	D
42		x5	2.6 mm
43	joint_link22_arm2	y5	68.2 um
44		z5	D
45		x6	D
46	joint_link13_arm3	y6	21.6 um
47		z6	213 um
48		x11	50 um
49	joint_link11_platform	y11	50 um
50		z11	D
51		x22	50 um
52	joint_link31_platform	y22	50 um
53		z22	D
54		x33	50 um
55	joint_link21_platform	y33	50 um
56		z33	13.3 um
57		x44	50 um
58	joint_link12_platform	y44	50 um
59		z44	37.9 um
60		x55	50 um
61	joint_link22_platform	y55	50 um
62		z55	398 um
63		x66	50 um
64	joint-link13_platform	y66	50 um
65		z66	50 um
16	height of the TCP	a	436 um
66	link 13	L0	0
67	link 11	L1	88 um
68	link 31	L2	151 um
69	link 21	L3	54.3 um
70	link 22	L4	213 um
71	link 12	L5	1.47 mm

Table 10. Error Budget (Assigned System Error = 50 um)

Parameter Number	Parameter Name	Measured Parameter Errors	Calibration Results with SVD (mm/deg)
1	theta1	3	-0.731778837
2	theta2		2.934613648
3	L1	0.246	-0.065823708
4	alpha1	/	0.005595871
5	beta1	/	0.009767543
6	a20	0.639	-0.600433798
7	alpha20	0.004297187	-0.054380834
8	L2	0.022	-0.652730647
9	alpha3	/	0.100085204
10	Rx	NA	0.237556976
11	Ry	NA	-0.297084061
12	Rz	NA	86.49124257
13	tx	NA	-61.06910063
14	ty	NA	-1934.277556
15	tz	NA	510.5174107
16	xpl	NA	22.96695136
17	ypl	NA	-56.41477281

	SVD	LM - Nonlinear optimization	Gauss Newton - Nonlinear optimization
Average Absolute Accuracy (mm)	0.11718325	0.11395309	0.11395309
Average Standard Deviation (MM)	0.04774522	0.04849159	0.04849159
Elapsed Time (s)	300	175	175

Table 11. Calibration Results of 2D Testing Bench

Error Parameter	Error Assigned (mm)	Error from Calibration (mm)
Arm3	0.02 0.05	0.019969 0.049904
Link13	0.01 0.02	0.012201 0.024424
Link12	0.02 0.05	0.018469 0.046093

Table 12. Calibration Result of TAU Robot

6.3 Approach Comparison and Summary

The results discussed above indicate that a closed form forward kinematic solution can be computed and finished much faster than the conventional iterative algorithms. The closed form solution is very difficult to obtain because the problem is highly nonlinear. The N-R method (iterative method) can give an accurate result but it usually takes an average of 4290 multiplications and 630 sine functions for the iterative N-R algorithm. The Polynomial Based method needs at least to solve a 16th-order polynomial equation, which is slow and solution is with spurious roots.

The proposed JAM algorithm can eliminate these drawbacks, and it has an effective closed-form solution with an accuracy of 1.53um.

Table 13 below summarizes the features of the methods proposed by the author in solving the parallel robotics problems involved. The methodology and approach are also used in other robotics applications to effectively increase system modeling, control and process accuracy.

Approaches	Description	Drawback	Accuracy
Polynomial Based	Reduces the resulting constraint equations into a high-order polynomial by the method of elimination.	Requires extremely complicated formulation procedures and has been known to be much slower than the numerical iteration such as the N-R method.	**60-70 um** Lee, H. S. and Han, M. 1999 IEEE
Newton-Raphson (N-R) Numerical Iteration	Among several iterative methods, it has been wisely employed due to its property of convergence.	Jacobian matrix obtained **numerically**, which is not efficient, and has a great influence on the convergence of numerical method.	**0.06 um** Researched in this paper
Extra-Sensor	Reduces the numbers of unknown variable by extra-sensor	Same as the polynomial-based method. Complicated hardware setup	**750 um** Geng, Z. and Haynes, L. 1994
Jacobian Approximation Method (JAM)	Analytic solution of Jacobian matrix + N-R method (one iteration)	**Analytical** Jacobian matrix is difficult to obtain for large-scale MIMO non-linear system.	**1.6 um** Closed-form solution Proposed in this paper

Table 13. Approaches in Parallel Robot Forward Kinematic Modeling

7. Conclusions

The TAU robot represents a new configuration of parallel robots. This robotic configuration is well adapted to perform with a high precision and high stiffness within a large working space compared with a serial robot. It has the advantages of both parallel robots and serial robots.

In this study, the kinematic modeling and error modeling are established with all errors considered using Jacobian matrix method for the robot. Meanwhile, a very effective Jacobian Approximation Method is introduced to calculate the forward kinematic problem instead of Newton-Raphson iteration method. It denotes that a closed form solution can be obtained instead of a numerical iteration solution. A full size Jacobian matrix is used in carrying out error analysis, error budget, and model parameter estimation and identification. Simulation results indicate that both Jacobian matrix and Jacobian Approximation Method are correct and with a level of accuracy of micron meters. ADAMS's simulation results are used in verifying the established models. Experimental results obtained based on both the lab prototype and industrial prototype show that the established models enabled the realization of high precision for the new class of robots.

The established models are also used in the development of other precision robotics systems. Precision robotic machining processes using existing serial robots have been realized successfully with industry partners involved. These precision processes include robotic milling of aluminum engine blocks, and belt grinding of complicated parts of curved surfaces such as engine blades, and human knee joint replacements.

Based on the analytical Jacobian matrix solution, SVD calibration is carried out for three parameters that contribute to the final position error, the accuracy of calibration is within 4um for individual components.

In the milling application of engine block, the position compensation procedure is proved, which reduces the error to < 0.1 mm compared with the original error of 0.5 mm.

8. Future work: error minimization and design optimization of tau robot

8.1 Problem Statement

To further increase the accuracy and performance of a robotic system, error minimization and parameter optimization in the design space will be a powerful tool. There are two spaces involved: the error space with numerous error sources, and the design parameter space with numerous robotic parameters associated with the D-H model. The current practice in robotic design optimization is to solve one of the problems, often the later. It is very difficult to solve both parameter problem and the error problem at the same time.

Based on the prior work about the Error modeling and Sensitivity analysis of Jacobian matrix, Position errors can be obtained in X, Y and Z directions as well as Jacobian matrix (error sensitivities). With these parameters for some given error sources, It is important that how to adjust the other error parameters so that the minimum global error can be obtained in whole workspace.

By analyzing error sensitivity results, the sensitivities vary according to different positions in whole workspace, so the optimal results have to satisfy whole workspace. It is a powerful tool for industrial robot design and development that a method capable of optimizing design parameters in two kinds of different optimization spaces through establishing an optimization criteria in two different independent and relative design and configuration (movement) spaces.

8.2 Problem Formulation – Proposed Object Function and Constrain Function

For Tau robot, the global error function is not only the function of component sizes but also the function of robot positions, and the global error comes from three directions (X, Y and Z) so this multi-objects optimal problem can be transformed into single object problem then combined global error function can be written as object function as follows:

$$F(X_i, \theta_i) = \omega_x F_x + \omega_y F_y + \omega_z F_z$$

(71)

Where θ_i is the position variables of robot, $\omega_x, \omega_y, \omega_z$ are weight factors denoted by designer according to the error budget, F_x, F_y and F_z are errors of X, Y, and Z directions, respectively, and

$$F_x = (C_x + \sum S_{i_x} X_i)^2$$

(72)

Where $F_y = (C_y + \sum S_{iy} X_i)^2$

$$F_z = (C_z + \sum S_{iz} X_i)^2$$

Where C_x, C_y, C_z are given error sources caused by manufacturing and assembly, and S_{ix}, S_{iy}, S_{iz} are just component's sensitivities calculated before in X, Y and Z directions. Here one attention point is that all parameters Ci and Si are functions of position parameters θ_i. The constrain function is $X_{i\min} \leq X_i \leq X_{i\max}$ From above one can transform the constrained optimal problem into unconstrained optimal problem by using the Lagrange multiplier Method as follows:

$$F(X_i, \theta_i) = \omega_x F_x + \omega_y F_y + \omega_z F_z + \sum_{j=1}^{k} \lambda_j g_j(X_i) \tag{73}$$

Where $g_j(X_i)$ is the constrain function.

In constrained optimization, the general aim is to transform the problem into an easier sub-problem that can then be solved and used as the basis of an iterative process. A characteristic of a large class of early methods is the translation of the constrained problem to a basic unconstrained problem by using a penalty function for constraints, which are near or beyond the constraint boundary. In this way the constrained problem is solved using a sequence of parameterized unconstrained optimizations, which in the limit (of the sequence) converge to the constrained problem. These methods are now considered relatively inefficient and have been replaced by methods that have focused on the solution of the Kuhn-Tucker (KT) equations. The KT equations are necessary conditions for optimality for a constrained optimization problem. If the problem is a so-called convex programming problem, then the KT equations are both necessary and sufficient for a global solution.
So from the Equation (73) next Kuhn-Tucker conditions should be satisfied as:

$$\nabla F(X^*, \theta^*) + \sum_{j=1}^{k} \lambda_j \nabla g_j(X^*) = 0$$

$$\lambda_j g_j(X^*) = 0 \qquad\qquad (j = 1, 2, ..., k) \tag{74}$$

$$\lambda_j \geq 0 \qquad\qquad (j = 1, 2, ... k)$$

For a given configuration, by using Sequential Quadratic Programming (SQP) method, Equation (74) can be solved.

8.3 Future Work - Optimization in the Whole Robotics Workspace

 Since from mentioned above the object functions have to satisfy whole work-space. A most serious configuration (movement position) has to be found so that one can optimize the object function in this situation.

  One needs to utilize the other method to optimize the object function since the relationship between the object function and position variable is not ex-plicit. Here "Pattern Search method" is adopted to search the most serious po-sition of robot that means to maximum the object function. The final optimal result can be obtained by using the Lagrange multiplier Method as the most serious position is known.

"Pattern Search method" is consisted of two steps 'move', one is exploratory move and other is module move. The former is to obtain the useful direction by calculating the variations of object function the later is to get a better new "point" instead of old "point" in the useful direction, which is similar to the gradient direction.

Description of Exploratory Move and Module Move:

Given initial point X$^{(0)}$, step length $\bar{\alpha} = (\bar{\alpha}_1, \bar{\alpha}_2, ..., \bar{\alpha}_n)^T$, $0 < \beta < 1$, error ε.

1. $0 \Rightarrow K$
2. Exploratory Move, $\bar{\alpha} \Rightarrow \alpha$
    2.1 $0 \Rightarrow i$, $X^{(k)} \Rightarrow \hat{X}^{(k,i)}$
    2.2 $\hat{X}^{(k,i)} + \alpha_{i+1}e_{i+1} \Rightarrow \widetilde{X}$, $\hat{X}^{(k,i)} - \alpha_{i+1}e_{i+1} \Rightarrow \overline{X}$, $\hat{X}^{(k,i)} \Rightarrow X$
    2.3   if $f(\widetilde{X}) < f(X)$,   then $\widetilde{X} \Rightarrow \hat{X}^{(k,i+1)}$;
          if $f(\widetilde{X}) \geq f(X) > f(\overline{X})$,   then $\overline{X} \Rightarrow \hat{X}^{(k,i+1)}$;
          if $f(\widetilde{X}) \leq f(X) \leq f(\overline{X})$,   then $X \Rightarrow \hat{X}^{(k,i+1)}$
    2.4 $i+1 \Rightarrow i$
    2.5   if $i < n$,   then, do   2.2;   if $i = n$,   do   3
3. if $\hat{X}^{(k,n)} \neq \hat{X}^{(k,0)}$,   then $\hat{X}^{(k,n)} \Rightarrow \hat{X}^{(k+1)}$,   do 5;
    if $\hat{X}^{(k,n)} = \hat{X}^{(k,0)}$,   then, do 4.
4. if $\|\alpha\| \leq \varepsilon$,   then solve the optimal solutions, X$^* = \hat{X}^{(k,n)}$. if $\|\alpha\| > \varepsilon$ and $\beta\alpha \Rightarrow \alpha$, go to 2.1
5. $2\hat{X}^{(k+1)} - X^{(k)} \Rightarrow \hat{\hat{X}}^{(k+1)}$, obtain $\hat{y}^{(k+1)}$ from $\hat{X}$ by exploratory move.
6. if $f(\hat{y}^{(k+1)}) < f(\hat{X}^{(k+1)})$,   then $\hat{y}^{(k+1)} \Rightarrow X^{(k+1)}, k+1 \Rightarrow k$, go to 2;
    if $f(\hat{y}^{(k+1)}) \geq f(\hat{X}^{(k+1)})$,   then $\hat{X}^{(k+1)} \Rightarrow X^{(k+1)}, k+1 \Rightarrow k$,
    $\beta\alpha \Rightarrow \alpha$ go to 2.1

So the calculation stops at the step 4.

Module move (Hooke-Jeeves) Flow Chart
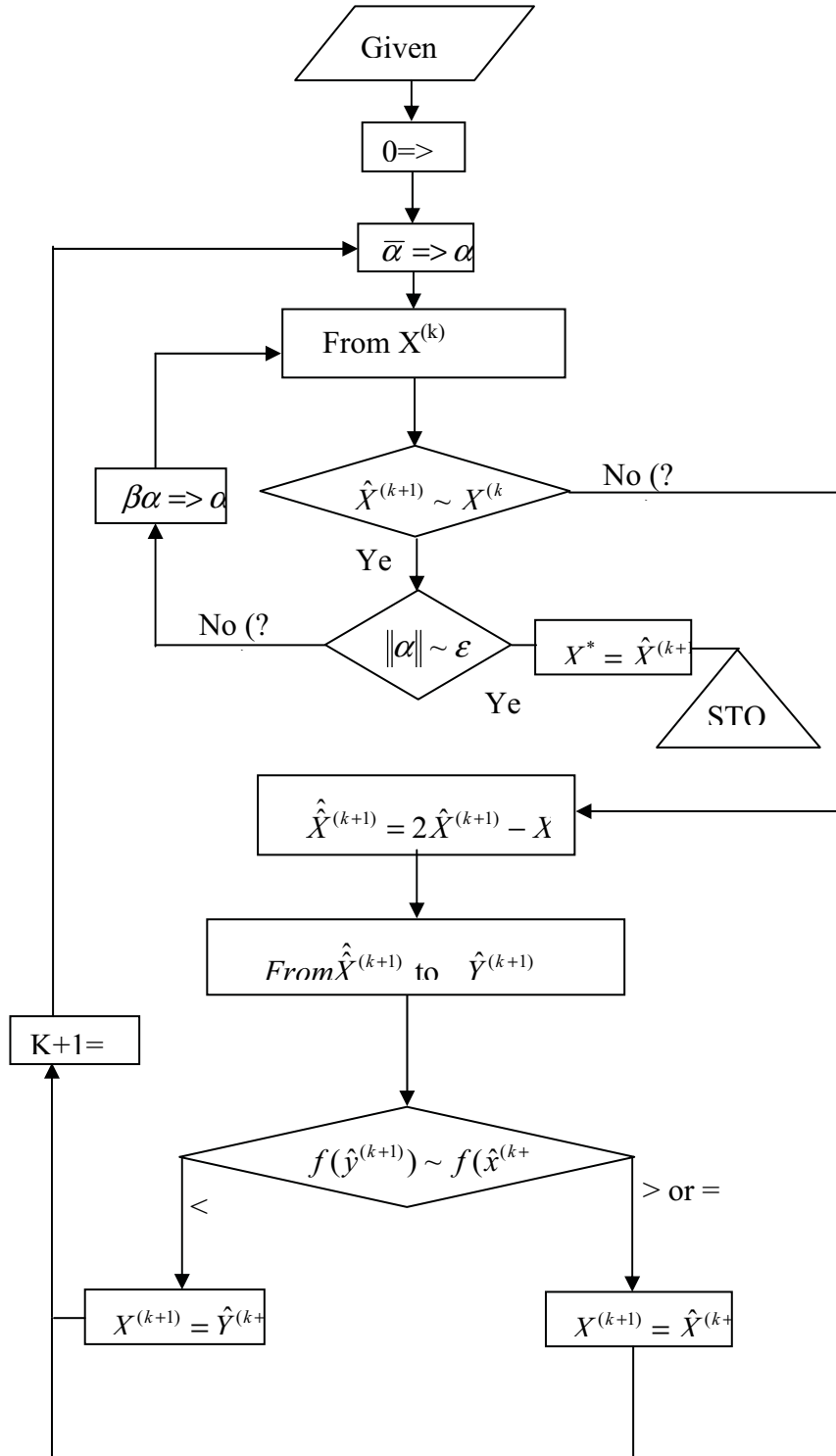
9. References

Abderrahim, M. and Whittake, A. R., "Kinematic Model Identification of Industrial manipulators," Robotics and Computer Integrated Manufacturing 16 (2000), 1-8

Brogangrdh, T., "Design of high performance parallel arm robots for industrial applications," Proceedings of A symposium Commemorating the Legacy, Works, and Life of Sir Robert Stawell Ball Upon the 100th Anniversary of A Treatise on the Theory of Screws , University of Cambridge, Trinity College, July 9-11, 2000

Brogangrdh, T., et al, "Device for relative movement of two elements," United States Patent 6425303, July 30, 2002

Brogangrdh, T., et al, "Device for relative movement of two elements," United States Patent 6336374, January 8, 2002

Brogangrdh, T., et al, "Device for relative movement of two elements," United States Patent 6301988, October 16, 2001

Cui, H., Zhu, Z., Gan Z., and Brogardh, T., "Kinematics Analysis and Error Modeling of Tau Parallel Robot", Robotics and Computer Integrated Manufacturing: An International Journal, v 21, n 6, December, 2005, p 497-505

Denavit, J., Hartenberg, H., 1955, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", Journal of Applied Mechanics, pp. 215- 221

Dhingra, A. K., Almadi, A. N. and Kohli, D., "Closed-Form Displacement Analysis of 8, 9 and 10-Link Mechanisms, Part II" Mechanism and Machine Theory, 35, 2000, pp 851-869

Dhingra, A. K., Almadi,A. N. and Kohli, D., "A Grobner-Sylvester Hybrid Method For Closed-Form Displacement Analysis of Mechanisms," 1998 ASME Design Engineering Technical Conference, Atlanta, GA

Dhingra, A. K., Almadi,A. N. and Kohli, D., "Closed-Form Displacement Analysis of 8, 9 and 10-Link Mechanisms," Mechanism and Machine Theory, 35, 2000, pp 821-850

Didrit, Olivier; Petitot, Michel & Walter, Eric "Guaranteed Solution of Direct Kinematic Problems for General Configurations of Parallel Manipulators," IEEE Transactions on Robotics and Automation, Vol. 14, No.2, April 1998

Gong, Chunhe, Yuan, Jingxia, and Ni, J., "Nongeometric Error Identification and Compensation for Robotic System by Inverse Calibration," International Journal of Machine Tools & Manufacture 40 (2000) 2119-2137

Griffis,M. & Duffy, J. "A Forward Displacement Analysis of a Class of Stewart Platform," Journal of Robotic System 6 (6), 703-720 (1989) by John Wiley & Sons, Inc

Lee, hyung Sang and Myung-chul Han, "The estimation for forward kinematic solution of stewart platform using the neural network", Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems

Lin,W.; Griffis, M.; & Duffy, J. "Forward Displacement Analyses of the 4-4 Stewart Platforms," Transaction of the ASME Vol. 114, September 1992, pp444-450

Patel, Amit J., Ehmann, K. F., "Calibration of a Hexapod Machine Tool Using a Redundant Leg," International Journal of Machine Tools & Manufacture 40 2000, 489-512

Prabjot Nanua, Kenneth J. Waldron, and Vasudeva Murthy, "Direct kinematic Solution of a Stewart platform," IEEE Transactions on Robotics and Automation, Vol. 6, No.4, August 1990

Raghavan, M., "The Stewart Platform of General Geometry Has 40 Configurations," ASME Journal of Mechanical Design, June 1993, Vol. 115, pp 277-282

Shi, Xiaolun; Fenton,R. G. "A Complete and General Solution to the Forward Kinematics Problem of Platform-Type Robotic Manipulators," IEEE, 1050-4729/ 1994, pp 3055-3062

Sreenivasan,S. V.; Waldron K. J. & Nanua,P. "Closed-Form Direct Displacement Analysis of a 6-6 Stewart Platform," Mech. Mach. Theory Vol. 29. No. 6, pp 855-864, 1994

Stewart, D., 1965, "A Platform with Six Degree-of-freedom", Proc, 1st. Mech. Eng. London, Volume 180, pp371-286

Tsai, L., 1999 "Robot Analysis-The Mechanics of Serial and Parallel Manipulators", John Wiley & Sons, Inc, ISBN: 0-471-32593-7

Wang, J. & Masory, O. "On the Accuracy of A Stewart Platform- Part I The Effect of Manufacturing Tolerances," IEEE, 1050-4729/ 1993, pp 114-120

Z. Jason Geng and Leonard S. Haynes, "A 3-2-1 Kinematic Configuration of a Stewart Platform and its Application to Six Degree of Freedom Pose Measurements," Robotics & Computer-Integrated Manufacture, Vol. 11, No. 1, pp23-34, 1994

Zhang, H., Wang, J., Zhang, G., Gan, Z., Pan, Z., Cui, H. and Zhu, Z. "Machining with flexible manipulator: Toward improving robotic machining performance", Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM 2005, 2005, p 1127-1132

Zhang,Chang-de, Song, Shin-Min, "Forward Kinematics of a Class of Parallel (Stewart) Platform with Closed-Form Solutions," Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, California-April 1991

Zhu, Z.; Cui, H. "Six degree of freedom measuring system", Apr., 2003, U.S. patent (7040033)

Networking Multiple Robots for Cooperative Manipulation

M. Moallem

1. Introduction

In this chapter, the development of an open architecture multi-robot system is studied. The environment consists of five serial-link robot manipulators operated using embedded control computers. The robot control computers are connected together through a network of supervisory computers. A preemptive multi-tasking Real Time Operating System (RTOS) running on the supervisory computers is used to perform supervisory and cooperative tasks involving multiple robots. The software environment allows for controlling the motion of one or more robots and their interaction with other devices. Development of modular components is discussed in this chapter along with typical laboratory procedures. The environment can be used to develop software for various robotic applications such as scheduling robotic tasks, cooperative manipulation, collision avoidance, internet-based telerobotics, and other networked robotic applications.

2. Overview of Networked Multi-robot Systems

With the advent of new computing, sensor, and actuator technologies, the application of robotic systems has been growing rapidly in the past decade. Robotic systems were originally developed due to their capability to increase productivity and operate in hazardous environments. In recent years, robotics has found its way to a completely new range of real-world applications such as training, manufacturing, surgery, and health care (Bernard et al., 1999; Craig, 1997; Goldberg et al., 2000; Taylor and Stoianovici, 2003). From the advanced manufacturing domain to daily life applications, Internet-based telerobotic systems have the potential to provide significant benefits in terms of telepresence, wider reachability, cost effectiveness and maximal resource utilization. Challenging problems with regard to Internet-based telerobotic systems include such issues as uncertain Internet time delays (Luo and Chen, 2000), system reliability, interaction capability (Schulz al., 2000), and augmented Human-Machine interfaces. Due to the emergence of new areas in the field of

robotics, there is a growing need for applications that go beyond classical ones such as simple pick-and-place operations involving a single robot.

Many conventional robot manipulators are *closed architecture,* meaning that the user does not have direct access robot's sensory data and actuator inputs. To operate a robot, the user is usually confined to a *Robot Programming Language* (RPL) that is specific to the robotic system being used (Craig, 1997). This is restrictive in many cases, including the robotic applications requiring coordination of such robots. For example, in developing robotic work-cells that require interaction of two or more robots at a time, there is a growing need for robots to share and exchange information through a network. Use of an RPL is restrictive in such cases due to the limited capability of the programming environment. In this work we present a laboratory setup that can be utilized in order to perform tasks requiring multi-robot scheduling and cooperation tasks using industry grade manipulators. The objective is to create a flexible software environment so that the robot programmer can perform robotic tasks using a programming language such as C/C++ and a real-time operating system.

3. Interconnection of Multiple Robotic Systems

In this section an overview of a multiple robotic system is presented. The setup consists of stand-alone robotic systems which are interconnected through a computer network to be used in cooperative applications.

3.1 Hardware and Software Configuration

Figure 1 illustrates a multiple robotic system comprised of three 6 degree-of-freedom (DoF) and two 7-DoF robots, all from *CRS, Inc.,* located at the Robotics Laboratory, University of Western Ontario, Canada. The two 7-DoF robots are mounted on movable tracks while the other three 6-DoF robots are mounted on stationary bases. Each robot is equipped with a gripper controlled by a servo motor and a 6-dof force/torque sensor. The computing environment is comprised of a host-target architecture. The target machines consist of Pentium computers running under the *VxWorks* real-time operating system from WindRiver Systems, Inc (www.wrs.com). The host machines are used for system monitoring and development tasks and run under the Solaris or Microsoft operating systems.

3.2 Networking and Communication Configuration

The local networking and communication platform utilizes two types of mechanisms as shown in Figure 2, consisting of *serial port* which connect target

machines to robot controllers, and an *Ethernet network*, which links together all the target and host machines. Many commercial robots use serial communication between the host computers and the robot controllers. In this setup, we use the RS232 serial lines to transmit control commands and sensory information between the target machines and robot controllers. The robot motion commands are issued by a target machine and are sent to the robot controllers. The robot controllers also transmit sensory information such as gripper distances, force sensor readings, and the status of executed commands, back to the target machines. Similarly, the target machines transmit sensory and command data through the network to other machines. The robot controllers are embedded computer systems without network connectivity and standard operating system support. Lack of network connectivity is a main drawback of many conventional robot controllers. In Figure 2, the three target machines run under the VxWorks real-time operating system. However, any other operating system that supports networking tools and inter-task synchronization and communication primitives such as semaphores, message queues, and signals, can be used to do the same job.
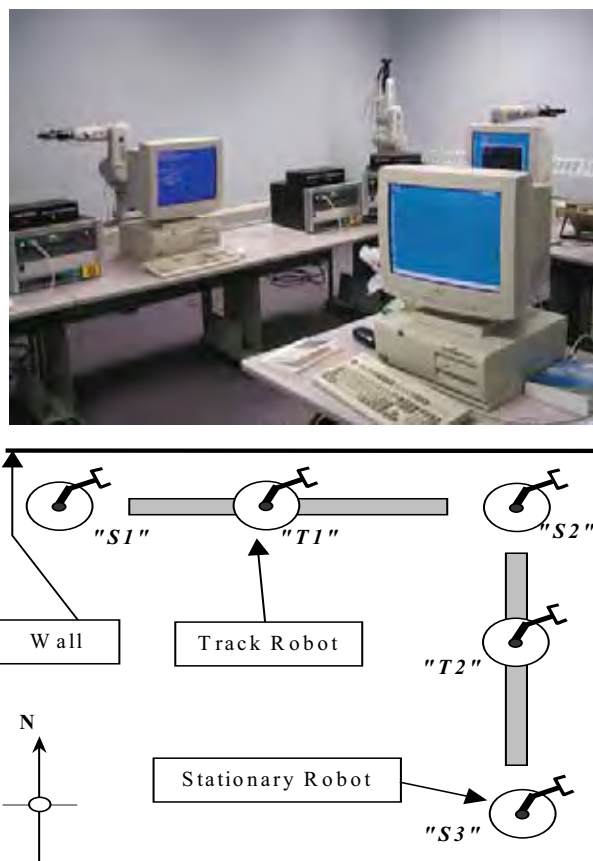
Figure 1. Multi-robot system (left) and layout of the robots (right)
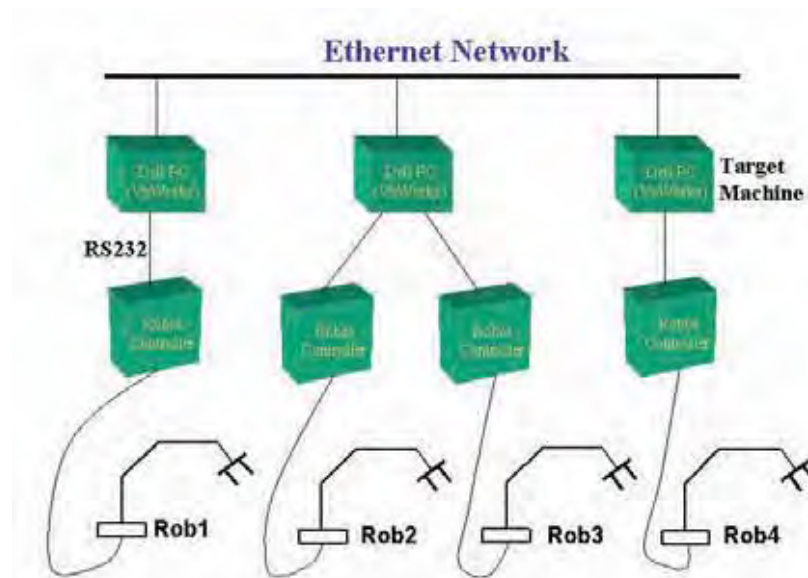
Figure 2. Configuration of the networked robotic system shown in Figure 1

3.3 Use of Real-time Operating Systems

Good practice in software engineering encourages the use of object-oriented programming for developing application software (Pressman, 1997). The main aspects of object-orientated programming are encapsulation, inheritance, modifiability, and reusability. In this regard, robotic systems are no exception. It is desirable to have software modules that can be easily ported to other platforms, to be modifiable, and can be reused for different robotic applications. This is particularly desirable in a laboratory setup where the functional and non-functional requirements of projects can change. Therefore, the availability of certain software modules would make it convenient to develop or modify code for new applications. On the other hand, the computer technology has got so powerful that an operating system can be used to develop and run applications on embedded computers. Nowadays operating systems are found in many devices and systems such as cell phones, wireless access points, robotics, manufacturing, and control applications. Many applications, including robotics, are real-time meaning that the computer must not only perform the calculations and logical operations correctly, but it must also perform them on time. In other words, correctness is as important as timeliness. Moreover, complex operations require modular programming which can be facilitated by using a real-time operating system. The operating system is responsible for operations such as controlling and allocating memory, prioritizing the execution of tasks, controlling input and output devices, networking operations, and managing files. The software developed using operating system facilities can be changed or modified easily without having to scrap the whole program.

4. Application Development for Distributed Robotic Applications

Distributed networked systems are increasingly becoming popular in industry, education, and research (Hung, 2000). Networked systems have the advantage of greater flexibility and better distribution of computing resources when compared to stand alone systems. Different networking architectures and protocols have been used in automation and control systems such as DeviceNet (DeviceNet Vendors Association, 1997), ProfiBus (http://www.profibus.com/), Manufacturing and Automation Protocol (Raji, 1994), ControlNet (ControlNet International, 1988), and Ethernet (see for example, Tanenbaum, 1996). Evaluation of the performance of these networks has been reported in the literature, for example in (Lian, *et al.*, 2001) and (Hung, 2000). The emergence of *networked systems* on the factory floor is driving the automation industry to embrace new network technologies. For improved performance and cost efficiency, robots used on a factory floor should be enabled to provide data related to manufacturing and process operations to the management in real-time and preferably using non-proprietary networks. In the following, an outline of the software framework for supervisory control of the robots depicted in Figures 1 and 2 is presented.

4.1 Application Development under a Real-Time Operating System

Real-time operating systems have emerged in the past decade to become one of the basic building blocks of embedded computer systems, including complex robotic systems. A modular approach to software development for time critical embedded systems calls for decomposition of applications into multiple tasks and use of operating system primitives. A real-time operating system can be used to run on the supervisory computers such as the Pentium computers shown in Figure 2. We have used the *Tornado* development environment which provides a graphical user interface and tools for developing real time multitasking applications under VxWorks (www.wrs.com). However, any other real-time operating system can be used for this purpose. In Figure 2, once the programs are compiled and linked, the tasks can be downloaded into the memory of the PC workstations running *VxWorks*. These computers are used as supervisory controllers that enable communication between robots through the network communication ports.

4.1.1 The Robot Module

The starting point for implementing modular software for robotic applications is representing the robot as a class consisting of private data attributes and member functions as shown in Figure 3.

Robot Class Diagram

Robot class

Private Attributes:
Id[string]   /* robot id */
pos[6]   /* defines the offset of the robot joints from the default */
dpos[6]   /*speed */
ddpos[6] /* acceleration */
tau[3]   /*?? */
force[3]   /*grip force */
torque[3]
gripDist
robotCommand[string]   /* the issued command string */
robotResponse[string]   /* robot response to the command */
serialPort   /* serial port number between the controller and work station */
fd   /*file descriptor for serial communication */

Member Functions:
Robot() /*Constructor */
InitializeRobot() /* called by constructor */
Calrdy()
moveToPoint(point)
SendRobotCommand(string)
openPort()   /* serial communication port */
closePort()

Figure 3. The *Robot Class* attributes and functions

In the class diagram of Figure 3, the robot identification, position, speed, torque, and other variables are defined as the attributes of the robot object. The commands sent to the robots are string variables stored in the robotCommand[ ] array. Similarly, the status received for each robot after executing a command is stored in the robotResponse[ ] array. The communication between the Pentium PCs in Figure 2 with the robot controller is two-way which is performed through the RS-232 serial interface. The serialPort attribute in Figure 3 is used to identify which serial port each robot object is using for communication. The member functions of the Robot Class shown in Figure 3 are used to initialize the robot object using *InitializeRobot()*, move the robot to a calibrated position using *Calrdy()*, move the robot to a particular point using *moveToPoint()*, send a command using *SendRobotCommand(),* and to open or close a serial port using *openPort()* and *closePort()*, respectively. If needed, the above class can be modified or other classes can be inherited from it.

One benefit of modular software development is the convenience of developing the modules one by one. After finishing each part, testing and debugging can be performed on other parts to be implemented. As a result, the final integration and testing can be done without much difficulty. In the following we discuss some of the projects that have been performed using this environment.

4.1.2 Controlling Robots through Serial Ports

Consider the implementation of a cooperative robotic task to be performed by the two robots indicated in Figure 4, in which the PC communicates with robot controllers through the RS-232 serial ports. Note that the tasks on the PC workstation are running concurrently using a real-time operating system (VxWorks in this case).
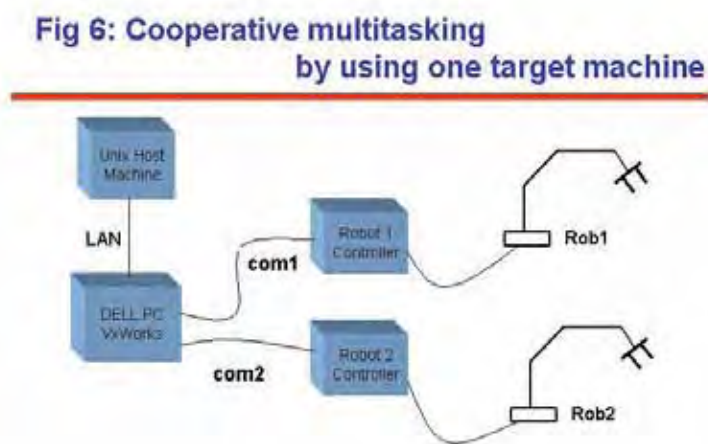
Figure 4. Cooperative multitasking by using one VxWorks station

Before running the system under *VxWorks*, the robot programming language *robcom* is used to send commands to the robot from the application shell, which is similar to MSDOS or UNIX prompts. For example, by issuing the command "*joint 5, 20*", the robot's fifth joint will rotate 20 degrees clockwise. This allows for the commands sent to the robot to be tested at the command prompt level before executing them from a program. The second step involves sending the commands through the serial ports using a high level program running under *VxWorks* instead of the application shell. For example, the function interface *SendRobotCommand()* in Figure 3 is written to send commands through the serial port, or the *moveToPoint()* command is to move the robot to a previously taught positions in the robot's workspace.

4.1.3 Object Handling

In this demonstration, one robot catches an object and passes it to a second robot. The goal is to develop code for coordination of motion of two robots using synchronization mechanisms such as *semaphores* provided by the operating

system. A semaphore is a token which if available, will cause the task to continue and if not available, will block the calling task until the token becomes available. At the beginning of the program, two robot objects are declared and initialized. The first robot is programmed to move and fetch the object from a known pick-up point that has been previously taught to it. Meanwhile, the second robot moves to the delivery position and waits for the first robot to deliver the object. A waiting mechanism is implemented using an empty semaphore by issuing a *semTake()* command in VxWorks which causes the task to block until the semaphore token becomes available. When the first robot has reached the delivery point and is ready to deliver the object, it releases the empty semaphore. The task running the first robot then unblocks, opens its gripper, and the process of transferring the object is completed. When the robot catches the object, it moves toward the release point where it allows the other robot to move to its final destination. At the end, both robots move to their calibration positions.

4.1.4 Network-based Cooperative Control of two Robots

Network communication allows more than two robots to perform cooperative tasks concurrently. In this scenario, socket programming under TCP/IP is used for communication between the *VxWorks* workstations in a client-server configuration. A server socket is assigned a well-known address which is constantly listening for client messages to arrive. A client process sends messages to the server via the server socket's advertised address. The hardware setup for this experiment is shown in Figure 5.
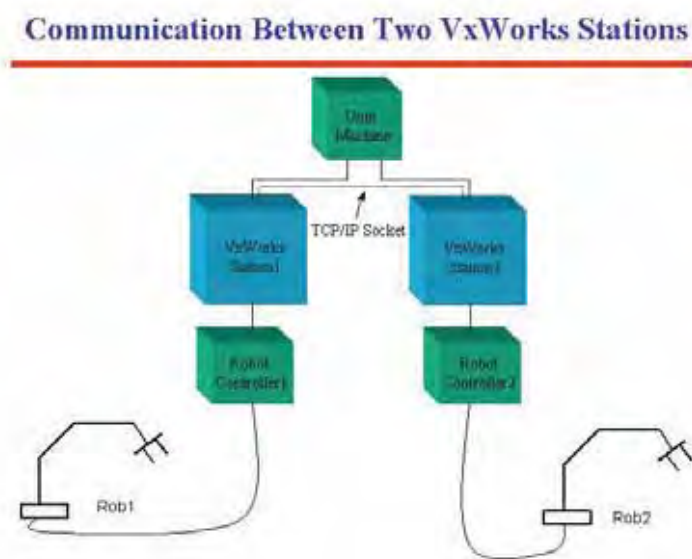
Figure 5. Hardware setup for TCP/IP communication

As indicated in Figure 5, two robots are connected to two different *VxWorks* workstations. The client program is run on one VxWorks workstation and the server program is run on the other station simultaneously. The previous demonstrations can be performed on this configuration too. For example, one robot can catch an object and pass it to a second robot in a similar manner as discussed before but by using the network interface. In this case, a set of special strings containing robot commands are defined on both the client and the server sides. These strings are then transmitted through sockets to perform cooperative tasks. The client-server mechanism can be used for synchronizing tasks with each other. At the beginning of the program, two robot objects are declared and initialized. In the server routine, a TCP socket is initialized, which listens for connection from the client while performing its own task. In the client routine, a socket is initialized and the connection request is sent to the server. After the connection is established, both client and the server can synchronize their operations.

4.1.5 Robot Interaction with Input-Output Modules

There are many situations that robots must coordinate their operations with external devices. For example in a smart manufacturing workcell, robots have to grab parts from devices such as indexers, or deliver parts to them in a synchronized manner. A rotary indexing table is a simple example that simulates part of a manufacturing process. The indexing table can rotate by means of a stepping motor. The motor controller is interfaced with a *VxWorks* station and a digital I/O card. The indexer and robot can be connected to the same computer or to separate computers on a shared network. The situation is similar to the previous example in part E. The program that controls the indexing table operation is spawned as a separate task which can coordinate its operation with other robots, for example by using semaphores or a TCP/IP client-server mechanism on a network.

4.1.6 Other Multi-Robot Operations

Several projects related to scheduling and cooperative operation of robots similar to those used in a manufacturing work-cell have been carried out using the setup described in this chapter. For example, referring to Figure 1, an object handling operation was developed where the first robot takes an object from a person and delivers it to the second robot. Then, the second robot delivers the object to the third robot, and so on, until the object is delivered to the fifth robot.

Another project was related to visualization of a robotic work-cell using the Matlab Virtual Reality Modeling Language (VRML) toolbox, from MathWorks, Inc. In this project, sensory data such as joint displacements are sent through

the network to a host computer that may be located in a control room. A visualization program written in VRML is running on the host computer. This computer obtains real-time sensory data from supervisory robot computers on the network and presents to the operator a visualization of the robotic work-cell. A main advantage of using this scheme over sending data through a camera vision system is the small bandwidth required for sending sensory data, as opposed to a relatively large bandwidth required for transmitting picture frames when a vision system is used.

The environment has also been used in a distributed robotic system with Internet connectivity where robot operations can be monitored and operated from the Internet (Wang *et al.,* 2003). The scheduling of robotic work-cells used in manufacturing has been another topic, where issues such as checking for deadlock situations, or scheduling the operation of multiple robotic systems with different timing requirements have been addressed (Yuan *et al.,* 2004).

5. Conclusion

In this chapter, some aspects of developing modular software for controlling robot operations were discussed. Many commercial robots have a closed architecture, which makes them difficult to program for certain applications involving multiple robots. A networked robotic system offers interesting possibilities in terms of developing novel applications. With the recent advancements made in the networking technologies it is important that students and engineers taking courses or projects in robotics and automation be familiar with the capabilities offered by new technologies.

6. References

Bernard, C., Kang, H., Sigh, S.K. and Wen, J.T. (1999), "Robotic system for collaborative control in minimally invasive surgery", *Industrial Robot: An international Journal*, Vol. 26, No. 6, pp. 476-484.

Craig, C.G. (1989), *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, Boston, MA.

Goldberg, K., Gentner, S., Sutter, C. and Wiegley, J. (2000), "The mercury project: A feasibility study for Internet robots" *IEEE Robotics & Auto. Magazine*, Vol. 7, No.1, pp. 35-40.

Pressman, R. (1997), *Software Engineering: A Practitioner's Approach,* McGraw-Hill, New York, NY.

DeviceNet Vendors Association (1997), *DeviceNet Specifications*, 2nd. ed. Boca Raton, FL.

Raji, R.S. (1994), "Smart Networks for Control," *IEEE Spectrum*, Vol. 31, pp. 49-55, June 1994.

ControlNet International (1988), *ControlNet Specifications*, 2nd ed. Boca Raton, FL.

Tanenbaum, A.S. (1996), *Computer Networks*, 3rd ed. Upper Saddle River, Prentice Hall, NJ.

Lian, F.-L. Moyne, J.R.; Tilbury, D.M. (2001), "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet**,**" *IEEE Control Systems Magazine*, Vol. 25, No. 1, pp. 66-83, 2001.

Hung**,** S.H., (2000), **"**Experimental performance evaluation of Profibus-FMS," *IEEE Robotics & Automation Magazine*, Vol. 7, No. 4, pp. 64-72.

Taylor, R.H. and Stoianovici, D. (2003), "Medical Robotics in Computer Integrated Surgery," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 765–781.

Luo, R.C., and Chen, T.M. (2000), "Development of a Multibehavior-based Mobile Robot for Remote Supervisory Control through the Internet", *IEEE/ASME Trans. on Mechatronics,* Vol.5, No.4, pp. 376-385.

Schulz, D., Burgard, W., Fox, D., Thrun, S. and Cremers, A.B., (2000), "Web Interfaces for Mobile Robots in Public Places", *IEEE Robotics & Auto. Magazine*, Vol. 7, No.1, pp. 48-56.

Wang, X-G., Moallem, M. and Patel, R.V., (2003), "An Internet-Based Distributed Multiple-Telerobot System," *IEEE Transactions on Systems, Man, and Cybernetics, Part A,* Vol. 33, No. 5, pp. 627- 634.

Yuan, P., Moallem, M. and Patel, R.V. (2004), "A Real-Time Task-Oriented Scheduling Algorithm for Distributed Multi-Robot System," *IEEE International Conference on Robotics and Automation*, New Orleans, LA.

Web-Based Remote Manipulation of Parallel Robot in Advanced Manufacturing Systems

Dan Zhang, Lihui Wang and Ebrahim Esmailzadeh

1. Introduction

During the last decade, the Web has gained widespread acceptance in both academic and business areas. The Web is used by many as a medium of sharing data, information, and knowledge. Today, it is widely used for development of collaborative applications to support dispersed working groups and organizations because of its platform, network and operating system transparency, and its easy-to-use user interface – Web browser. In addition to the Web technology, Java has brought about a fundamental change in the way that applications are designed and deployed. Java's "write once, run anywhere" model has reduced the complexity and cost traditionally associated with producing software on multiple distinct hardware platforms. With Java, the browser paradigm has emerged as a compelling way to produce applications for collaboration over the Internet. As business grows increasingly diversified, the potential of this application is huge. Targeting distributed, real-time monitoring and control in manufacturing sectors, a framework with high efficiency for cyber collaboration is carefully examined.

The objective of this research is to develop a Web-based digital shop floor framework called *Wise-ShopFloor* (Web-based integrated sensor-driven e-ShopFloor) for distant shop floor monitoring and control. The *Wise-ShopFloor*, with an appropriate architecture for effective data communication among a dispersed engineering team, can serve real-time data from bottom up and can function as a constituent component of e-manufacturing. The framework is designed to use the popular client-server architecture and VCM (view-control-model) design pattern with secured session control. The proposed solutions for meeting both the user requirements demanding rich data sharing and the real-time constraints are: (1) using interactive Java 3D models instead of bandwidth-consuming camera images for visualization; (2) transmitting only the sensor data and control commands between models and device controllers for monitoring and control; (3) providing users with thin-client graphical interface for navigation; and (4) deploying control logic in an application server. A

proof-of-concept prototype system is developed on top of the framework to demonstrate one of its potential applications on shop floor monitoring and control. It utilizes the latest Java technologies, including Java 3D and Java Servlets, as enabling technologies for system implementation. Instead of camera images, a physical device of interest is represented by a Java 3D model with behavioural control nodes embedded. Once downloaded from an application server, the Java 3D model works on behalf of its counterpart showing behaviours for visualization at a client side, but remains alive by connecting with the physical device through low-volume message passing.

This chapter presents the basis of the framework for building web-based collaborative systems that can be used for distributed manufacturing environments. It first outlines related work, followed by the concept and architecture of the framework. The *Wise-ShopFloor* concept is then demonstrated through a typical case study on device modelling, monitoring, and control. The benefits enabled by the framework include quick responses by reduced network traffic, flexible monitoring by sensor-driven 3D models, and interactive control by real-time feedback.

2. Business Service Management

Business firms generate revenues and profits through effective execution of business processes. During the last two decades, business processes have become increasingly automated, requiring IT service to support business operations. Delivering reliable and consistent levels of business and IT service is critical to the operations of business firms. In recent years, many business organizations have switched from a technical-focused IT management model to a business-oriented IT management framework that links technical capabilities to organizational needs. Using IT management tools to deliver real-time service level management not only meets service goals, but also generates greater business value for the organization [1, 2].

Recent advances in IT technologies have made a variety of new business processes possible. The proposed Web-based digital shop floor framework is an example of this type of new business processes that could not exist before. It uses new IT technology to improve the performance of business operations. This business process application is expected to result in significant increases in productivity and revenues.

The goal of our combined web-based and sensor-driven approach is to significantly reduce network traffic with Java 3D models, while still providing end users with an intuitive environment. The largely reduced network traffic also makes real-time monitoring, control, inspection, and trouble-shooting practical for users on relatively slow hook-ups such as modem connections. Participating in the collaborative system, users not only can feel reduced network traffic

by real-time interactions with quick responses, but also can obtain more flexible control of the real world. In the near future, open-architecture devices (such as OpenPLCs and Open-CNC Controllers, etc.) will have web servers and Java virtual machines embedded. This will make the proposed *Wise-ShopFloor* framework more efficient for real-time monitoring and control.

3. Research Background

Initially, parallel kinematic machines (PKMs) were developed based on the Stewart platform that is a 6 DOF prismatic parallel mechanism with extensible legs. Commercial hexapods including VARIAX of Giddings & Lewis, Ingersoll hexapod, Tornado of Hexel, Geodetic of Geodetic Technology Ltd., are all based on this structure. To overcome the problems in hexapods with extensible legs, such as stiffness and heat [3,4], recently, hexapods with fixed-leg lengths have been envisioned, for example, Hexaglide of the Swiss Federal Institute of Technology [5], LINAPOD of Stuttgart University [6], and HexaM of Toyoda [7]. Hexapods with extensible telescopic legs are not suitable for linear motors, but the fixed-leg length hexapods are. Hexapods with revolute joints were also reported in the literature, for example, DELTA robot, which can reach an acceleration of up to 20g in some area of the workspace [8].

The hexapods with fixed-leg lengths are sometimes termed as sliding-leg hexapods, because sliding of the fixed-length legs along their guideways drives the moving platform. There are basically three configurations in terms of guideway angle, vertical, horizontal and angular. In the vertical configuration, gravitational force is in the moving direction. While weight does not contribute to friction, motors have to overcome weight in the upward movement. In the horizontal configuration, gravitational force is perpendicular to the moving direction and weight would fully contribute to friction. Angular configuration is in between in terms of gravitational force and friction force.

Since machining operation requires five axes at most, new configurations with less than six parallel axes would be more appropriate. Development on new configurations is mainly on three axes PKMs. Examples include Triaglide [9], Tetrahedral Tripod [10], and Tricept of SMT Tricept [11]. Three axis PKMs can be combined with 2 axis systems, such as x-y stage, to form five axis machines. The Wise-ShopFloor is designed to provide users with a web-based and sensor-driven intuitive shop floor environment where real-time monitoring and control are undertaken. It utilizes the latest Java technologies, including Java 3D and Java Servlets, as enabling technologies for system implementation. Instead of camera images (usually large in data size), a physical device of interest (e.g. a milling machine or a robot) can be represented by a scene graph-based Java 3D model in an applet with behavioural control nodes embedded. Once downloaded from an application server, the Java 3D model is rendered by the

local CPU and can work on behalf of its remote counterpart showing real be-
haviour for visualization at a client side. It remains alive by connecting with
the physical device through low-volume message passing (sensor data and
user control commands). The 3D model provides users with increased flexibil-
ity for visualization from various perspectives, such as walk-through and fly-
around that are not possible by using stationary optical cameras; whereas the
largely reduced network traffic makes real-time monitoring, remote control,
on-line inspection, and collaborative trouble-shooting practical for users on
relatively slow hook-ups (e.g. modem and low-end wireless connections)
through a shared *Cyber Workspace* [12].

By combining virtual reality models with real devices through synchronized
real-time data communications, the *Wise-ShopFloor* allows engineers and shop
floor managers to assure normal shop floor operations and enables web-based
trouble-shooting – particularly useful when they are off-site.

Figure 1 shows how it is linked to a real shop floor. Although the *Wise-ShopFloor* framework is designed as an alternative of camera-based monitoring
systems, an off-the-shelf web-ready camera can easily be switched on remotely
to capture unpredictable real scenes for diagnostic purposes, whenever it is
needed. In addition to real-time monitoring and control, the framework can
also be extended and applied to design verification, remote diagnostics, virtual
machining, and augmented virtuality in construction. It is tolerant to hostile,
invisible or non-accessible environments (e.g. inside of a nuclear reactor or
outside of a space station).
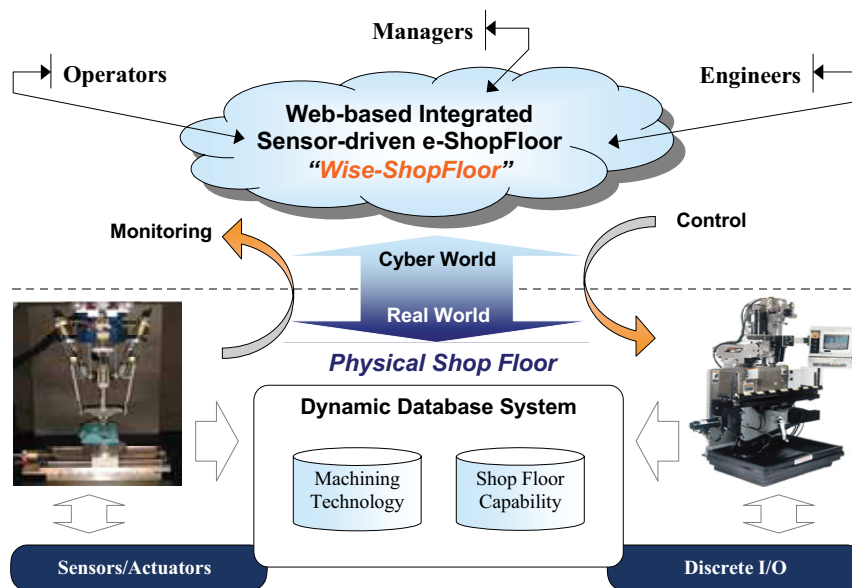
Figure 1. Concept of Wise-ShopFloor

4. Architecture Design

As shown in Figure 2, the framework is designed to use the popular client-server architecture and VCM (view-control-model) design pattern with built-in secure session control. The proposed solutions for meeting both the user requirements of rich visual data sharing and the real-time constraints are listed below.

1. Using interactive scene graph-based Java 3D models instead of bandwidth-consuming camera images for shop floor visualization;
2. Transmitting only the sensor data and control commands between models and device controllers for remote monitoring and control;
3. Providing users with thin-client graphical user interface for shop floor navigation; and
4. Deploying major control logic in a secured application server.
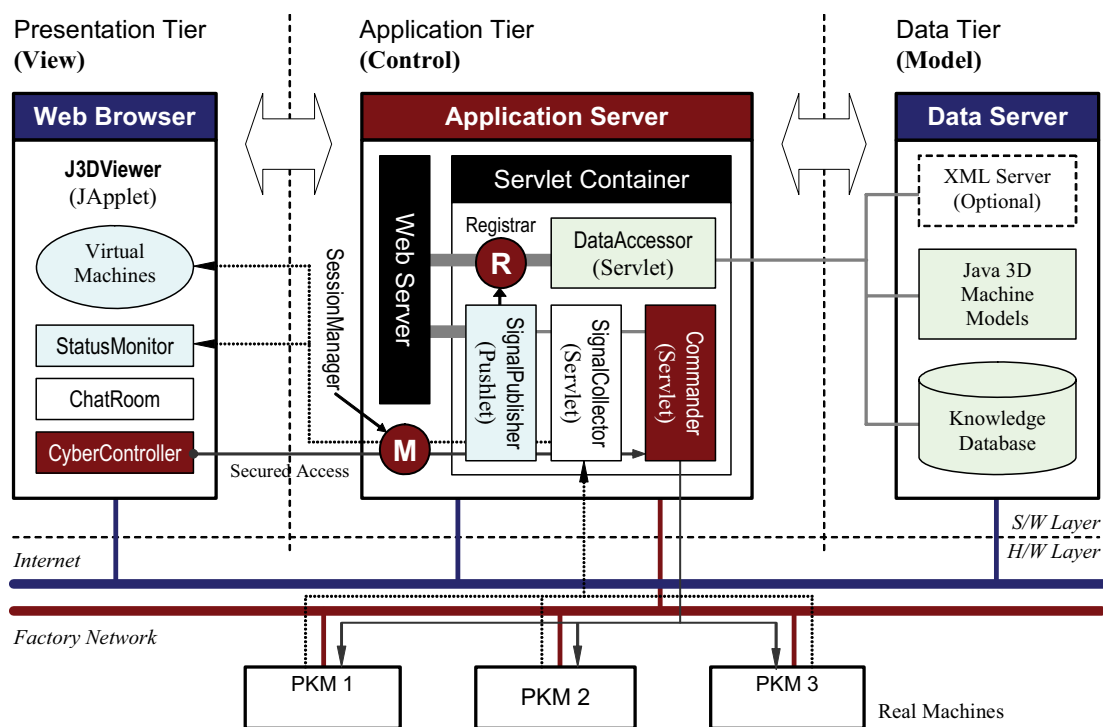
Figure 2. Architecture of Wise-ShopFloor

The mid-tier application server handles major security concerns, such as session control, viewer registration, data collection/distribution, and real device manipulation, etc. A central *SessionManager* is designed to look after the issues of user authentication, session control, session synchronization, and sensitive

data logging. All initial transactions need to pass through the *SessionManager* for access authorization. In a multi-client environment – the *Wise-ShopFloor*, different clients may require different sets of sensor data for different models. Constrained by network security, a Java 3D model residing in an applet is not allowed to communicate directly with a real device through socket communication. It is also not efficient to have multiple clients who share the same model talking with the same device at the same time. The publish-subscribe design pattern is adopted to collect and distribute sensor data at the right time to the right client efficiently. As a server-side module, the *SignalCollector* is responsible for sensor data collection from networked physical devices. The collected data are then passed to another server-side module *SignalPublisher* who in turn multicasts the data to the registered subscribers (clients) through applet-servlet communication. A *Registrar* is designed to maintain a list of subscribers with the requested sensor data. A Java 3D model thus can communicate indirectly with sensors no matter where the client is, inside a firewall or outside. The JMF (Java Media Framework) is chosen for the best combination between applets and servlets. For the same security reasons, a physical device is controllable only by the *Commander* that resides in the application server. Another server-side component called *DataAccessor* is designed to separate the logical and physical views of data. It encapsulates JDBC (Java Database Connectivity) and SQL codes and provides standard methods for accessing data (Java 3D models, knowledge base of the devices, or XML documents). The knowledge base is found helpful for device trouble-shooting, while XML will be used for high-level data communication in future extensions.

Although the global behaviours of Java 3D models are controlled by the server based on real-time sensor signals, users still have the flexibility of monitoring the models from different perspectives, such as selecting different 3D machine models, changing viewpoint, and zooming, through *J3DViewer* at the client side. Authorized users can submit control commands through *CyberController* to the application server. The *Commander* at server-side then takes over the control for real device manipulations. Another client-side module *StatusMonitor* can provide end users with a view of run-time status of the controlled device. For the purpose of collaborative trouble-shooting, a *ChatRoom* is included in the framework for synchronized messaging among connected users.

A proof-of-concept prototype is developed on top of the framework to demonstrate its application on remote monitoring and control. Figure 3 shows one snapshot of the web user interface of the prototype. A more detailed discussion from device modelling to control is provided in Section 6 through a case study of a Tripod test bed.
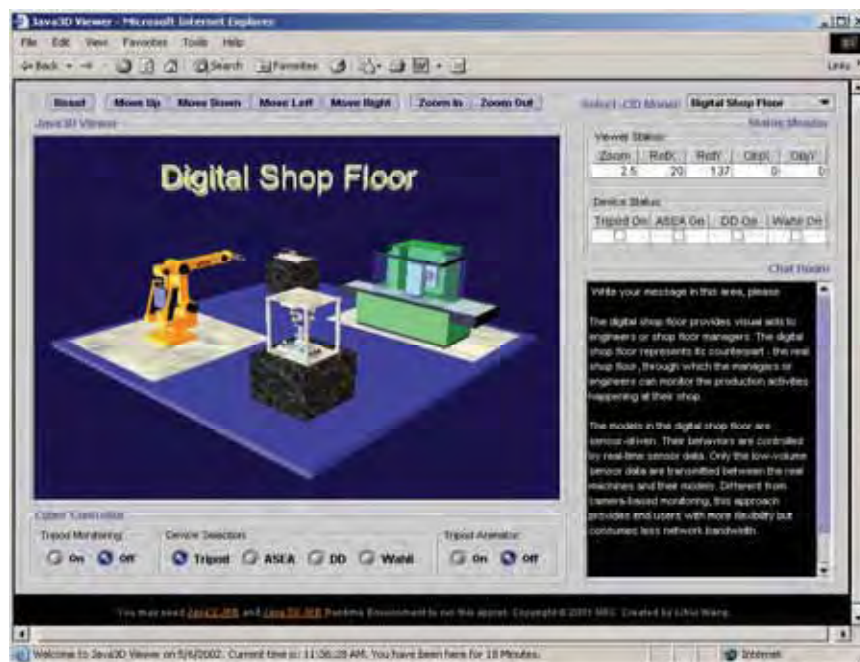
Figure 3. Web user interface for remote monitoring and control

5. Shop Floor Security

According to an NCMS report [13], there is a growing consensus that linking shop floor hardware to the Internet will become the backbone technology for collaborative manufacturing. However, a major concern of implementing Internet or Web-based collaborative manufacturing systems is the assurance that proprietary information about the intellectual property owned by the organization or information about the company's operations is available only to authorized individuals. Any web-based collaborative systems must accommodate privacy of the individuals and organizations involved in collaborative activities. Gathering and processing information about the activities of individuals or groups while managing or operating processes or devices via computer networks can provide a great deal of detail concerning the ways in which the individuals interact as well as process-related information. In a highly competitive manufacturing environment, the information about the operations of or the information provided by individuals or organizations should only be shared by those involved. Clearly, it is also important to avoid security disasters for hardware at shop floor level. Web-based remote monitoring and control typically involve sharing information in the form of detailed run-time operations, as well as real-time and mission-critical hardware controls. For general acceptance of the *Wise-ShopFloor*, the secrecy of the proprietary information must be properly maintained. For security, our approach depends on the familiar security infrastructure built into the Java platform. This security

architecture consists of byte-code verification, security policies, permissions, and protection domains. In addition to the security infrastructure, other security and privacy issues are considered in the framework for implementation, including digital rights management for information access and sharing, data encryption, and process confidentiality protection.

Figure 4 shows how a remote end user can get access indirectly to the real shop floor without violating shop floor security. All data communication between the end user and a shop floor device goes through the application server, and is processed by a server-side module before passing the data onto its receiver. As mentioned in Section 4, only the server-side modules are allowed to collect sensor data or manipulate devices within their limits. On the other hand, all end users are physically separated from the real shop floor by using segmented networks (Intranet/Internet, and Factory Network) with the application server as a gateway.
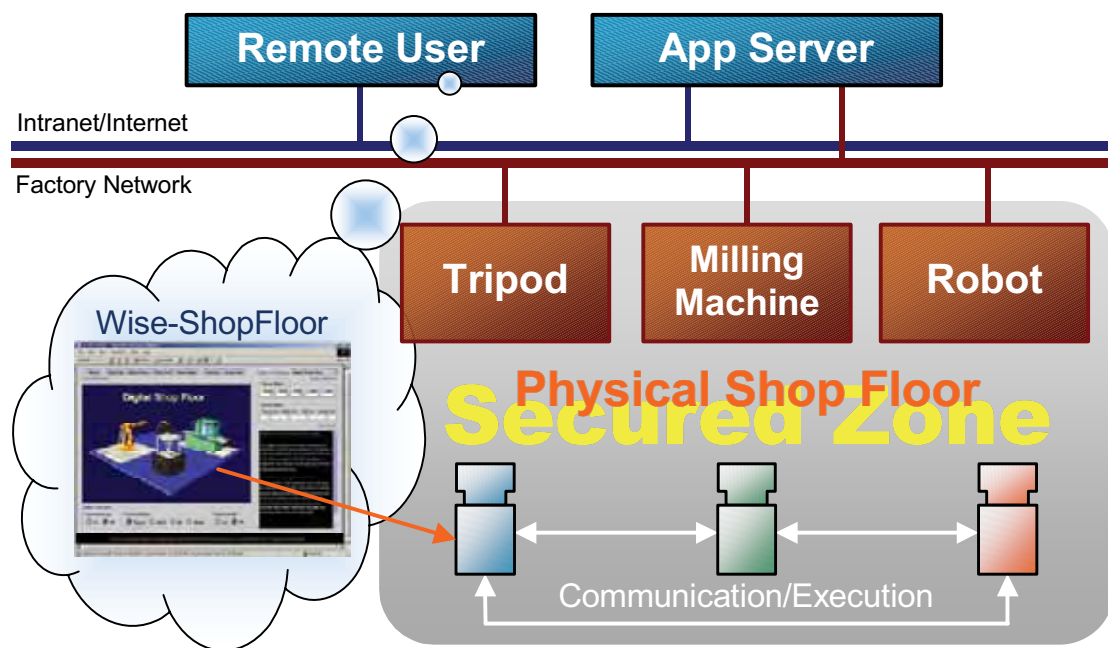
Figure 4. Indirect secure access to physical shop floor

6. Implementation

This section describes how a physical device is modelled, monitored, and controlled. The Tripod is a parallel kinematic machine developed at IMTI's lab [14]. Instead of camera images, the Tripod is modelled by using the scene graph-based interactive Java 3D with behavioural control nodes embedded. Once downloaded from the application server, it behaves in the same way as its physical counterpart for remote monitoring and control at client-side, facilitated by the model-embedded kinematics and sensor signals of the real Tripod.

6.1 Java 3D Modelling for Tripod

Java 3D is designed to be a mid to high-level fourth-generation 3D API [15]. What sets a fourth-generation API apart from its predecessors is the use of scene-graph architecture for organizing graphical objects in the virtual 3D world. Unlike the display lists used by the third-generation APIs (such as VRML, OpenInventor, and OpenGL), scene graphs can isolate rendering details from users while offering opportunities for more flexible and efficient rendering. Enabled by the scene-graph architecture, Java 3D provides an abstract, interactive imaging model for behaviour and control of 3D objects. Because Java 3D is part of the Java pantheon, it assures users ready access to a wide array of applications and network support functionality [16]. Java 3D differs from other scene graph-based systems in that scene graphs may not contain cycles. Thus, a Java 3D scene graph is a directed acyclic graph. The individual connections between Java 3D nodes are always a direct relationship: parent to child. Figure 5 illustrates a scene graph architecture of Java 3D for the Tripod. This test bed is a gantry system, which consists of an x-table and a Tripod unit mounted on a y-table. The end effecter on the moving platform is driven by three sliding-legs that can move along three guide-ways, respectively.

As shown in Figure 5, the scene graph contains a complete description of the entire scene with a virtual universe as its root. This includes the geometry data, the attribute information, and the viewing information needed to render the scene from a particular point of view. All Java 3D scene graphs must connect to a *Virtual Universe* object to be displayed. The *Virtual Universe* object provides grounding for the entire scene. A scene graph itself, however, starts with *BranchGroup* (BG) nodes (although only one BG node in this case). A *BranchGroup* node serves as the root of a sub-graph, or branch graph, of the scene graph. The *TransformGroup* nodes inside of a branch graph specify the position, the orientation, and the scale of the geometric objects in the virtual universe. Each geometric object consists of a *Geometry* object, an *Appearance* object, or both. The *Geometry* object describes the geometric shape of a 3D object. The *Appearance* object describes the appearance of the geometry (colour, tex-

ture, material reflection characteristics, etc.). The behaviour of the Tripod model is controlled by *Behaviour* nodes, which contain user-defined control codes and state variables. Sensor data processing can be embedded into the codes for remote monitoring. Once applied to a *TransformGroup* node, the so-defined behaviour control affects all the descending nodes. In this example, the movable objects (X-Table, Y-Table, and Moving Platform) are controlled by using three control nodes, for on-line monitoring/control and off-line simulation. As the Java 3D model is connected with its physical counterpart through the control nodes by low-volume message passing (real-time sensor signals and control commands, etc.), it becomes possible to remotely manipulate the real Tripod through its Java 3D model (see also [17]).
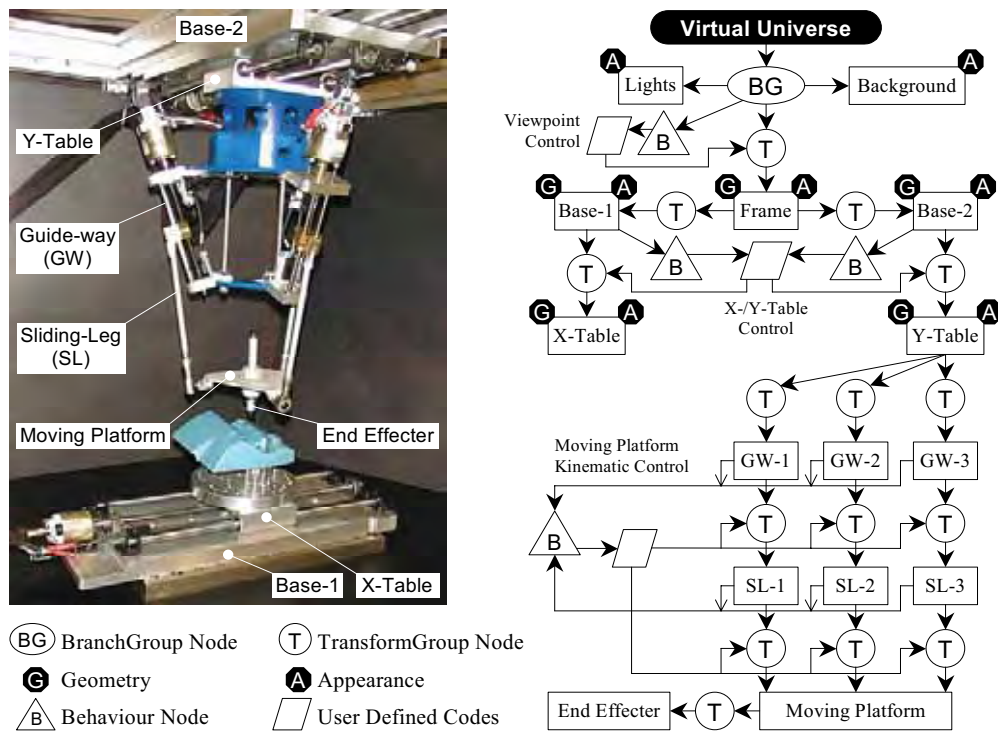
Figure 5. Java 3D scene graph architecture for Tripod

6.2 Kinematic Modelling for Tripod

Kinematics studies the geometric properties of the motion of points without regard to their masses or to the forces acting upon them. While the scene graph is the emergent standard hierarchical data structure for computer modelling of 3D worlds, kinematic models of physical devices or mechanisms that have external constraints or constraints that span interior nodes do not fit comfortably

into its open-branched tree topology. In the case of our Tripod monitoring and control, models of both constrained kinematics and inverse kinematics are solved separately and embedded into the behaviour control nodes in a scene graph to calculate the motions of respective components. Typically, constraints can be expressed in a number of equations or inequalities that describe the relationships among Tripod components. Based on sensor signals collected from the real Tripod, both constrained kinematic model and inverse kinematic model of the Tripod are needed to calculate the positions and orientations of the three sliding-legs and moving platform for 3D Tripod model rendering.

For the purpose of mathematical formulation, a Tripod kinematic model is shown upside-down in Figures 6 and 7. It is a 3-dof parallel mechanism with linear motion component actuators, the type of linear motion actuated machines with fixed leg lengths and base joints movable on linear guideways (e.g. HexaM, Eclipse, Hexaglide, GeorgV, Z^3 Head). This mechanism consists of three kinematic chains, including three fixed length legs with identical topology driving by ballscrews, which connects the fixed base to the moving platform. In this 3-dof parallel mechanism, the kinematic chains associated with the three identical legs consist, from base to platform, of an actuated linear motion component (ballscrew in the case), a revolute joint (connected by a nut), a fixed length moving link, and a spherical joint attached to the moving platform. The arrangement of the structure would be subject to bending in the direction parallel to the axis of the revolute joint. The advantages of the structure are: 1) with this basic structure of parallel mechanism, it can be easily extended to 5-dof by adding two gantry type of guideways to realize the 5-dof machining; 2) with the fixed length legs, one can freely choose the variety of leg forms and materials and use linear direct driver to improve the stiffness; and 3) due to reduced heat sources, it is possible to keep the precision in a high level and to maintain a stable stiffness if compared with variable legs.

The kinematic equation for the position of the ith spherical joint is given as

$$\mathbf{p}_i = \mathbf{h} + \mathbf{R}\mathbf{p}'_i \tag{1}$$

where, $\mathbf{p}_i = [\,p_{ix}, p_{iy}, p_{iz}\,]^T$ is the vector representing the position of the ith joint in the global coordinate system O-xyz, $\mathbf{p}'_i$ is the vector representing the same point but in the local coordinates C-$x'y'z'$, $\mathbf{h} = [\,x_c, y_c, z_c\,]^T$ is the vector representing the position of the moving platform, and $\mathbf{R}$ is the rotation matrix of the moving platform in terms of rotation angles θ_x, θ_y, and θ_z about x, y, and z axis, respectively.
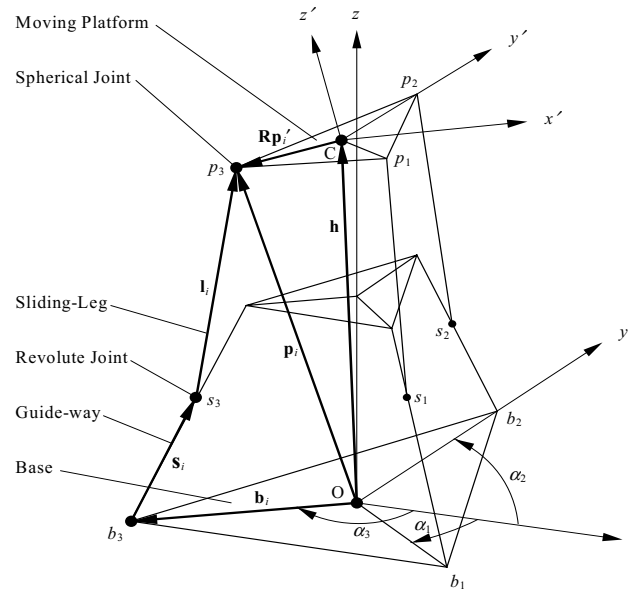
Figure 6. Tripod kinematic model

Among the six motion components of the moving platform, it is known that x_c, y_c, and θ_z are dependent variables. The constraint equations can be derived as

$$x_c = -\frac{\sqrt{3}}{3} L_p \cos\theta_y \sin\theta_z \tag{2a}$$

$$y_c = \frac{\sqrt{3}}{6} L_p \left(-\cos\theta_y \cos\theta_z + \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_y \sin\theta_z \right) \tag{2b}$$

$$\theta_z = \arctan\left(-\frac{\sin\theta_x \sin\theta_y}{\cos\theta_x + \cos\theta_y} \right) \tag{2c}$$

While constrained kinematics of the Tripod is used for monitoring, inverse kinematics is needed for position control. Considering the ith sliding-leg/guide-way system, the kinematic equation of the position of the ith spherical joint, i.e. eq. (1), can be re-written as

$$\mathbf{p}_i = \mathbf{b}_i + \mathbf{s}_i + \mathbf{l}_i \tag{3}$$

where $\mathbf{b}_i$ is the vector representing the position of the lower end of the ith guide-way attached to the base, $\mathbf{s}_i$ is the vector representing the displacement along the ith guide-way, and $\mathbf{l}_i$ is the vector representing the ith sliding leg.

Since $\mathbf{s}_i = s_i \mathbf{u}_i^s$ and $\mathbf{l}_i = l_i \mathbf{u}_i^l$, where $\mathbf{u}_i^s$ and $\mathbf{u}_i^l$ are the direction vectors of the ith guide-way and the ith leg, respectively, the actuator displacement s_i can be solved considering that the leg length is a constant

$$\left| \mathbf{p}_i - \mathbf{b}_i - s_i \mathbf{u}_i^s \right| = l_i \qquad (4)$$

where l_i is the length of the ith sliding leg. For given θ_x, θ_y, and z_c, dependent variables x_c, y_c, and θ_z can be determined by eqs. (2a) - (2c), then $\mathbf{h}$ and $\mathbf{R}$ are fully defined. With this, $\mathbf{p}_i$ can be determined by eq. (1), and subsequently s_i can be solved using eq. (4). The true solution of eq. (4) should be the one closer to the previous value, that is

$$s_i = \left(s_i^{(k)}, \quad \min_{k=1,2} \left| s_i^{(k)} - s_i(j-1) \right| \right) \qquad (5)$$

where j stands for the jth step. In practice, the initial value of s_i is provided by an encoder. For the sake of brevity, interested readers are referred to [18] for further details of the Tripod kinematics.

6.3 Remote Monitoring and Control

Web-based remote device monitoring and control are conducted by using the *StatusMonitor* and *CyberController*, which communicate indirectly with the device controller through an application server. In the case of Tripod monitoring and control, they are further facilitated by the kinematic models, to reduce the amount of data travelling between web browsers and the Tripod controller.

Figure 7. CAD model of Tripod

The required position and orientations of the moving platform are converted into the joint coordinates s_i ($i = 1, 2, 3$) by the inverse kinematics for both Java 3D model rendering at client-side and device control at server-side. The three sliding-legs of the Tripod are driven by three 24V DC servomotors combined with three lead screws. Each actuator has a digital encoder (1.25 µm/count) for position feedback. The position data s_i ($i = 1, 2, 3$) of the sliding-legs are multi-cast to the registered clients for remote monitoring, while only one user at one time is authorized to conduct remote control. A sampling rate of 1 kHz is used for the case study. Figure 8 shows how the Tripod is manipulated from one state to another within the proposed *Wise-ShopFloor* framework.
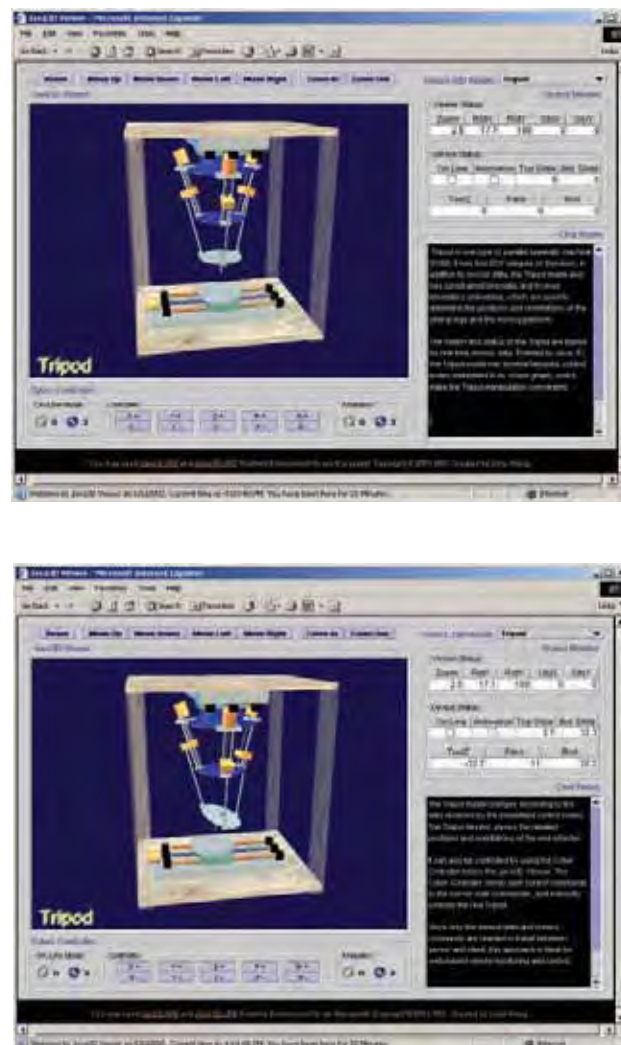
Figure 8. Web-based remote monitoring and control

6.4 Managerial Implications

The *Wise-ShopFloor* is a business process that is based on new IT technology to execute business processes. It leverages the IT management tools to deliver reliable and secured transmission of data between the end users and real shop floors. It provides not only an efficient mechanism for real-time monitoring and control in manufacturing, but it also improves a manufacturing firm's business performance. The implementation of this client-server architecture is likely to result in significant increases in productivity and revenues.

7. Conclusions

This chapter presents the *Wise-ShopFloor* framework and describes detailed three-tier architecture. The goal of the web-based approach is to reduce network traffic with Java 3D models, while still providing users with intuitive environments. Participating in the *Wise-ShopFloor*, users not only can feel reduced network traffic by real-time interactions, but also can obtain more flexible control of their real shop floors. The application in modern manufacturing system is demonstrated for its feasibility and the promise of this novel approach to the growing distributed shop floor environments. As decentralization of business grows, a large application potential of this research is anticipated, in addition to remote real-time monitoring and control.

8. References

Cao, J., Li, M.L., Zhang, S.S. and Den, Q. N. 2004. Composing Web Services Based on Agent and Workflow. Grid and Cooperative Computing, Part 1. Berlin: Springer-Verlag Berlin, pp. 948-955.

Zeng, L. Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., and Chang, H. 2004. QoS-Aware Middleware for Web Services Composition", IEEE Transactions on Software and Engineering, May. 30(5): 311-327.

G. Pritschow, "Research and Development in the Field of Parallel Kinematic Systems in Europe", Parallel Kinematic Machines – Theoretical Aspects and Industrial Requirements, edited by Boër, C.R., Molinari-Tosatti, L, and Smith, K.S., pp.1-16, Springer-Verlag, (1999).

J. Tlusty, J. Ziegert, and S. Ridgeway, "Fundamental Comparison of the Use of Serial and Parallel Kinematics for Machine Tools", Annals of the CIRP, Vol. 48/1, pp. 351-356, (1999).

M. Honegger, A. Codourey, and E. Burdet, "Adaptive Control of the Hexaglide, a 6 DOF Parallel Manipulator", Proceedings of the 1997 IEEE

International Conference on Robotics and Automation, Vol. 1, pp. 543-548, (1997).

G. Pritschow, and K.-H. Wurst, "Systematic Design of Hexapods and Other Parallel Link Systems", Annals of the CIRP, Vol. 46/1, pp. 291-295, (1997).

M. Suzuki, K. Watanabe, T. Shibukawa, T. Tooyama, and K. Hattori, "Development of Milling Machine with Parallel Mechanism", Toyota Technical Review, Vol. 47 No. 1, pp. 125-130, (1997).

F. Pierrot, "From Hexa to HexaM", International Parallelkinematik-Kolloquium IPK'98, ETH Zurich, pp. 75-84, (1998).

H.K. Tönshoff, C. Soehner, and H. Ahlers, "A New Machine Tool Concept for Laser Machining", Proceedings of International Seminar on Improving Machine Tool Performance, San Sebastian, pp.199-124, (1998).

B.S. El-Khasawneh, and P.M. Ferreira, "The Tetrahedral Tripod", Parallel Kinematic Machines – Theoretical Aspects and Industrial Requirements, edited by Boër, C.R., Molinari-Tosatti, L, and Smith, K.S., pp. 419-430, Springer-Verlag, (1999).

Kochan, A., "Parallel Robots Perfect Propellers", Industrial Robot, Vol. 23, No. 4, pp. 27-30, (1996).

L. Wang, B. Wong, W. Shen and S. Lang, "Java 3D Enabled Cyber Workspace", *Communications of the ACM*, Vol.45, No.11, pp. 45 – 49, 2002.

NCMS, "Factory-Floor Internet: Promising New Technology or Looming Security Disaster", *Manufacturing In Depth*, National Center for Manufacturing Sciences, November, 2001.

Dan Zhang, L. Wang and Sherman Y. T. Lang, 2005, Parallel Kinematic Machines: Design, Analysis and Simulation in an Integrated Virtual Environment, Transactions of the ASME Journal of Mechanical Design, Vol.127, Issue 7, pp. 580-588

J. Barrilleaux, *3D User Interfaces with Java 3D*, Manning Publications Co., 2001.

H. Sowizral, K. Rushforth and M. Deering, *The Java 3D API Specification*, Addison-Wesley, 2001.

L. Wang, F. Xi, D. Zhang and M. Verner, "Design Optimization and Remote Manipulation of a Tripod", International Journal of Computer Integrated Manufacturing, Vol.18, No.1, pp.85-95, 2005

Human-Robot Interaction Control for Industrial Robot Arm through Software Platform for Agents and Knowledge Management

Tao Zhang, Vuthichai Ampornaramveth and Haruki Ueno

1. Introduction

At present, industrial robot arms have been widely adopted in many areas. Unfortunately, operation of them is not easy to master for workers due to complex architectures as well as various control patterns required for each situation. Therefore, if there is a user-friendly human-robot interface and through this interface workers can operate industrial robot arms by their familiar language, it will remarkably reduce the difficulty of the usage of them. The aim of this research is to develop a new human-robot interaction control approach for industrial robot arms by means of software platform for agents and knowledge management in order to construct a symbiotic human-robot system that could be adopted in industrial area (Ueno, 2002).

Conventionally, industrial robot arms only can be operated by experts who should possess sufficient knowledge on features of industrial robot arms and be able to control their movement for performing a task. To improve the human-robot interface, researchers have already developed some software interfaces for the operation of industrial robots (Mizukawa et al, 2002) (Konukseven et al, 2004) (Sales et al, 2004) (Cengiz, 2003). Unfortunately, effective use of these interfaces still depends on the technical training. This paper proposes a knowledge-based human-robot interaction control approach in conjunction with a humanoid robot Robovie as a communication robot (Tao Zhang et al, 2005) for industrial robot arms. With this method, an operator can easily interact with the autonomous communication robot by his natural language. The communication robot transfers the sentences of the operator to a sentence parser. The key words extracted by the sentence parser are then sent to a software platform, called SPAK (Software Platform of Agents and Knowledge Management) (Ampornaramveth et al, 2004). In SPAK, it maintains sufficient knowledge on this human-robot system. According to the defined human-robot interaction control in SPAK, industrial robot arms can be correctly operated according to operator's request. With the proposed method, a person is not required to be an expert of industrial robot arms but just an ordinary operator of industrial robot arms. He can operate industrial robot arms like an expert.

Although there are many types of industrial robot arms to be operated, the knowledge on these robots can be easily defined in SPAK in a uniform manner by using frame-based knowledge representation schema. The knowledge is maintained as a key component of a communication robot, i.e. a dialog robot, for the robot arms. Therefore, a person only needs to explain his requests to the communication robot. SPAK can assist the person to select appropriate robots and arrange their operations to satisfy the person's requests to achieve tasks. In addition, SPAK can control different types of robots even they use various kinds of operation systems. From the side of operators, it is no need to possess knowledge on the operations of different types of robots.

The remainder of this chapter is organized as follows. In section 2, human-robot system as well as its interaction control process is modelled by frame-based knowledge representation. In section 3, human-robot system is defined in SPAK according to its knowledge model using XML format. Through human-robot interaction, industrial robot arm is controlled by use of SPAK via wireless network in section 4. Section 5 introduces an actual system comprised of human, humanoid robot (Robovie) and industrial robot arm (MELFA) and its experimental results demonstrate the effectiveness of the proposed human-robot interaction control method.

2. Modelling of Human-Robot System

Human-robot interaction control for an industrial robot arm is based on the interaction between an operator of the robot arm and a communication robot. The operator's request is transferred to SPAK via wireless network and converted into commands of the robot arm. SPAK can control the robot arm with these commands and get the feedback signals from the robot. By converting these feedback signals into the sentences of natural language and speaking out these sentences by the communication robot, the operator can understand the status of the robot arm and continue his operation successfully. From this operation process, the definition of human-robot system in SPAK is one of the important components. In order to implement the definition of human-robot system, the modelling of human-robot system is necessary.

The modelling of human-robot system is based on the frame-based knowledge representation. It is well known that frame representation systems are currently the primary technology used for large-scale knowledge representation in Artificial Intelligence (AI) (Koller & Pfeffer, 1998). A frame is a data-structure for representing a stereotyped situation (Minsky, 1974). Attached to each frame are several kinds of information, called knowledge. Collections of related frames are linked together into frame-systems. The structure of a frame is consisted of several items, such as Frame name, Frame type, A-kind-of, Descendants, Slots, etc. (Tairyou, 1998). A frame consists of slots, each of which

has different roles in description of knowledge. Table 1 and 2 shows the definition of a frame as well as its slot.

Items	Meanings
Frame name	Identification for frame
Frame type	Type of frame
A-kind-of	Pointer to parent frame for expressing IS_A relation
Descendants	Pointer list to children frame
Has-part	Components of this frame
Semantic-link-from	Links from other frames according to their semantic relation
Semantic-link-to	Links to other frames according to their semantic relations
Slots	Components of the frame

Table 1.Meanings of each item in a frame

Items	Meanings
Slot name	Identification for slot
Role	Purpose of slot
From	Source of slot
Data type	Explain the attribute of information recorded into the value
Value	Slot value
Condition	Condition of slot
Argument	Argument for slot
If-required	If slot is required, check this item.
If-shared	If slot can be shared with other frames, check this item.
Frame-related	A frame related with slot
Frame-list-related	Several frames related with slot
Default	If the slot value is not determined, the default value can be recorded. But the value and default value cannot be given at the same time.

Table 2.Meanings of each item in a slot

Using frames and their slots, a human-robot system can be modelled simply. This knowledge model is comprised of different frames to represent various pieces of knowledge. For instance, frames for robots include features of a communication robot and industrial robot arms as well as their operation commands. Particularly, the frames for the communication robot include the knowledge on a human-robot interface. At present, a human-robot interface can be implemented by vision recognition, robot speech, physical input, etc. While, frames for users include much information about the users. All frames for the knowledge model are organized by their ISA relations in a hierarchy. That is, a lower level frame is a subclass of its upper level frame. The bottom frames are the instances of the upper level frame. Based on these relations, a

human-robot interaction control can be defined in frames. Table 3 illustrates a part of an example of a frame about a communication robot.

Frame:	Communication robot
Type:	Instance
A-kind-of:	Robot
Descendants:	Empty
Has-part:	(mouth, motor, eyes)
Semantic-link-from:	Empty
Semantic-link-to:	Behavior
…	…

Table 3. A frame for communication robot

In the model of a human-robot system, the human-robot interaction control can be described as below. As a human operator is interacting with a communication robot, the sentences given by the operator are recognized and transferred to a software sentence parser. By this parser, the key words extracted from the sentences are sent to SPAK. In the SPAK knowledge base there exist many frames on the operation for the industrial robot arm. If the condition of an operation of the robot arm is satisfied according to the key words, the commands to the robot arm defined in this frame will be sent to the robot arm. And the robot arm will then move along the commands. After movement, the robot arm will send a feedback signal to SPAK through a software agent, which translates the feedback signal into the key words that the SPAK knowledge system can understand. As SPAK receives them, it will form some sentences corresponding to these words and send them to the communication robot. The operator can then hear the spoken sentences from the communication robot and decide the next operation.

3. Definition of Human-Robot System in Software Platform

3.1 SPAK

In order to implement a human-robot interaction control for an industrial robot arm, a software platform, called SPAK (Software Platform for Agents and Knowledge Management), has been recently developed. It is a frame-based knowledge engineering environment (Ampornaramveth et al, 2004). It provides a central module, which acts as a blackboard for comminucation channels, knowledge processing brain, memory, and does judgment, task planning and execution. It also provides software tools necessary for integration of various

existing modules over a TCP/IP network. The features of SPAK are "platform-independent" as existing robots and software modules often rely on different platforms or operation systems, "network-aware" as the modules must interact on a network, supporting "software agent" and being "user friendly". SPAK is targeted to be the platform on which a group of coordinative robots (or their agents) operate on top of frame knowledge.

SPAK consists of the following software components:

- **GUI Interface:**

  A user-friendly graphical interface to the internal knowledge manager and the inference engines. It provides the users direct access to the frame-based knowledge.

- **Knowledge Database and Knowledge Manager:**

  This is the SPAK core module, which maintains the frame systems as Java class hierarchy, and performs knowledge conversion to/from XML format.

- **Inference Engines:**

  Verify and process information from external modules, which may result in instantiation or destruction of frame instances in the knowledge manager, and execution of predefined actions.

- **JavaScript Interpreter:**

  Interprets JavaScript code which is used for defining condition and procedural slots in a frame. It also provides access to a rich set of standard Java class libraries that can be used for customizing SPAK to a specific application.

- **Base Class for Software Agent:**

  Provide basic functionality for developing software agents that reside on networked robots.

- **Network Gateway:**

  This is a daemon program allowing networked software agents to access knowledge stored in SPAK. All SPAK network traffics are processed here.

3.2 Definition of human-robot system in SPAK

In SPAK, a human-robot system is defined according to its knowledge model by means of XML format (www.xml.com). Table 4 illustrates an example about the frame described in the XML format in SPAK for the behavior of face detection of a communication robot, which is implemented for an experiment. XML is a markup language for Internet documents containing structured information.

```
<FRAME>
  <NAME>FaceDetect</NAME>
  <ISA>Behavior</ISA>
  <ISINSTANCE>FALSE</ISINSTANCE>
  <SLOTLIST>
    <SLOT>
      <NAME>mSensor</NAME>
      <TYPE>TYPE_INSTANCE</TYPE>
      <CONDITION>COND_ANY</CONDITION>
      <ARGUMENT>Bust</ARGUMENT>
      <VALUE></VALUE>
      <REQUIRED>TRUE</REQUIRED>
      <SHARED>TRUE</SHARED>
    </SLOT>
    <SLOT>
      <NAME>mMouth</NAME>
      <TYPE>TYPE_INSTANCE</TYPE>
      <CONDITION>COND_ANY</CONDITION>
      <ARGUMENT>MouthIO</ARGUMENT>
      <VALUE></VALUE>
      <REQUIRED>TRUE</REQUIRED>
      <SHARED>TRUE</SHARED>
    </SLOT>
    <SLOT>
      <NAME>onInstantiate</NAME>
      <TYPE>TYPE_STR</TYPE>
      <CONDITION>COND_ANY</CONDITION>
      <ARGUMENT></ARGUMENT>
      <VALUE>rpc("recogFace", "");</VALUE>
      <REQUIRED>FALSE</REQUIRED>
      <SHARED>TRUE</SHARED>
    </SLOT>
  </SLOTLIST>
</FRAME>
```

Table 4. XML format in SPAK

With XML format, frame structure as well as its contents written by slots can be defined easily. Particularly, the frame system can be illustrated in the SPAK Graphic User Interface (GUI). Besides, corresponding to the XML file, there is an interpreter to translate XML-based specifications into relative commands.
The meaning of this format can be explained corresponding to the definition of a frame in Table 1 and 2. Between <FRAME> and </FRAME> defines a frame. NAME refers to the frame name. ISA refers to the item of "A-kind-of". ISINSTANCE refers to the frame type. Between <SLOTLIST> and </SLOTLIST> defines slots. Each SLOT gives several contents, including NAME, TYPE, CONDITION, ARGUMENT, VALUE, REQUIRED, SHARED, etc. With this XML format, a frame and its slots can be described in detail. Therefore, human-robot system can be defined in SPAK by means of XML format.

4. Implementation of Human-Robot Interaction Control

In section 2, the interaction control process for an industrial robot arm has been explained. As implementing the human-robot interaction control, several technologies should be integrated and employed by SPAK. Firstly, for the interaction between an operator and a communication robot, SPAK integrates several technologies for implementing a human-robot interface, such as face detection, image recognition, speech recognition etc. Concerning face detection, by use of the robot "eyes" which are installed with video cameras, the robot can get the video image of the human's face. With the face detection program stored in the embedded computer, the robot can recognize the human by the computer vision technology. Of course, if it is the first time for this specific person, the robot will store the information about this person's face in his knowledge database by its learning function. When the robot looks this person's face again, he can recognize him at once. Thus it sends the recognition results to SPAK. Another technique that robot can use for human-robot interface is by means of speech. Since the robot system has a software program for speech recognition, SPAK just sends the recognized sentences to the communication robot, who can speak out the sentences sent by SPAK to the operator.

Another important technology is to parse a natural sentence. In this human-robot system, an operator can give his requests to the robot arm in natural language. By means of voice recognition software, such as IBM ViaVoice (http://www-306.ibm.com/software/voice/viavoice/), the sentences given by the operator can be recognized and stored into the computer. Since SPAK can generate the commands for the robot arm by several key words, it is necessary to parse the sentences and to extract the key words from them for SPAK. We have developed a simple sentence parser for SPAK using the technique of Case Grammar taking into account the features of the operations of robot arm (Bruce, 1975). With this software component, key words about objective of control, movement of robot arm, etc., will be picked out and sent to SPAK. If some inconsistencies are detected in the sentences regarding the operation of the robot arm, this parser can also generate some relative words about the inconsistencies and send them to SPAK. SPAK will ask the operator to change his commands again through the speech module of the communication robot.

Although the system hierarchy has expressed the relations among frames, the execution of frame contents must be implemented in SPAK by inference engines. The inference engines used in SPAK are feedforward and backward chainings. The feedforward chaining is usually used when a new fact is added to the knowledge base and we want to generate its consequences, which may add new other facts, and trigger further inferences. The backward chaining starts with something we want to prove, find implication facts that would allow us to conclude it. It is used for finding all answers to a question posed to the knowledge base.

Besides, the control of robots also needs their different driving programs of robots for their actions as well as some software agents to convert the signals between robots and SPAK. Particularly, for feedback control, the robot arm should be able to generate feedback signal about their status and send them to SPAK.

5. Experiment

5.1 Experiment setup

An experimental symbiotic human-robot system with human-robot interaction control for industrial robot arms has been developed, which composed of Robovie (autonomous communication robot), MELFA (industrial robot arm) and SPAK, as illustrated by Fig. 1.

Robovie was developed by Advanced Telecommunications Research Institute International (ATR) of Japan. It is designed as a kind of autonomous communication robots that can communicate and interact with human in our daily environment. It has 11 degrees of freedom (DOFs) for body motions and sensors such as an omni-directional camera, a set of pan-tilt zooming cameras, a microphone, ultrasonic range sensors, tactile sensors, etc. It can move around with its wheels. Robovie also provides many demo programs for speech, vision, motion, etc. Based on the embedded functions of Robovie, we have developed several programs for monitoring the status of all sensors and sending them to SPAK, defining many actions of Robovie as well as instructing Robovie to talk with human.
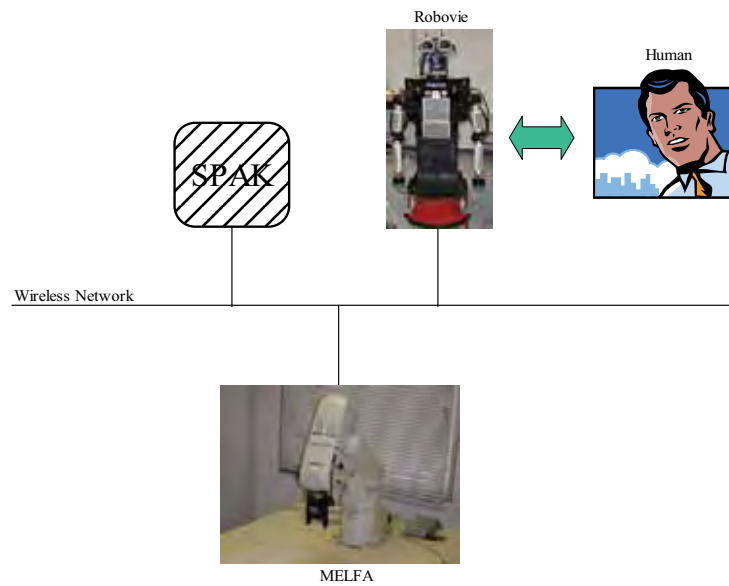
Figure 1. An actual human-robot system

Robovie can be used in the field of education, nursing, medical treatment, etc. Robovie and human can make interaction by face detection, speech and user input, so that Robovie can get the operator's commands and transfer it to SPAK.

MELFA is a kind of industrial articulated robot arm, produced by Mitsubishi Electric Co., Japan The robot arm used in this experiment has 6 DOFs generated by 6 axes. Each axis is driven by AC servomotor. The servo controller of MELFA is CR1-571. Its programming language is MELFA-BASIC IV. MELFA can move strictly along a trajectory in three-dimensional space. The tip of the robot arm can hold or release workpiece.

Figure 2. Environment of operation-Robovie system

In this experiment system, Robovie, MELFA, SPAK and other software components are connected via wireless TCP/IP network. SPAK and other software components (such as vision recognition, sentence parser, software agents, etc.) are running on different computers. Therefore, this system is comprised of a distributed system. The core of this system is SPAK. Fig.2 illustrates the environment of operator-Robovie interaction for the experiment.

5.2 Results

Before performing the task in experiment, the human-robot system must be defined in SPAK according to the knowledge model of this system. Using the modeling method explained in section 2 the system can be modeled and defined in SPAK, as illustrated by Fig.3. From this figure, we can see that all robots and users have been defined in SPAK including their features. Each frame has the structure as Table 1 and 2 and all of them form a frame hierarchy. Fig.4 shows the contents of frame "MELFA" which is to initiate the MELFA before operating it and Fig.5 shows the contents of frame "MELFAOn", by which the operation of turning on the power of MELFA will be executed. .
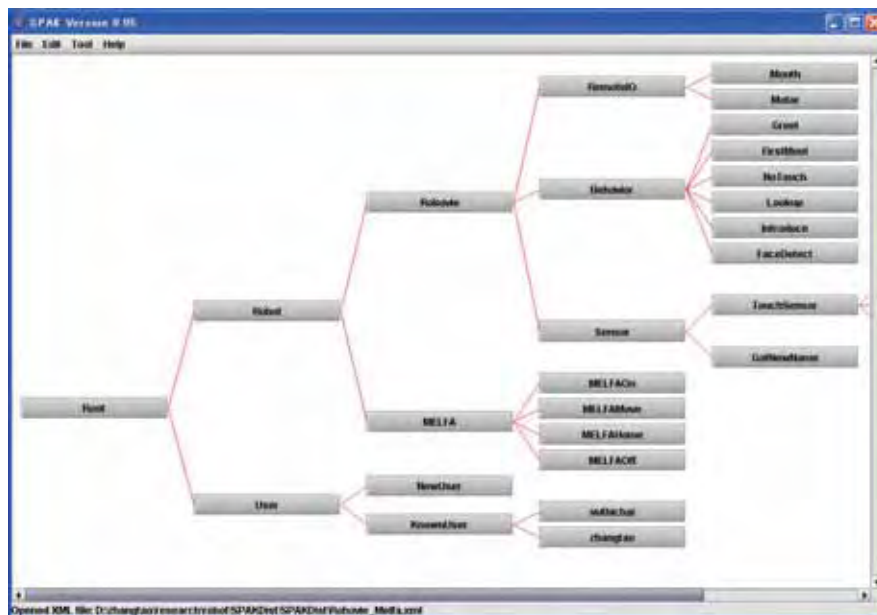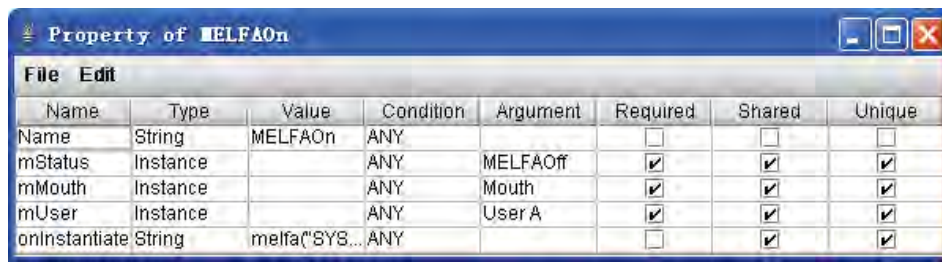
Figure 3. Interface of SPAK

Figure 4. SPAK slot editing table for frame "MELFA"

Figure 5. SPAK slot editing table for frame "MELFAOn"

In Fig.6, for understanding the principle of executing frames in SPAK, the detail contents and relations of the frames of MELFA in SPAK are illustrated as an example. There exist four frames for four kinds of atomic actions of MELFA, such as "MELFAOn", "MELFAMove", "MELFAHome" and "MELFAOff". Each of them is related with the conditions expressed by the frames of "Greet", "Mouth", users and atomic actions. If the frames for the conditions are actuated at the same time, the frame with these condition frames will be actuated and the command given in the special slot "onInstantiate" in this frame will be sent to the servo controller of MELFA. MELFA will then move by the control of servo controller. For instance, the actuation of the frame "MELFA-Move" is based on the actuation of the frames of "Greet", "Mouth" and "MELFAOn". With this system, we have made experiment according to the following scenario. It describes an episode of the experiment.

Robovie:	(Robovie looks at the operator's face and try to recognize it.)
Robovie:	*Hi, how are you! I have never seen you before. What is your name?*
Operator:	(The operator types his name XXX.)
Robovie:	*Hi, XXX. Do you want to operate MELFA now?*
Operator:	*Has MELFA been ready*?
Robovie:	*Yes. How do you want MELFA to move*?
Operator:	*I want MELFA to hold workpiece A and move from P_0 to position P_1.*
Robovie:	*O.K.*
MELFA:	(MELFA starts to perform the task given by the operator.)
Robovie:	*MELFA has finished its jobs. If you want it to perform another job, please give your command.*
	...

From the experiment, operator's commands can be correctly sent to SPAK and SPAK can convert them into the relative commands of MELFA according to the knowledge model of the system. Therefore, MELFA can perform the task correctly. Besides, for ensuring the reliability of the operation, there are some measures for safety adopted in this system. Firstly, the consistency check of the operator's sentences is installed in the sentence parser. If the operator's sentence is inconsistent with the operation of MELFA, the parser will ask operator

to give his sentence again. Secondly, SPAK provides some frames for confirmation. Before performing the movement of MELFA, SPAK will instruct Robovie to repeat the task given by operator again. If operator confirms its correctness, MELFA will be controlled to move. If there exist any problems, SPAK will instruct Robovie to tell operator by speech as soon as possible. Thirdly, MELFA also has some measures to protect itself. With this system, the error information generated by MELFA can be also transferred to SPAK and spoken out by Robovie.
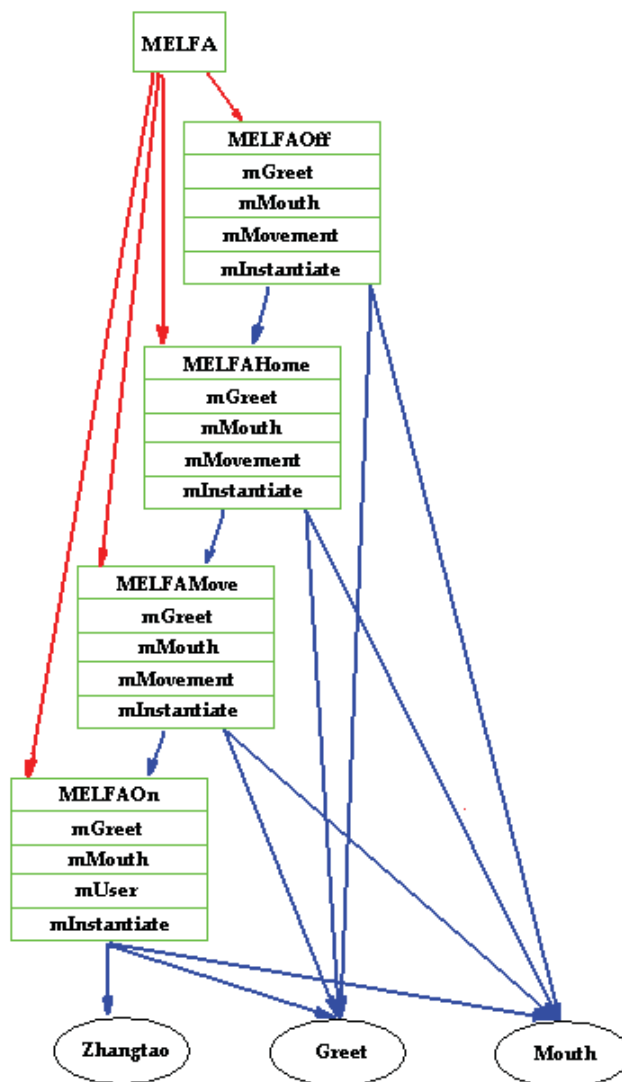
Figure 6. Frames for the operation of MELFA in SPAK

6. Discussions

6.1 Flexibility and extensibility of the proposed human-robot interative control method

Since the proposed human-robot interaction control method is based on the knowledge model and implemented by SPAK, it has strong flexibility and extensibility. The flexibility of the proposed method is reflected by that, in the knowledge model it is possible to define various kinds of frames to control robot behaviors for different tasks based on human requests. SPAK can integrate many software components to support the implementation of human-robot interaction control for industrial robot arms. The proposed method separates the human-robot interaction control of robots into two stages: designing the knowledge model and implementation in the SPAK. It is possible to adopt same ways (same robots) to implement different tasks or adopt different ways (different robots) to implement same tasks. Therefore, high-level knowledge model design or low-level software component development in SPAK provides a large space to flexibly that performs the tasks for human requests.

The extensibility of the proposed method is resulted from the following several points of views. This method is open to many types of robots. Any new types of robots can be described by the frames in the knowledge model and integrated into the SPAK. The behaviors of multiple robots can be easily combined for a specific task. Different combinations can extend the ability of the system to serve to human. Except natural language, many types of human-robot interaction can be integrated by the proposed method. With the increase of the complexity of tasks, the control performance of the proposed method can be improved by extending the functions described in the knowledge model and implemented by SPAK.

6.2 Knowledge management and improvement by learning in SPAK

Due to the increase of the knowledge required for human-robot system, the knowledge management and improvement becomes more and more important. This is the current research topic for improving the proposed method and realizing the goal of constructing a highly intelligent symbiotic human-robot system in the future. The knowledge management includes the organization and optimization of knowledge. Although a frame and its slots provides strong ability to describe various types of objects and frame hierarchy can organize the frames into a system, it needs mechanism to classify the different types of frames, renew the contents of frames, delete the useless frames, detect the logical errors among frames, and so on. Currently, an expert manages the knowledge manually for a human-robot system. It is definitely needed to de-

velop functions to automatically fulfill these kinds of knowledge management. To imitate human ability, learning new knowledge by means of SPAK for improving the knowledge model is being developed in our research. At present, the learning function is implemented as a Java-based program and developed based on the interaction oriented learning approach. It can exchange information with SPAK and create new frames in SPAK. During the process of interaction, a user will teach a communication robot how to perform a robot behavior or accomplish a specific task by various behaviors for different robots. According to user's teaching, new strategy frames can be created in SPAK by the learning function. The learning function performs a searching process in SPAK. New strategy frames are created in SPAK when the relative behavior frames or strategy frames for robot behaviors are found. The searching route is according to the types of robots and their titles of frames. Since SPAK is a Java-based software system and it consists of Java classes to deal with the manipulation of frames, it is therefore easy to create frames and put them into the new strategy frames in SPAK. If SPAK can not find the existing behavior frames or strategy frames, it will ask the user to provide more information about the new behaviors of robots and therefore create new frames in SPAK. Since this learning function is still under development, it will be continuously improved by the further research.

6.3 Future issues

In order to realize the final goal to construct a highly intelligent symbiotic human-robot system, concerning the proposed control method there still exist many attractive, challengeable issues needed to solve. As mentioned above, the knowledge model adopted by the proposed method needs to be improved on its management. We want to combine the proposed method with other kinds of control approaches to improve the control performance for a symbiotic human-robot system, such as adaptive control, robust control, fuzzy control, etc. In addition, the reliability of the proposed human-robot interaction control of robots is needed to continuously improve. We will apply this method for more different typical tasks in wide areas. With the improvement of robot functions in the near future, the proposed method will be able to adopt for more real applications.

7. Conclusions

A new human-robot interaction control approach for industrial robot arm was proposed. In this method, operator's requests can be directly obtained by an autonomous communication robot through human-robot interaction, and converted into control instructions for industrial robot arms by SPAK. In addition, the status of the robot arm can be obtained by SPAK and informed to the operator by the speech function of the communication robot. Based on this method, an experimental symbiotic human-robot system for industrial robot arms was constructed and evaluated. The experimental results demonstrate the effectiveness of the proposed method. The proposed idea of human-robot interaction control approach for industrial robot arms has strong flexibility and extensibility. It can be extended to construct many kinds of symbiotic human-robot systems including different kinds of robots. The knowledge management and improvement can be realized by learning functions in SPAK. In the future research, more issues will be considered to solve and the control performance of human-robot system will be further improved. We believe that the symbiotic human-robot system with human-robot interaction control has great potentials for future human society.

8. References

Ampornaramveth, V.; Kiatisevi, P. & Ueno, H. (2004). SPAK: Software Platform for Agents and Knowledge Management in Symbiotic Robotics. *IEICE Trans. Information and Systems*, Vol. E87-D, No. 4, pp. 886-895.

Bruce, B. (1975). Case systems for natural language. *Artificial Intelligence*, Vol. 6, pp. 327-360.

Cengiz, M. C. (2003). Software development for man-machine interface for an industrial robot. *Thesis of master degree.*

Koller, D. & Pfeffer, A. (1998). Probabilistic frame-based systems. *Proc. of the 15th National Conference on AI* (AAAI-98), pp. 580-587.

Konukseven, E. L. & Abidi, A. (2004). Developmen of man machine interface software for an industrial robot. *Proc. of 2004 IEEE Symposium on Virtual Environment, Human-Computer Interfaces and Measurement Systems,* (VECIMS), pp. 49-53.

Minsky, M. (1974). A Framework for representing knowledge. *MIT-AI Laboratory Memo 306.*

Mizukawa, M.; Matsuka H., et al, (2002). ORiN: open robot interface for the network - the standard and unified network interface for industrial robot applications. *Proceedings of the 41st SICE Annual Conference*, Vol. 2, pp. 925-928.

Sales, J.; Fernandez, R., et al, (2004). Telecontrol of an industrial robot arm by means of a multimodal user interface: a case study. *Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 1, pp. 76-81.

Tairyou, G. (1998). Development of frame-based knowledge engineering environment ZERO. *Master Thesis*, Tokyo Denki University.

Tao Zhang, H. Ueno (2005), A Frame-Based Knowledge Model for Heterogeneous Multi-Robot System, IEEJ Trans. EIS, Vol.125, No.6, pp.846-855.

Ueno, H. (2002). A knowledge-based information modeling for autonomous humanoid service robot. *IEICE Transactions on Information and systems*, Vol. E85-D, No. 4, pp. 657-665.

Spatial Vision-Based Control
of High-Speed Robot Arms

Friedrich Lange and Gerd Hirzinger

1. Introduction

Industrial robots are known to execute given programs at high speed and at the same time with high repeatability. From non industrial scenarios as in (Nakabo et al., 2005) or (Ginhoux et al., 2004) we know that cameras can be used to execute high-speed motion even in those cases in which the desired path is not a priori given but online sensed. In general, such visual tracking tasks of following a randomly moving target by a camera are not accurate enough for industrial applications. But there the work-piece will scarcely move randomly. So in this chapter we concentrate on another type of visual servoing, in which the path that has to be followed is fixed in advance, but not given by a robot program.

A robot-mounted camera is a universal sensor which is able to sense poses with high accuracy of typically less than a millimeter in the close-up range. At the same time with high shutter speed the robot may move fast, e.g. at 1 m/s. In contrast to expensive high-speed cameras, yielding a high frame rate of e.g. 1 kHz as in (Nakabo et al., 2005), we restrict on a standard CCIR camera, to meet the requirements of a cost-effective hardware. Off-the-shelf cameras are fundamental for the acceptance in industry. So an important feature of our method is an appropriate control architecture that tolerates low sampling rates of sensors. The camera is mounted at the robot arm and measures the work-piece pose (given by boundary lines) with respect to the tool center point (tcp) of the robot. More precisely, the camera is mounted laterally to provide enough space for a tool. So with constant moving sense we have a predictive sensor as in (Meta-Scout, 2006). With alternating moving sense the camera has to be tilted so that the corresponding nominal line point $\mathbf{p}_n$ comes approximately to the center of the image (see Fig. 1d). In this case segments of the lines might be occluded by the tool. A tilted mounting yields a priori unknown distances of the different line points and thus complicates the equations. As well, the task frame defined by the tcp, and the camera frame are different.

In contrast to current research (Comport et al., 2005), we assume that problems with image processing, feature detection and projections are solved, which

holds for our simple black and white scenario. In addition, the initial configuration is supposed to be close to the target configuration, so that large rotations as in (Tahri and Chaumette, 2005) do not appear.
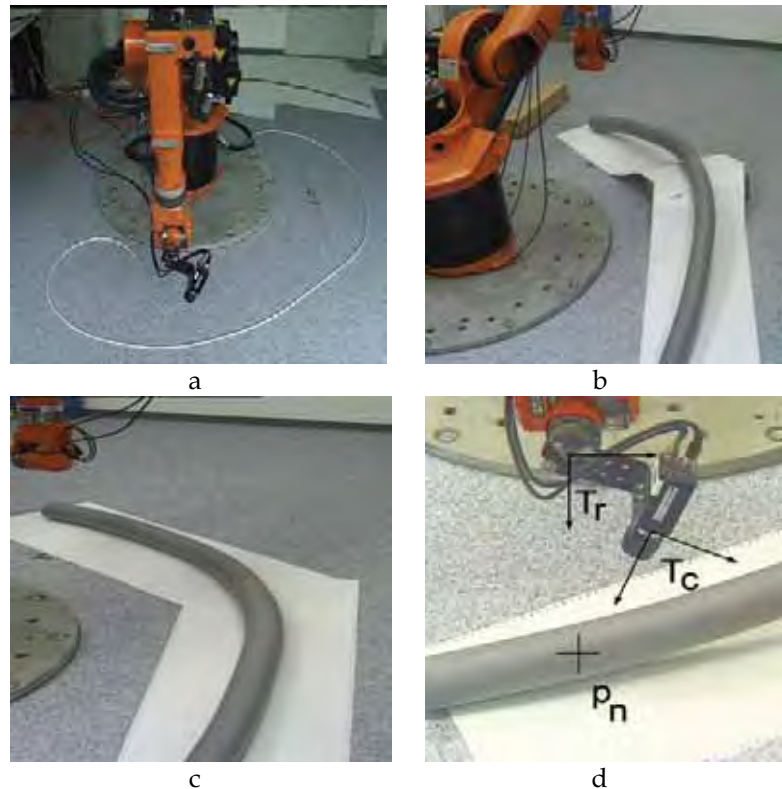
Figure 1. Sample tasks with natural and artificial background and tilted mounted camera

We show several planar and spatial sample tasks (Fig. 1). They are used to sense (curved) lines or edges which are coarsely parallel to the desired motion. This allows to *refine* a coarsely programmed path at full speed.

According to the notation of this chapter the programmed path is called *reference path* and the corresponding lines of the nominal scenario are *nominal lines*. In reality the *sensed lines* will differ from the nominal lines, thus defining the *desired path*, where a path is given by trajectories of positions and orientations.

In our experiments we program a horizontal circular path which is online modified in radial and in vertical direction using image data. A possible application is the spraying of glue to flexible or imprecisely positioned work-pieces. Other applications are laser cutting, or soldering. In these cases the desired path is determined during motion, e.g. by surveying of the boundary lines of the work-piece. For all these tasks high accuracy at high speed is required.

Similar approaches have been proposed predominantly with simple scenarios, see e.g. (Meta-Scout, 2006) or (Lange & Hirzinger, 2002). Other methods handle with multiple lines with point-by-point given nominal location, thus allowing both translational and rotational sensor induced path corrections. However they require complex computations as (Lange & Hirzinger, 2003), or work at lower speed as (Gangloff & de Mathelin, 2002) or (Rives & Borrelly, 2000). The complexity comes from rotations between nominal and real world which in practice are not significant. Therefore we now present a method which denies time consuming iterations to solve systems of trigonometric equations.

The chapter is organized as follows: In section 2 we present a control concept that allows different sampling rates for the robot controller and the sensor. This leads to a universal position controller (section 3) and a task-dependent computation of the desired path (section 4). If the sensible lines yield a non continuous or even a non contiguous path, a smoothing method is required to compute an executable desired path. This is explicated in section 5. Experimental results of continuous paths and from the tracking of lines with vertices are then demonstrated in section 6.

2. Control concept

Instead of directly using sensor data to control the robot, we use a predictive architecture which separates position control from the sensing of the desired path (see Fig. 2). Position control provides what we call a Cartesian ideal robot. This means that for all sampling steps its arm position $\mathbf{x}_a$ is identical to the desired pose $\mathbf{x}_d$. The realization of such an ideal robot will be explained in the sequel. Sensor data affect the system by a module that computes the desired poses at the sampling steps. This is presented in section 4.
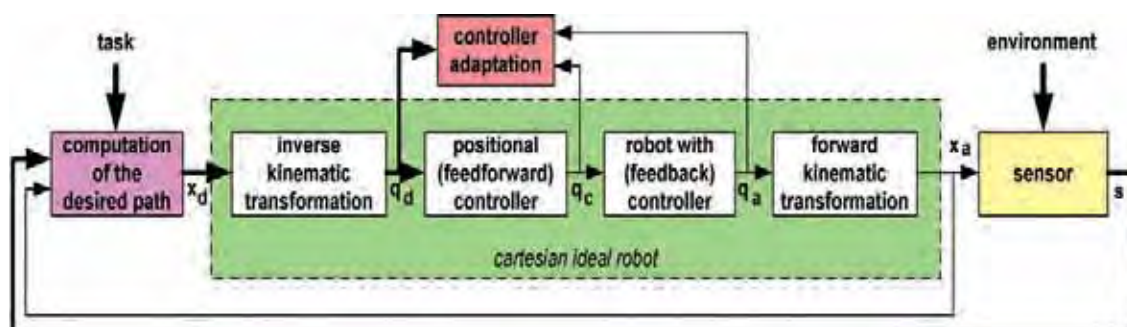
Figure 2. Architecture of control

The fusion of the two parts, position control and the computation of the desired path, yields control from image data.

This concept allows different sampling rates of the camera and the robot controller. Our approach is to integrate image data in each case in the next positional sampling step. As in (Zhang et al., 2003), time delays owing to processor load are tolerable as long as the instant of exposure is known.

3. Position control

The core of this chapter is not restricted to a special position control method. A standard industrial position controller will do if the industrial interface allows online modified references. Such an interface, usually called sensor interface, is required since the desired positions $\mathbf{x}_d$ are not assumed to be available long before the motion.

A standard sensor interface as (Pertin & Bonnet des Tuves, 2004) allows to send online desired values and to receive present positional values. For the purpose of this chapter we refer to the desired values of the sensor interface as a *command* vector $\mathbf{q}_c$, to distinguish between *desired positions* $\mathbf{q}_d$ (of the interface of the ideal robot) and the input to the industrial controller. The actual joint values are called *arm position* $\mathbf{q}_a$.

For curved paths to be executed at high speed we recommend to increase accuracy using advanced control methods which process not only the current desired pose but also the desired speed and acceleration or - what we use - the desired poses of multiple future sampling steps as (Clarke et al., 1987) or (Grotjahn & Heimann, 2002).

Fig. 2 shows an adaptive feed-forward controller as in (Lange & Hirzinger, 1996) as an add-on to the standard industrial feedback controller. A possible controller equation could be

$$\mathbf{q}_c(k) = \mathbf{q}_d(k) + \sum_{i=0}^{n_d} \mathbf{K}_{qi} \cdot (\mathbf{q}_d(k+i) - \mathbf{q}_d(k)) \tag{1}$$

where $\mathbf{K}_{qi}$ stands for the adapted parameters. This controller can be seen as a filter of the desired positions. This proposed controller works in joint space since there are substantial couplings when considering robot dynamics in Cartesian space. In joint space the estimation of couplings can be restricted to some additional adaptive parameters.

This feed-forward controller uses the special predictive interface from (Lange & Hirzinger, 1996), defined either in joint space or in Cartesian coordinates: In

every sampling step the desired poses of the next n_d sampling steps are re-quired, with n_d corresponding to twice the time constant of the controlled ro-bot. This is a key aspect for the realization of an ideal robot.

Bold face lines in Fig. 2 represent data for current and future time steps, i.e. predictions of sensor data are required.

In the case of time-invariant environments, using predictions of the progres-sion of the desired path enables the position controller to compensate all dy-namical delays that otherwise would directly affect performance. The uncer-tainty becomes small enough so that the robot can track a continuous line with very little path error.

4. Computation of the desired path

The task of this computation is to determine the desired poses of next n_d sam-pling steps. It is advantageous to use a reference path with respect to which nominal lines are defined. Then, the task is to find the path which lies with re-spect to the real (sensed) lines, in the same way as the reference path to the nominal lines. This path is called *desired path* and is to be transferred to the ideal robot.

Thus with a camera as sensor and the particular task of tracking lines, predic-tion of the next n_d sampling steps of the desired path reduces to measuring the course of the contour and to keep the values that correspond to future time steps.

For each time-step we compute the transformation matrix ${}^r\mathbf{T}_d$ which describes the frame of a point of the *desired path* $\mathbf{T}_d$ with respect to the frame of the cor-responding point of the *reference path* $\mathbf{T}_r$. The so called *sensor correction* ${}^r\mathbf{T}_d$ is computed from the postulation that the *desired path* is defined by the actually *sensed line* $\mathbf{p}_s$ in the same way as the *reference path* is given by the *nominal line* points $\mathbf{p}_n$. $\mathbf{p}$ is a point vector and, as $\mathbf{T}$, expressed in homogeneous coordi-nates. All these variables depend on time. The time index k is omitted, how-ever, in order to simplify the notation.

This gives the fundamental equation

$$ {}^d\mathbf{p}_s = {}^r\mathbf{p}_n \tag{2} $$

and then

$$ {}^r\mathbf{p}_s = {}^r\mathbf{T}_d \cdot {}^d\mathbf{p}_s = {}^r\mathbf{T}_d \cdot {}^r\mathbf{p}_n. \tag{3} $$

Neglecting rotational sensor corrections yields

$$^r\mathbf{p}_s = \,^r\mathbf{p}_d + \,^r\mathbf{p}_n.\tag{4}$$

If we have at least two lines and their nominal line positions $^r\mathbf{p}_{n_i}$ we can compute the components of $^r\mathbf{p}_d$, namely $^r x_d$ and $^r z_d$ if y is understood as the direction of motion. $^r y_d$ is defined to be zero since we evaluate lines points which are in the x-z-plane of the reference system. Because of the neglected rotational sensor corrections this means also

$$^r y_s = 0 = \,^d y_s.\tag{5}$$

In the image the lines are detected by single points. Here we assume that the camera is oriented such that lines are approximately vertical in the image (see Fig. 3). We use a simple edge detection algorithm which evaluates a single image row. Horizontal line searching gives the best accuracy since it allows processing of single image fields. The detected line points are corrected by a rectification algorithm using previously identified camera and lens distortion parameters.
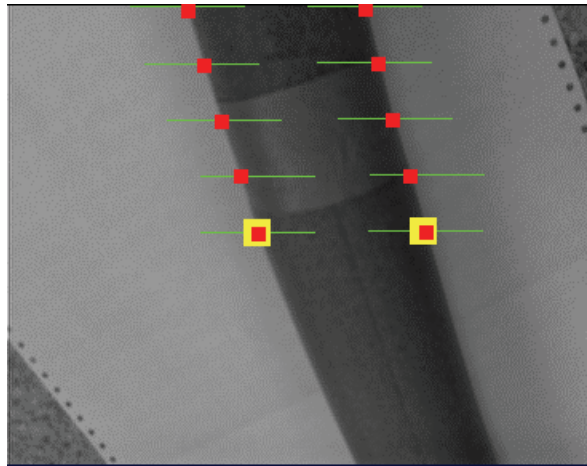
Figure 3. View from the robot mounted camera with 2 lines and 5 sensed points each. The desired positions of the lines[1] in the image are marked with big yellow blocks. The windows for line search are shown by horizontal lines.

---

[1] The desired position of a line in the image is the image point of the current nominal line point when the tcp pose of the exposure is the reference pose. If these desired line positions coincide in the image with the sensed lines, the actual pose complies with the desired path.

At least for curved paths the sensed line points will not share image row with the corresponding nominal image points. So horizontal search starting at the expected image position of a time step will give image points corresponding to the reference of different time instants. Therefore we represent the lines in the image by polynomials. All future computations are then based on these *line polynomials*

$$\xi = f(\eta) \tag{6}$$

with ξ und η as horizontal and vertical normalized image coordinates of a line point.

Using the projection equation

$$^c\mathbf{p}_s = \begin{pmatrix} \xi \cdot {}^c z_s & \eta \cdot {}^c z_s & {}^c z_s & 1 \end{pmatrix}^T, \tag{7}$$

where $^c z_s$ is the distance between the camera and the line point, we get the pose vector $^r\mathbf{p}_s$ from the reference pose of the tcp to a sensed line point

$$^r\mathbf{p}_s = \begin{pmatrix} ^r x_s \\ ^r y_s \\ ^r z_s \\ 1 \end{pmatrix} = {}^r\mathbf{T}_c \begin{pmatrix} \xi \cdot {}^c z_s \\ \eta \cdot {}^c z_s \\ {}^c z_s \\ 1 \end{pmatrix}. \tag{8}$$

With equation (5) we get

$$0 = {}^r y_s = \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix} \cdot {}^r\mathbf{T}_c \cdot {}^c\mathbf{p}_s = ({}^r\mathbf{T}_{c10} \cdot \xi + {}^r\mathbf{T}_{c11} \cdot \eta + {}^r\mathbf{T}_{c12}) \cdot {}^c z_s + {}^r\mathbf{T}_{c13}, \tag{9}$$

where for example $^r\mathbf{T}_{c10}$ is the component (1,0) of the transformation matrix $^r\mathbf{T}_c$. This transformation matrix expresses the actual pose of the camera with respect to the reference pose of the tcp. At least for curved reference paths or a camera which is tilted with respect to the tool or the lines (see Fig. 1d) this transformation differs from identity.

The equations for $^r x_s$ and $^r z_s$ are a little bit more complicated to evaluate. With $\mathbf{T}_r$ and $\mathbf{T}_d$ having the same orientation, using equation (4) and (2) we can write

$$^r x_s = {}^r x_d + {}^d x_s = {}^r x_d + {}^r x_n, \tag{10}$$

where $^r x_n$ is the nominal distance of the line with respect to the reference trajectory of the tcp. This distance is given, e.g. as the space between the desired cutting edge and a visible edge on the work-piece. $^r x_d$ is the wanted path correction which is the same for different lines:

$$^r x_d = (^r \mathbf{T}_{c00} \cdot \xi + {}^r \mathbf{T}_{c01} \cdot \eta + {}^r \mathbf{T}_{c02}) \cdot {}^c z_s + {}^r \mathbf{T}_{c03} - {}^r x_n.$$ (11)

By comparing $^r x_d$ of two lines we get

$$(^r \mathbf{T}_{c00} \cdot \xi_0 + {}^r \mathbf{T}_{c01} \cdot \eta_0 + {}^r \mathbf{T}_{c02}) \cdot {}^c z_{s_0} + {}^r \mathbf{T}_{c03} - {}^r x_{n_0}$$
$$=$$ (12)
$$(^r \mathbf{T}_{c00} \cdot \xi_1 + {}^r \mathbf{T}_{c01} \cdot \eta_1 + {}^r \mathbf{T}_{c02}) \cdot {}^c z_{s_1} + {}^r \mathbf{T}_{c03} - {}^r x_{n_1}.$$

Likewise we compute

$$^r z_d = (^r \mathbf{T}_{c20} \cdot \xi + {}^r \mathbf{T}_{c21} \cdot \eta + {}^r \mathbf{T}_{c22}) \cdot {}^c z_s + {}^r \mathbf{T}_{c23} - {}^r z_n$$ (13)

and thus

$$(^r \mathbf{T}_{c20} \cdot \xi_0 + {}^r \mathbf{T}_{c21} \cdot \eta_0 + {}^r \mathbf{T}_{c22}) \cdot {}^c z_{s_0} + {}^r \mathbf{T}_{c23} - {}^r z_{n_0}$$
$$=$$ (14)
$$(^r \mathbf{T}_{c20} \cdot \xi_1 + {}^r \mathbf{T}_{c21} \cdot \eta_1 + {}^r \mathbf{T}_{c22}) \cdot {}^c z_{s_1} + {}^r \mathbf{T}_{c23} - {}^r z_{n_1}.$$

where $^r z_{n_0}$ and $^r z_{n_1}$ are the nominal distances to the lines, as e.g. the distance between the laser and the work-piece. Usually the distances to the two lines are the same.

In the case of two lines, with two equations (9), equation (12), equation (14), and two equations (6) we have a total of six equations to determine $\xi_0, \xi_1, \eta_0, \eta_1, {}^c z_{s_0}$, and $^c z_{s_1}$. Because of the nonlinearity, equation (6), a numerical solution is required which usually converges within 2 iterations. The wanted components $^r x_d$ and $^r z_d$ of the path correction are calculated by inserting the computed variables into the equations (11) and (13), using any of the lines.

If only one line is visible, we need a priori information, e.g. the distance of the line or, more precisely, the plane in which the line lies. If this plane is parallel to the x-y-plane of the tool frame we can use $^r z_d = 0$. In this case the system of equations is limited to equations (6), (9), and (13) to determine ξ, η, and $^c z_s$ which are inserted into equation (11).

Strictly speaking we use a priori information as well with two lines. It is the assumption that the plane of the lines is parallel to the x-y-plane of the tool. A varying distance can be interpreted as a non zero roll angle. So the specification is restricted to the pitch angle of the plane, i.e. the rotation around the y-axis of the reference frame of the tcp. The computation of this value needs at least three lines, see (Gangloff and de Mathelin, 2002).

For every capture we compute path corrections for multiple reference poses $\mathbf{T}_r$ but fixed camera pose $\mathbf{T}_c$ which is the pose at the time instant of the exposure. To avoid the computation for all n_d sampling steps (see section 3), we represent the path corrections also as polynomials, using again parameter estimation methods. The resulting polynomials allow to readout the wanted path modifications with minimal effort. Therefore the method is still suitable for $n_d \approx 20$. With an appropriate feed-forward control it yields minimal path errors.

The indirect computation of the desired poses by using path modifications with respect to a reference path is advantageous since curved paths with varying orientation are allowed. Solely the path correction itself is assumed to be done without rotations.

5. Computation of a smooth desired path

A problem may occur with the presented method if a line has to be followed that is not as continuous as the lines of Fig. 1. With the path of Fig. 4 the robot tries to execute a velocity step at the vertices between straight edges without considering acceleration limitations.

To prevent this case, we apply impedance-based control. A filter smoothes the sensed edge so that the resulting contour can be tracked.

Figure 4. Sample task with vertices within the line

Regarding Fig. 5 this means that instead of the sensed edge (red points) a smoothed edge (blue points) is used to compute the desired path for the inner position control loop. In force control scenarios such a filter is well known and is called impedance filter. We adopt this expression although in this case sensor data do not represent forces.
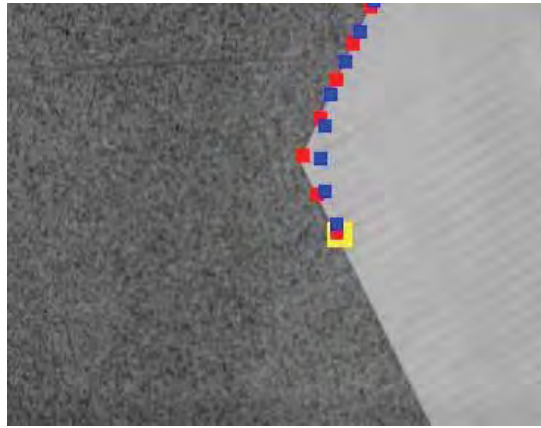
Figure 5. View from the robot-mounted camera. Sensed edge points (red) and the computed smoothed path (blue) are depicted. The yellow block visualizes the current desired pose.

It is worth noting that this kind of impedance-based control does not affect the control law but only the desired trajectory. The stiff position control is maintained. This is useful if different compliances are desired in the individual components of the Cartesian pose vector. A typical example is a horizontal motion where the height of the end-effector with respect to the floor has to be accurately kept while the horizontal components are sensor controlled and therefore have to be compliant with respect to vertices of the target line.

5.1 Impedance-based control

For reasons of clarity we now restrict on Cartesian positions $\mathbf{x}$ instead of positions and orientations as in section 4. So vectors are used instead of matrices of homogeneous coordinates, and therefore the orientation is not affected by the impedance-based control.

It is assumed further that there is a reference trajectory, the originally programmed motion $\mathbf{x}_r(k)$. With respect to this motion the impedance law

$$\mathbf{E} \cdot {}^r\mathbf{x}_d + \mathbf{D} \cdot {}^r\dot{\mathbf{x}}_d + \mathbf{M} \cdot {}^r\ddot{\mathbf{x}}_d = {}^r\mathbf{x}_a + {}^a\mathbf{s} - \mathbf{s}_r = \Delta\mathbf{s}_r \qquad (15)$$

defines the desired trajectory ${}^r\mathbf{x}_d$ where $\mathbf{x}_a$ is the actual pose of the tcp. ${}^a\mathbf{s}$ are the sensed edge data, expressed with respect to the tcp. We here assume that the transformation from the camera system to the tcp system is known. $\mathbf{s}_r$ represents a reference sensor value, i.e. a specified distance between tcp and edge. Fig. 6 demonstrates this setup.
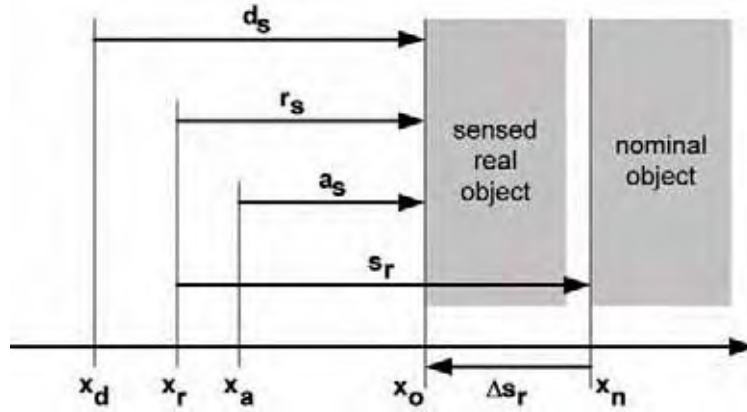
Figure 6. Robot tool positions and sensed distances to a fixed object

Note that within this section the index $*_d$ represents filtered desired values which are usually different from ${}^r\mathbf{T}_d$ or ${}^r\mathbf{p}_d$ in equations like (3) or (4). The latter will now be denoted by $\mathbf{x}_a + {}^a\mathbf{s} - \mathbf{s}_r$ where $\mathbf{x}_a + {}^a\mathbf{s}$ corresponds to an object pose $\mathbf{T}_o$ and $-\mathbf{s}_r$ to ${}^o\mathbf{T}_d$.

With $\mathbf{E} = \mathbf{I}$ and $\mathbf{D} = \mathbf{M} = \mathbf{0}$, (15) represents explicit sensor-based control as in section 4. If, in contrast, we specify $\mathbf{M} > \mathbf{0}$, the resulting trajectory will smooth the corners since then the accelerations are weighed in addition to the deviations from the trajectory of explicit sensor-based control. With $\mathbf{M} > \mathbf{0}$ we further specify $\mathbf{D} > \mathbf{0}$ in order to avoid oscillations.

With ${}^r\dot{\mathbf{x}}_d(k) = ({}^r\mathbf{x}_d(k+1) - {}^r\mathbf{x}_d(k-1))/2T_0$
and ${}^r\ddot{\mathbf{x}}_d(k) = ({}^r\mathbf{x}_d(k+1) - 2\,{}^r\mathbf{x}_d(k) + {}^r\mathbf{x}_d(k-1))/T_0^2$
we compute ${}^r\mathbf{x}_d(k+i)$ from images taken at the time step k by

$$\mathbf{E} \cdot {}^r\mathbf{x}_d(k+i) + \frac{\mathbf{D}}{2T_0} \cdot \left({}^r\mathbf{x}_d(k+i+1) - {}^r\mathbf{x}_d(k+i-1) \right)$$

$$\tag{16}$$

$$+ \frac{\mathbf{M}}{T_0^2} \cdot \left({}^r\mathbf{x}_d(k+i+1) - 2\,{}^r\mathbf{x}_d(k+i) + {}^r\mathbf{x}_d(k+i-1) \right) = {}^r\mathbf{x}_a(k) + {}^a\mathbf{s}(k+i/k) - \mathbf{s}_r .$$

$^a\mathbf{s}(k+i/k)$ reflects the sensed edge position for time step $(k+i)$, sensed in the image at time step k. Equation (16) gives a system of equations for $^y x_d(k+i)$ with given k. So the solution computes the desired trajectory from the actual position and the sensor values at time step k.

The result can be seen considering a vertex or a discontinuity between $^a\mathbf{s}(k+i/k)$ and $^a\mathbf{s}(k+i+1/k)$. With $\mathbf{D}=\mathbf{M}=\mathbf{0}$ a vertex or a discontinuity of $^r\mathbf{x}_d(k+i)$ would be reproduced. In contrast, $\mathbf{M}>\mathbf{0}$ will smooth the trajectory. Smoothing is not restricted by causality. Instead, the desired trajectory is affected far before the discontinuity of $^a\mathbf{s}(k+i/k)$. It is still affected after the discontinuity, as well.

Equation (15) defines the compliance between the reference trajectory and the new desired position. The same compliance is valid for the more intuitive relation between the sensed edge and the desired system since can be derived in the same way as (15). The right hand side expression is independent of the resulting desired robot motion $\mathbf{x}_d$

$$\mathbf{E}\cdot {}^d\mathbf{x}_o + \mathbf{D}\cdot {}^d\dot{\mathbf{x}}_o + \mathbf{M}\cdot {}^d\ddot{\mathbf{x}}_o = \mathbf{s}_r + (\mathbf{E}-\mathbf{I})\cdot {}^r\mathbf{x}_o + \mathbf{D}\cdot {}^r\dot{\mathbf{x}}_o + \mathbf{M}\cdot {}^r\ddot{\mathbf{x}}_o \qquad (17)$$

.

5.2 Modified approach

Experiments with impedance-based control according to Section 5.1 show a time delay between the sensed and the optimized trajectory. This comes from the fact that a discontinuity of the sensed edge does not cause a position error in relation to an acceleration dependent expression and a velocity dependent expression but the sum of all three terms is minimized. So for example a negative acceleration may compensate a positive position error.

Therefore we define three independent equations that have to be minimized altogether for all time steps.

$$\mathbf{E}\cdot {}^r\mathbf{x}_d(k+i) = {}^r\mathbf{x}_a(k) + {}^a\mathbf{s}(k+i/k) - \mathbf{s}_r \qquad (18)$$

$$\frac{\mathbf{D}}{2T_0}\cdot \left({}^r\mathbf{x}_d(k+i+1) - {}^r\mathbf{x}_d(k+i-1) \right) = \mathbf{0} \qquad (19)$$

$$\frac{\mathbf{M}}{T_0^2}\cdot \left({}^r\mathbf{x}_d(k+i+1) - 2\,{}^r\mathbf{x}_d(k+i) + {}^r\mathbf{x}_d(k+i-1) \right) = \mathbf{0} \qquad (20)$$

Minimization is done in a least squares sense where $\mathbf{E}$, $\mathbf{D}$, and $\mathbf{M}$ are the weighting factors.

5.3 Implementation

Instead of solving (16) or minimizing the mean errors of (18) to (20) in every time step, a filter can be computed since we have a linear system of equations. This filter gives ${}^r\mathbf{x}_d(l)$ with $k \leq l \leq k+n_i$ from the values of ${}^r\mathbf{x}_a(k) + {}^a\mathbf{s}(k+i/k)$ with $0 \leq i \leq n_i$. This filter is called impedance filter. Its coefficients are found by a single optimization process in the systems (16) or (18) to (20) respectively.

We now have to specify the number n_i of elements of this filter. To be able to compute n_d time steps of ${}^r\mathbf{x}_d(l)$ we need $n_i \gg n_d$. In practice however, n_i is usually limited by the visible range of the edge. n_i has been chosen sufficiently large if ${}^r\mathbf{x}_d(l)$ proves to be time invariant. This requires that the initial conditions ${}^r\mathbf{x}_d(l)$ with $l < k$ have been computed in the same way. The end conditions ${}^r\mathbf{x}_d(l)$ with $l > k+n_i$ are set to ${}^r\mathbf{x}_d(l) = \Delta\mathbf{s}_r(l)$.

6. Experiments

As first experiments we consider a bent tube that has to be followed at a speed of 0.7 m/s by a KUKA KR6/1 robot (see Fig. 1b or 1c) using the industrial controller KRC1. Image processing, i.e. detection of the boundary lines of the tube, runs in field mode of the camera at a rate of 50 Hz. This is asynchronous to the control rate of 83 Hz.

Previously identified camera reconstruction errors are compensated. E.g. lens distortion is identified according to (CalLab, 1999) and the sensed line points are rearranged accordingly. In addition, some parameters as the constant time shift between the time instant of the exposure and the reception of the image in the computing module are estimated and incorporated thereafter, see (Lange and Hirzinger, 2005) for details.

All tasks for robot control and image processing run in parallel on a 400 MHz processor of the industrial controller KRC1. Control uses the operating system VxWorks and vision runs under Windows95 which is one of the VxWorks tasks. Thus the additionally required hardware is limited to a camera and a standard frame grabber.

Table 1 displays the reached accuracy.

Experiment with	1 line	2 lines
Mean pixel error	1.2 pixel	1.4 pixel
Maximum pixel error	2.9 pixel	4.4 pixel
Mean path error (horizontal, vertical)	0.3 mm	0.3 mm 0.9 mm
Maximum path error (horizontal, vertical)	1.0 mm	1.0 mm 2.4 mm
Mean deviation from reference path	51 mm	37 mm 65 mm
Maximum deviation from reference path	77 mm	78 mm 98 mm

Table 1. Path error when following a bent tube with 0.7 m/s evaluating in the horizontal plane (1 boundary line) or in space (2 boundary lines) respectively

When tracking only one boundary line (Fig. 1c), the tube lies horizontally on the floor. In this case, in spite of big differences between nominal and actual tube, a mean pixel error of less than 1 pixel is reached even without filtering of the edge data.

With a non planar layout (Fig. 1b) we evaluate both boundary lines to determine a spatial path. The vertical accuracy is inferior because on the one hand the measuring of the distance is ill-conditioned due to the geometrical setup, and on the other hand because elastic oscillations of the robot are excited. Compliance in the robot joints prevents accurate measurements of the camera pose, which influences the detected line poses. Nevertheless the mean control error in both degrees of freedom (dof) falls below one millimeter (Fig. 7b and Fig. 7c). The reached path accuracy equals thus the complex method of (Lange & Hirzinger, 2003).

As a second experiment the robot has to follow, as well at 0.7 m/s, a light cable which lies on the ground (see Fig. 1a and Fig. 7a). This demonstrates that, in spite of the structured floor, there is no artificial background required
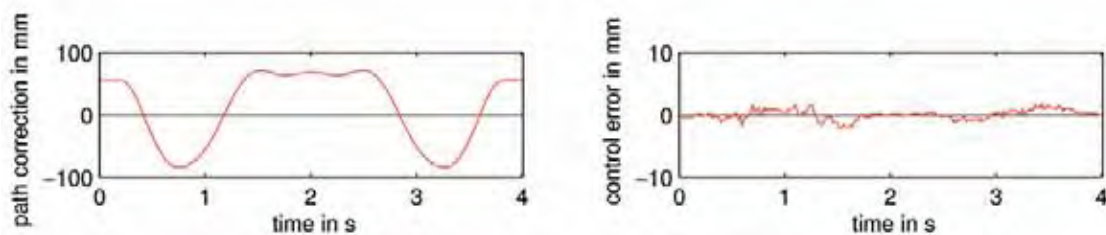
a   Following a cable as in Figure 1a
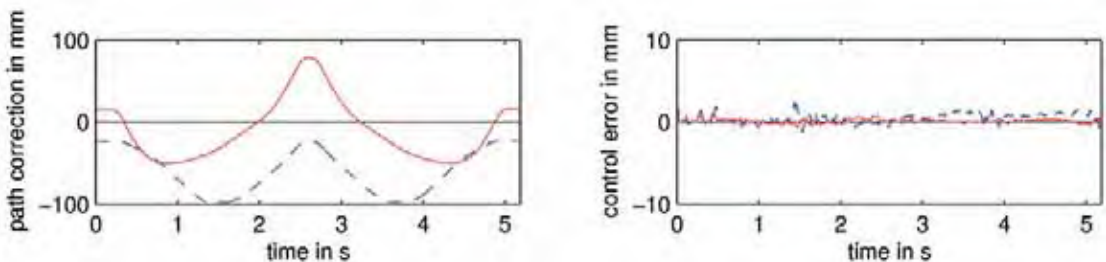
b   Following a tube in space as in Figure 1b (solid = horizontal, dashed = vertical)
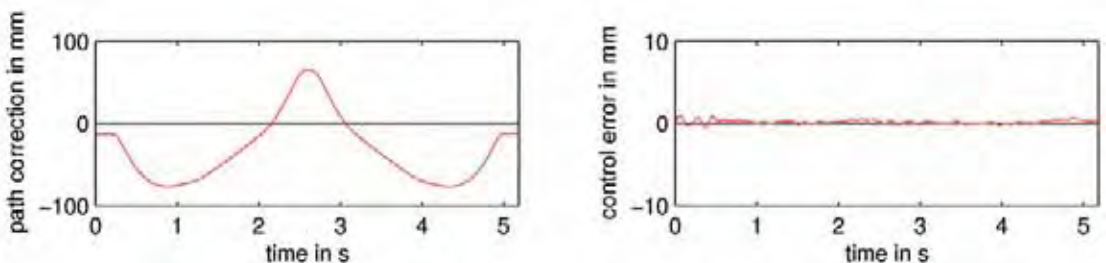
c   Following a planar tube as in Figure 1c

Figure 7. Sensor induced path correction and control error (note the different scales)

Path corrections are horizontal only, and computed by different methods. Because of bigger deviations from the reference path, path errors (see Table 2) are somewhat bigger than in the previous experiment. This is caused by approximation errors between the polynomials and the real shape of the line. Besides, exposure instant uncertainties will cause additional errors.

The errors of the method of this chapter are listed in the right-hand column. They are compared, first, with a method which, using the same control architecture, works without the predictive interface of the position controller, and therefore without feed-forward control. So with this method in the inner control loop, only the standard feedback controller is used. In contrast, the computation of the desired path is unchanged. Depending on the shape of the line, only a small increase of the path errors appears with this setup. This is because although the position control is not predictive, the vision system still uses the upper and lower regions of the image, thus allowing predictions of the desired path.

Control	without use of sensor data	by visual servoing with PD controller without prediction	with position controller without feed-forward with prediction	with position controller with feed-forward with prediction
Mean pixel error	-	35 pixel	1.9 pixel	1.3 pixel
Max. pixel error	-	63 pixel	4.7 pixel	3.6 pixel
Mean path error	95 mm	9.7 mm	0.8 mm	0.6 mm
Max. path error	157 mm	19.9 mm	2.4 mm	1.5 mm

Table 2. Path error when following a cable using different methods

A further alternative is a classical visual servoing algorithm which only evaluates the location of the line in the center of the image, i.e. without any prediction. Control is implemented using a coarsely optimized PD algorithm for the Cartesian signal in x direction of the reference system. This controller ignores the camera positions.

$$
\begin{aligned}
{}^r x_{cx}(k) = {}^r x_{cx}(k-1) \\
+ K_P \cdot {}^r e_x(k) + K_D \cdot ({}^r e_x(k) - {}^r e_x(k-1))
\end{aligned}
\tag{21}
$$

Because of the tilted mounted camera (see Fig. 1d), a control error of

$$
{}^r e_x = ({}^r z_n - {}^a \mathbf{T}_{c23}) \cdot \frac{{}^a \mathbf{T}_{c00} \cdot \xi + {}^a \mathbf{T}_{c01} \cdot \eta + {}^a \mathbf{T}_{c02}}{{}^a \mathbf{T}_{c20} \cdot \xi + {}^a \mathbf{T}_{c21} \cdot \eta + {}^a \mathbf{T}_{c22}} + ({}^a \mathbf{T}_{c03} - {}^r x_n)
\tag{22}
$$

is evaluated. $^a\mathbf{T}_{cij}$ are the elements of the (constant) transformation matrix between tcp and camera. In this experiment the performance is poor, and without limitation[2] the robot would leave the allowed range of accelerations.

For the non continuous layout of the sheets in Fig. 4 we restrict on the modified method of section 5.2 because this method outperforms the other one, except for special cases, as in (Lange et al., 2006), where both methods are applicable. This time the programmed path is a straight line in y-direction, executed back and forth at 0.7 m/s and with $\mathbf{s}_r = 0$. In contrast to the previous experiments, with 600 mm the distance between the camera and the edge is about twice as much as before. This reduces the positional resolution but it enlarges the visible region to about n_i = 25 sampling steps of the controller which is sufficient with $n_d = 15$. Such an extension is useful to fulfil $n_i \gg n_d$.

The results are displayed in Fig. 8, on the left hand side with respect to the reference path and on the right hand side with respect to the sensed edge. With explicit image-based control (Fig. 8a) or small D as in Fig. 8b we result in high accelerations at the vertices in the sheets. On the other side, the edges are tracked as closely as possible. This desired path is accurately executed by the ideal position controlled robot besides a couple of time-steps in which the acceleration limits of the robot are reached. Nevertheless a mean tracking error of about 1 mm is reached.

By specifying $M = 0.001\,\mathrm{s}^2$ and $D = 0.06\,\mathrm{s}$ we define a desired trajectory that leaves the edges but that shows smooth transitions (see Fig. 5). Bigger impedance parameters, as in Fig. 8d are not recommended since then n_i limits the smoothing. Note that the vertex which is halfway on the left hand side diagrams of Fig. 8 is not smoothed because this is the point of reversing.

Videos of the reported and additional experiments can be found in (Lange, 2006).

---

[2] For convenience, accelerations exceeding the limitations are scaled to the feasible values.
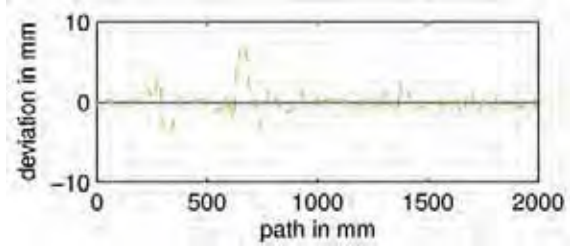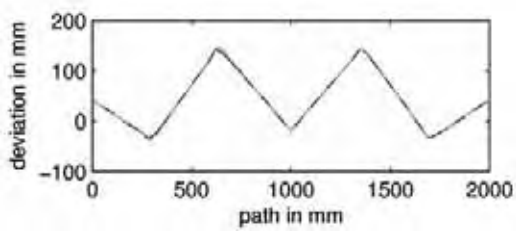
a   Explicit sensor-based control yielding a mean distance to the edge of 1.2 mm
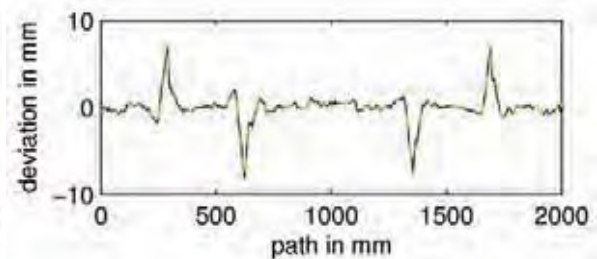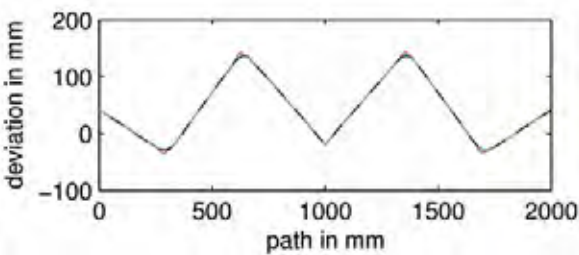
b   Impedance-based sensor-based control with $D = 0.02$ s and $M = 0.001$ s^2
     yielding a   mean distance to the edge of 1.7 mm
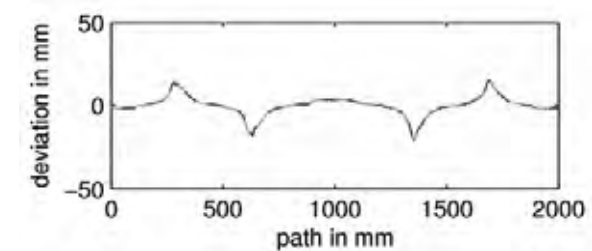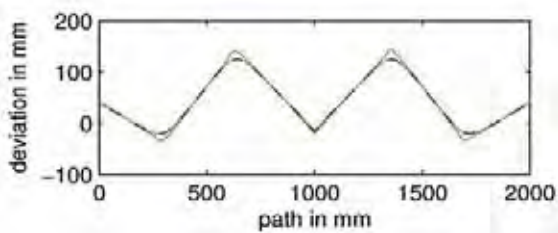
c   Impedance-based sensor-based control with $D = 0.06$ s and $M = 0.001$ s^2
     yielding a mean distance to the edge of 5 mm
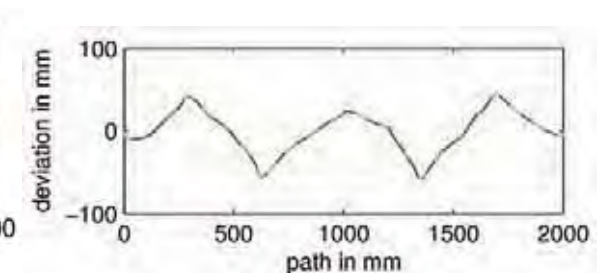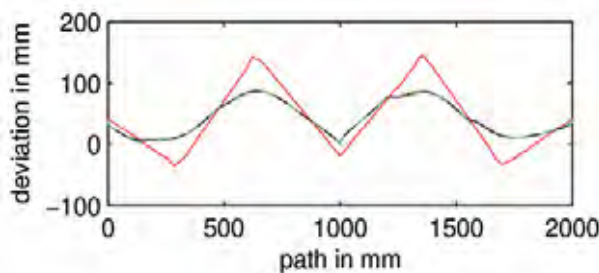
d   Impedance-based sensor-based control with $D = 0.2$ s and $M = 0.001$ s^2
     yielding a mean distance to the edge of 23 mm

Figure 8. Plots of ${}^{r}x_d$ (left) and ${}^{o}x_d$ (right) when tracking the sensed edge (red), back
and forth, using camera-based impedance control. Desired (black) and actual (dashed
green) path are almost identical, besides the experiment without smoothing.

7. Conclusion

The article shows that vision systems are also applicable for tracking tasks with high robot speed. Even accurate control of the robot is possible. The crucial fact is the predictive image evaluation, maybe in combination with an adaptive positional feed-forward control.

The results are reached by means of a cost effective method for which additional hardware is limited to standard components. In order to guarantee low costs, image processing, computation of the desired path and dynamical control are executed using only the standard processor of an industrial robot controller, in our case the KUKA KRC1.

For non continuous or non contiguous paths we propose smoothing similar to impedance control. Then a reduction of the standards of position accuracy is tolerated for specified components. The user has to specify parameters in order to find a compromise between a smooth trajectory and minimum path deviations with respect to the sensed edges. The modified impedance-based method effects that the robot follows the sensed edge as accurate as possible, except for vertices or discontinuities where the path is smoothed. Among the two shown methods the latter it is favourable if a predictive sensor is used.

The methods are demonstrated in tasks where the location and the shape of planar or spatial lines are used to modify the robot path during motion. For continuous lines the mean resulting path error is below 1 mm, whereas for lines with vertices, exact tracking is not executable with a robot. Then the amount of smoothing determines the deviation from the lines.

Future work will improve the measurements of the camera pose e.g. using an observer, thus avoiding oscillations caused by joint elasticity.

8. References

CalLab. http://www.robotic.dlr.de/VISION/Projects/Calibration/CalLab.html, 1999.

Clarke, D. W. , C. Mohtadi, and P. S. Tuff. Generalized predictive control - part I. the basic algorithm. *Automatica*, 23(2):137–148, 1987.

Gangloff, J. A. and M. F. de Mathelin. Visual servoing of a 6-dof manipulator for unknown 3-d profile following. *IEEE Trans. on Robotics and Automation*, 18(4):511–520, August 2002.

Ginhoux, R. et al. Beating heart tracking in robotic surgery using 500 Hz visual servoing, model predictive control and an adaptive observer. In *Proc. 2004 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 274–279, New Orleans, LA, April 2004.

Comport, A. I., D. Kragic, E. Marchand, and F. Chaumette. Robust real-time visual tracking: Comparison, theoretical analysis and performance evaluation. In *Proc. 2005 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2852–2857, Barcelona, Spain, April 2005.

Grotjahn, M. and B. Heimann. Model-based feedforward control in industrial robotics. *The International Journal on Robotics Research*, 21(1):45–60, January 2002.

Lange, F. and G. Hirzinger. Learning of a controller for non-recurring fast movements. *Advanced Robotics*, 10(2):229–244, April 1996.

Lange, F. and G. Hirzinger. Is vision the appropriate sensor for cost oriented automation? In R. Bernhardt and H.-H. Erbe, editors, *Cost Oriented Automation (Low Cost Automation 2001)*, Berlin, Germany, October 2001. Published in IFAC Proceedings, Elsevier Science, 2002.

Lange, F. and G. Hirzinger. Predictive visual tracking of lines by industrial robots. *The International Journal on Robotics Research*, 22(10-11):889–903, Oct-Nov 2003.

Lange, F. and G. Hirzinger. Calibration and synchronization of a robot-mounted camera for fast sensor-based robot motion. In *Proc. 2005 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3911–3916, Barcelona, Spain, April 2005.

Lange, F.. Video clips. http://www.robotic.de/?id=43., 2006.

Lange, F., M. Frommberger, and G. Hirzinger. Is impedance-based control suitable for trajectory smoothing? In *Preprints 8th IFAC Symposium on Robot Control (SYROCO 2006)*, Bologna, Italy, Sept. 2006.

Meta-Scout GmbH. SCOUT joint tracking system. http://www.scout-sensor.com/index-engl.html.

Nakabo, Y., T. Mukai, N. Fujikawa, Y. Takeuchi, and N. Ohnishi. Cooperative object tracking by high-speed binocular head. In *Proc. 2005 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1585–1590, Barcelona, Spain, April 2005.

Pertin, F. and J.-M. Bonnet des Tuves. Real time robot controller abstraction layer. In *Proc. Int. Symposium on Robots (ISR)*, Paris, France, March 2004.

Rives, P. and J.-J. Borrelly. Real-time image processing for image-based visual servoing. In M. Vincze and G. D. Hager, editors, *Robust vision for vision-based control of motion*, pages 99–107. IEEE Press, 2000.

Tahri, O. and F. Chaumette. Complex objects pose estimation based on image moments invariants. In *Proc. 2005 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 438–443, Barcelona, Spain, April 2005.

Zhang, J., R. Lumia, J. Wood, and G. Starr. Delay dependent stability-limits in high performance real-time visual servoing systems. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pages 485–491, Las Vegas, Nevada, Oct. 2003.

Visual Control System for Robotic Welding

De Xu, Min Tan and Yuan Li

1. Introduction

In general, the teaching by showing or offline programming is used for path planning and motion programming for the manipulators. The actions preset are merely repeated in the working process. If the states of work piece varied, the manufacture quality would be influenced too intensely to satisfy the demand of production. In addition, the teaching by showing or offline programming costs much time, especially in the situations that much manufacture variety with little amount. The introduction of visual measurement in robot manufacture system could eliminate the teaching time and ensure the quality even if the state of the work piece were changed. Obviously, visual control can make the robot manufacture system have higher efficiency and better results (Bolmsjo et al., 2002; Wilson, 2002).There are many aspects concerned with the visual control for robotic welding such as vision sensor, image processing, and visual control method.As a kind of contactless seam detecting sensors, structured light vision sensor plays an important role in welding seam tracking. It has two categories. One uses structured light to form a stripe, and the other uses laser scanning. Structured light vision is regarded as one of the most promising methods because of its simplicity, higher accuracy and good performance in real-time (Wu & Chen, 2000). Many researchers pay their attention to it (Bakos et al., 1993; Zou et al., 1995; Haug & Pristrchow, 1998; Zhang & Djordjevich, 1999; Zhu & Qiang, 2000; Xu et al., 2004). For example, Bakos established a structured light measurement system, which measurement precision is 0.1mm when the distance is 500 mm. Meta Company provides many kinds of laser structured light sensors. In general, the sensor should be calibrated before putting into action. Camera calibration is an important classic topic, and a lot of literatures about it can be found (Faugeras & Toscani, 1986; Tsai, 1987; Ma, 1996; Zhang, 2000). But the procedure is complicated and tedious, especially that of the laser plane's calibration (Zhang & Djordjevich, 1999). Another problem in structured light vision is the difficulty of image processing. The structured light image of welding seam is greatly affected by strong arc light, smog and splash in the process of arc welding (Wu & Chen, 2000). Not only the image is rough, but also its background is noisy. These give rise

to difficulty, error and even failure of the processing of the welding seam image. Intelligent recognition algorithms, such as discussed in (Kim et al., 1996; Wu et al., 1996), can effectively eliminate some of the effects. However, besides intelligent recognition algorithm, it is an effective way for the improvement of recognition correctness to increase the performance of image processing.

The visual control methods fall into three categories: position-based, image-based and hybrid method (Hager et al., 1996; Corke & Good, 1996; Chaumette & Malis, 2000). As early as 1994, Yoshimi and Allen gave a system to find and locate the object with "active uncalibrated visual servoing" (Yoshimi & Allen, 1994). Experimental results by Cervera et al. demonstrated that using pixel coordinates is disadvantageous, compared with 3D coordinates estimated from the same pixel data (Cervera et al., 2002). On the other hand, although position-based visual control method such as (Corke & Good, 1993; 1996) has better stableness, it has lower accuracy than former because the errors of kinematics and camera have influence on its precision. Malis et al. proposed hybrid method that controls the translation in image space and rotation in Cartesian space. It has the advantages of two methods above (Malis et al., 1998; 1999; Chaumette & Malis, 2000).

In this chapter, a calibration method for the laser plane is presented, which is easy to be realized and provides the possibility to run hand-eye system calibration automatically. Second, the image processing methods for the laser stripe of welding seam are investigated. Third, a novel hybrid visual servoing control method is proposed for robotic arc welding with a general six degrees of freedom robot.The rest of this chapter is arranged as follows. The principle of a structured light vision sensor is introduced in Section 2. And the robot frames are also assigned in this Section. In Section 3, the laser plane equation of a structured light visual sensor is deduced from a group of rotation, in which the position of the camera's optical centre is kept unchangeable in the world frame. In Section 4, a method to extract feature points based on second order difference is proposed for type V welding seams. A main characteristic line is obtained using Hotelling transform and Hough transform. The feature points in the seam are found according to its second difference. To overcome the reflex problem, an improved method based on geometric centre is presented for multi-pass welding seams in Section 5. The profiles of welding seam grooves are obtained according to the column intensity distribution of the laser stripe image. A gravity centre detection method is provided to extract feature points on the basis of conventional corner detection method. In Section 6, a new hybrid visual control method is concerned. It consists of a position control inner loop in Cartesian space and two outer loops. One outer loop is position-based visual control in Cartesian space for moving in the direction of the welding seam, i.e. welding seam tracking; another is image-based visual control in image space for adjustment to eliminate the errors in tracking. Finally, this chapter is ended with conclusions in Section 7.

2. Structured light vision sensor and robot frame

2.1 Structured light vision sensor

The principle of visual measurement with structured light is shown in Fig. 1. A lens shaped plano-convex cylinder is employed to convert a laser beam to a plane, in order to form a stripe on the welding works. A CCD camera with a light filter is used to capture the stripe. It is a narrow band filter to allow the light in a small range with the centre of laser light wavelength to pass through. It makes the laser stripe image be very clear against the dark background. A laser emitter, a plano-convex cylinder lens, and a camera with a light filter constitute a structured light vision sensor, which is mounted on the end-effector of an arc welding robot to form a hand-eye system. The camera outputs a video signal, which is input to an image capture card installed in a computer. Then the signal is converted to image (Xu et al., 2004a).
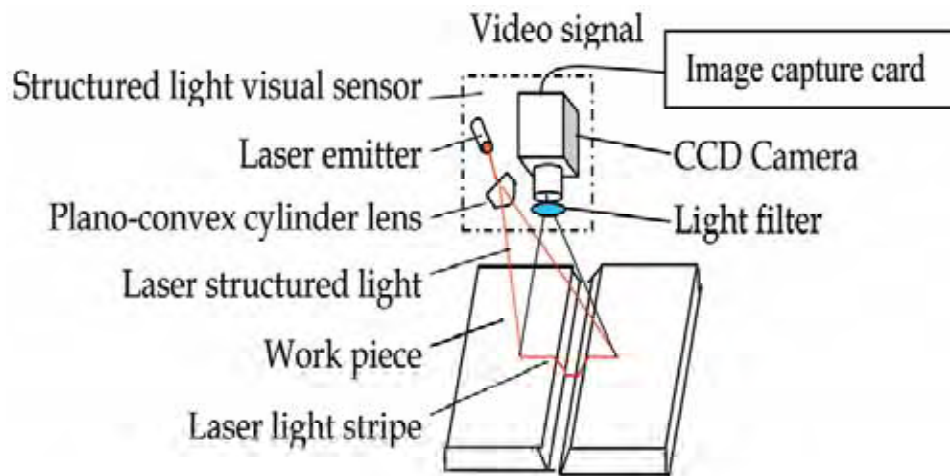
Figure 1. The principle of structured light vision sensor

2.2 Robot frame assignment

Coordinates frames are established as shown in Fig. 2. Frame W represents the original coordinates, i.e. the world frame. Frame E the end-effector coordinates. Frame R the working reference coordinates. Frame C the camera coordinates. The camera frame C is established as follows. Its origin is assigned at the optical centre of the camera. Its z-axis is selected to the direction of the optical axis from the camera to the scene. Its x-axis is selected as horizontal direction of its imaging plane from left to right. $^{w}T_{r}$ indicates the transformation from

frame W to R, i.e. the position and orientation of frame R expressed in frame W. And ^rT_c is from frame R to C, ^wT_e from frame W to E, ^eT_c from frame E to C.
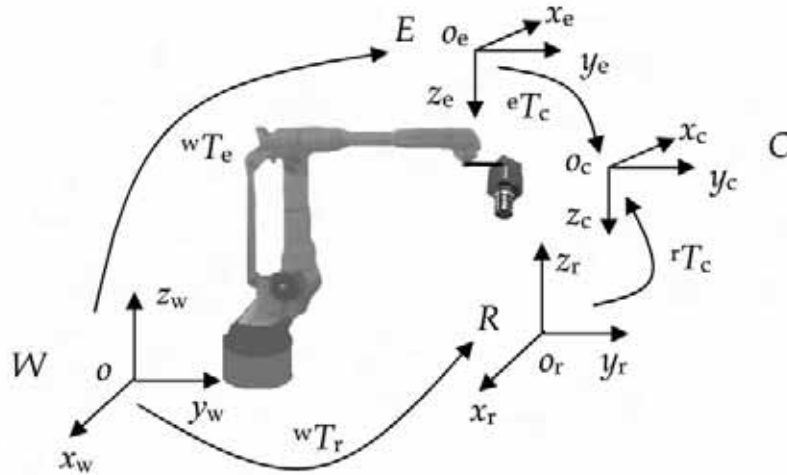
Figure 2. The sketch figure of coordinates and transformation

3. Laser plane calibration

3.1 Calibration method based on rotation

Generally, the camera is with small view angle, and its intrinsic parameters can be described with pinhole model, as given in (1). Its extrinsic parameters can be given in (2).

$$
\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & u_0 \\ 0 & k_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c / z_c \\ y_c / z_c \\ 1 \end{bmatrix} = M_{in} \begin{bmatrix} x_c / z_c \\ y_c / z_c \\ 1 \end{bmatrix}
\tag{1}
$$

where $[u, v]$ are the coordinates of a point in an image, $[u_0, v_0]$ denote the image coordinates of the camera's principal point, $[x_c, y_c, z_c]$ are the coordinates of a point in the camera frame, M_{in} is the intrinsic parameters matrix, and $[k_x, k_y]$ are the magnification coefficients from the imaging plane coordinates to the image coordinates. In fact, $[k_x, k_y]$ are formed with the focal length and the magnification factor from the image size in mm to the imaging coordinates in pixels.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = {}^c M_w \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \qquad (2)$$

where $[x_w, y_w, z_w]$ are the coordinates of a point in the object frame, and ${}^c M_w$ is the extrinsic parameter matrix of the camera, i.e. the transformation from the camera frame C to the world frame W. In ${}^c M_w$, $\bar{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$ is the direction vector of the x-axis, $\bar{o} = \begin{bmatrix} o_x & o_y & o_z \end{bmatrix}^T$ is that of the y-axis, $\bar{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T$ is that of the z-axis for the frame W expressed in the frame C, and $\bar{p} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$ is the position vector.

Camera calibration is not a big problem today. But laser plane calibration is still difficult. Therefore, the calibration of structured light vision sensor is focused on laser plane except camera. In the following discussion (Xu & Tan, 2004), the parameters of a camera are supposed to be well calibrated in advance.

Assume the equation of the laser light plane in frame C is as follows

$$ax + by + cz + 1 = 0 \qquad (3)$$

where a, b, c are the parameters of the laser light plane.

An arbitrary point P in laser stripe must be in the line formed by the lens centre and the imaging point $[x_{c1}, y_{c1}, 1]$. Formula (4) shows the equation of the line in frame C.

$$\begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} x_{c1} & y_{c1} & 1 \end{bmatrix}^T t \qquad (4)$$

where $x_{c1} = x_c / z_c$, $y_{c1} = y_c / z_c$, t is an intermediate variable.

On the other hand, the imaging point $[x_{c1}, y_{c1}, 1]^T$ can be calculated from (1) as follows.

$$\begin{bmatrix} x_{c1} & y_{c1} & 1 \end{bmatrix}^T = M_{in}^{-1} \begin{bmatrix} u & v & 1 \end{bmatrix}^T \qquad (5)$$

From (3) and (4), the coordinates of point P in frame C can be expressed as the functions of parameter a, b, and c, given in (6). Further more, its coordinates $[x_w, y_w, z_w]$ in frame W can be had as given in (7).

$$\begin{cases} x = -x_{c1} /(ax_{c1} + by_{c1} + c) \\ y = -y_{c1} /(ax_{c1} + by_{c1} + c) \\ z = -1 /(ax_{c1} + by_{c1} + c) \end{cases} \tag{6}$$

$$\begin{bmatrix} x_w & y_w & z_w & 1 \end{bmatrix}^T = {}^wT_e \; {}^eT_c \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T \tag{7}$$

Let

$${}^wT_e \; {}^eT_c = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \bar{n} & \bar{o} & \bar{a} & \bar{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

then (9) is deduced from (7) and (8).

$$\begin{cases} x_w = n_x x + o_x y + a_x z + p_x \\ y_w = n_y x + o_y y + a_y z + p_y \\ z_w = n_z x + o_z y + a_z z + p_z \end{cases} \tag{9}$$

If the surface of work piece is a plane, the points in the laser stripe will satisfy its plane equation (10).

$$Ax_w + By_w + Cz_w + 1 = 0 \tag{10}$$

in which A, B and C are the parameters of the work piece plane in frame W. Submitting (9) to (10), then

$$A(n_x x + o_x y + a_x z) + B(n_y x + o_y y + a_y z) + \\ C(n_z x + o_z y + a_z z) + Ap_x + Bp_y + Cp_z + 1 = 0 \tag{11}$$

Let $D = Ap_x + Bp_y + Cp_z + 1$. It is sure that the lens centre of the camera, $[p_x, p_y, p_z]$, is not on the plane of work piece. Therefore the condition $D \neq 0$ is satisfied. Equation (11) is rewritten as (12) via divided by D and applying (6) to it (Xu & Tan, 2004).

$$A_1(n_x x_{c1} + o_x y_{c1} + a_x) + B_1(n_y x_{c1} + o_y y_{c1} + a_y) \\ + C_1(n_z x_{c1} + o_z y_{c1} + a_z) - ax_{c1} - by_{c1} - c = 0 \tag{12}$$

here $A_1 = A/D$, $B_1 = B/D$, $C_1 = C/D$.

If the optical axis of the camera is not parallel to the plane of the laser light, then $c \neq 0$ is satisfied. In fact, the camera must be fixed in some direction except that parallel to the plane of the laser light in order to capture the laser stripe. Dividing (12) by c, then

$$A_2(n_x x_{c1} + o_x y_{c1} + a_x) + B_2(n_y x_{c1} + o_y y_{c1} + a_y)$$
$$+ C_2(n_z x_{c1} + o_z y_{c1} + a_z) - a_1 x_{c1} - b_1 y_{c1} = 1 \tag{13}$$

where $A_2 = A_1/c$, $B_2 = B_1/c$, $C_2 = C_1/c$, $a_1 = a/c$, $b_1 = b/c$.

In the condition that the point of the lens centre $[p_x, p_y, p_z]$ is kept unchangeable in frame O, a series of laser stripes in different directions are formed with the pose change of the vision sensor. Any point in each laser stripe on the same plane of a work piece satisfies (13). Notice the linear correlation, only two points can be selected from each stripe to submit to formula (13). They would form a group of linear equations, whose number is as two times as that of stripes. If the number of equations is greater than 5, they can be solved with least mean square method to get parameters such as A_2, B_2, C_2, a_1, b_1.
Now the task of laser calibration is to find the parameter c. The procedure is very simple. It is well known that the distance between two points P_i and P_j on the stripe is as follows

$$d = \sqrt{(x_{wi} - x_{wj})^2 + (y_{wi} - y_{wj})^2 + (z_{wi} - z_{wj})^2} = \sqrt{d_x^2 + d_y^2 + d_z^2} \tag{14}$$

in which, $[x_{wi}, y_{wi}, z_{wi}]$ and $[x_{wj}, y_{wj}, z_{wj}]$ are the coordinates of point P_i and P_j in the world frame; d_x, d_y, d_z are coordinates decomposition values of distance d. Submitting (6) and (9) to (14), then

$$d_x = \frac{1}{c}[n_x(\frac{x_{c1j}}{a_1 x_{c1j} + b_1 y_{c1j} + 1} - \frac{x_{c1i}}{a_1 x_{c1i} + b_1 y_{c1i} + 1})$$

$$+ o_x(\frac{y_{c1j}}{a_1 x_{c1j} + b_1 y_{c1j} + 1} - \frac{y_{c1i}}{a_1 x_{c1i} + b_1 y_{c1i} + 1}) \tag{15}$$

$$+ a_x(\frac{1}{a_1 x_{c1j} + b_1 y_{c1j} + 1} - \frac{1}{a_1 x_{c1i} + b_1 y_{c1i} + 1})] = \frac{1}{c} d_{x1}$$

In the same way, d_y and d_z are deduced. Then

$$d = \pm \frac{1}{c}\sqrt{d_{x1}^2 + d_{y1}^2 + d_{z1}^2} = \pm \frac{1}{c} d_1 \Rightarrow c = \pm \frac{d_1}{d} \tag{16}$$

where d_1 is the calculated distance between two points on the stripe with parameters a_1 and b_1, and d is the measured distance with ruler.

Then parameters a and b can be directly calculated from c as formula (17). Applying a, b, and c to (6), the sign of parameter c could be determined with the constraint $z>0$.

$$\begin{cases} a = a_1 c \\ b = b_1 c \end{cases} \tag{17}$$

3.2 Experiment and results

The camera in the vision sensor was well calibrated in advance. Its intrinsic parameters M_{in} and extrinsic ones ${}^{e}T_c$ were given as follows.

$$M_{in} = \begin{bmatrix} 2620.5 & 0 & 408.4 \\ 0 & 2619.1 & 312.2 \\ 0 & 0 & 1 \end{bmatrix}, \quad {}^{e}T_c = \begin{bmatrix} -0.0867 & -0.6620 & -0.7444 & 51.9160 \\ -0.0702 & 0.7495 & -0.6583 & -89.9243 \\ 0.9938 & -0.0048 & -0.1115 & 35.3765 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

in which the image size is 768×576 pixels.

3.2.1 Laser Plane Calibration

A structured light vision sensor was mounted on the end-effector of an arc welding robot to form a hand-eye system. The laser stripe was projected to a plane approximately parallel to the XOY plane in frame W. The poses of the vision sensor were changed through the end-effector of the robot for seven times. And the lens centre point $[p_x, p_y, p_z]$ was kept unchangeable in frame W in this procedure. So there were seven stripes in different directions. Any two points were selected from each stripe to submit to (13). Fourteen linear equations were formed. Then the parameters such as A_2, B_2, C_2, a_1, b_1 could be obtained from them. It was easy to calculate the length d_1 of one stripe with a_1 and b_1, and to measure its actual length d with a ruler. In fact, any two points on a laser stripe satisfy (14)-(16) whether the laser stripe is on a plane or not. To improve the precision of manual measure, a block with known height was employed to form a laser stripe with apparent break points, as seen in Fig. 3. The length d_1 was computed from the two break points. Then parameters of the laser plane equation were directly calculated with (13)-(17). The results are as follows.
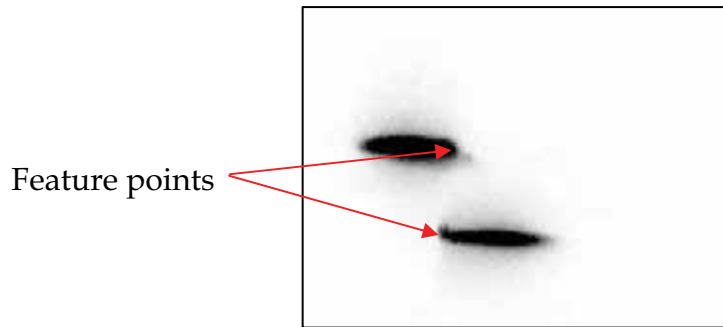
Figure 3. A laser stripe formed with a block

d=23mm, d_1=0.1725, a=-9.2901×10^{-4}, b=2.4430×10^{-2}, c=-7.5021×10^{-3}.

So the laser plane equation in frame C is:
-9.2901×10^{-4}x+2.4430×10^{-2}y-7.5021×10^{-3}z+1=0.

3.2.2 The verification of the hand-eye system

A welding seam of type V was measured by use of the structured light vision sensor to verify the hand-eye system. The measurements were conducted 15 times along the seam. Three points were selected from the laser stripe for each time, which were two edge points and a bottom one. Their coordinates in frame W were computed via the method proposed above. The results were shown in Table 1.

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_{w1}	1019.16	1007.22	1003.34	999.55	991.11	983.95	977.16	973.85	971.13	967.05	963.89	959.34	954.90	952.73	946.64
y_{w1}	38.68	72.51	82.53	93.17	116.12	136.33	153.67	163.10	171.24	182.04	190.31	202.58	214.91	220.74	236.73
z_{w1}	47.09	46.70	46.85	46.66	46.74	46.50	46.74	46.93	46.74	46.68	46.73	46.85	46.73	46.37	46.74
x_{w2}	1031.54	1019.68	1016.36	1012.30	1004.21	996.99	990.81	987.51	984.34	980.33	977.27	972.92	968.07	965.91	960.26
y_{w2}	24.70	57.74	67.19	78.23	100.78	120.92	137.80	147.07	155.73	166.29	174.60	186.46	199.29	205.46	221.04
z_{w2}	37.41	36.38	36.32	36.39	36.20	35.93	35.81	35.91	36.09	35.89	35.94	35.80	36.02	36.05	35.93
x_{w3}	1031.10	1019.13	1015.84	1011.80	1003.76	996.31	989.97	986.84	983.85	979.87	976.71	971.97	967.45	965.37	959.34
y_{w3}	43.27	76.36	86.24	96.94	120.13	140.46	156.70	166.41	174.89	185.57	194.14	206.13	218.65	225.10	241.18
z_{w3}	47.53	46.75	46.71	46.61	46.74	46.62	46.18	46.49	46.54	46.40	46.61	46.60	46.60	46.76	46.98

Table 1. The measurement results of a welding seam of type V

Row 1 was the sequence number of measurement points. Row 2 was one of outside edges of the seam. Row 4 was another. Row 3 was its bottom edge. All data were with unit mm in the world frame. The measurement errors were in the range ±0.2mm. The measurement results are also shown in the world frame and XOY plane in Fig. 4 respectively. Fig. 4 is the data graph shown in 3D space, and Fig. 4 on XOY plane in frame W. It can be seen that the results were well coincided with the edge lines of the seam (Xu & Tan, 2004).
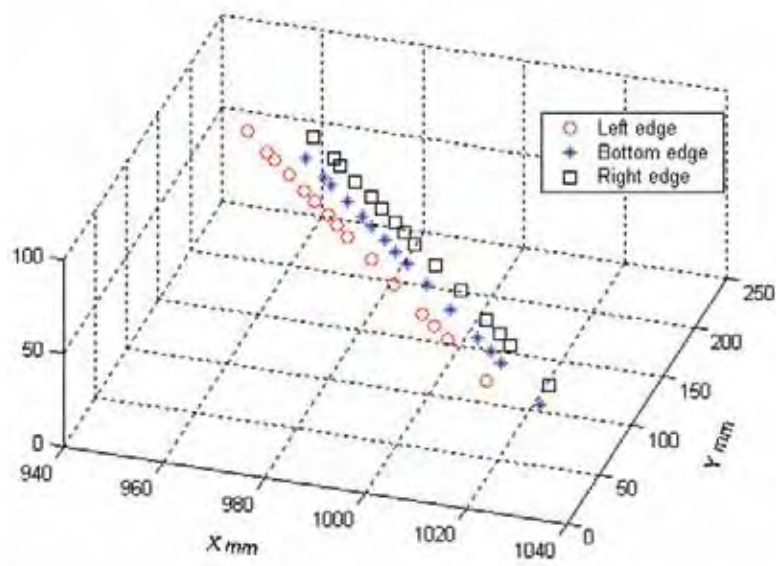
(a) 3D space
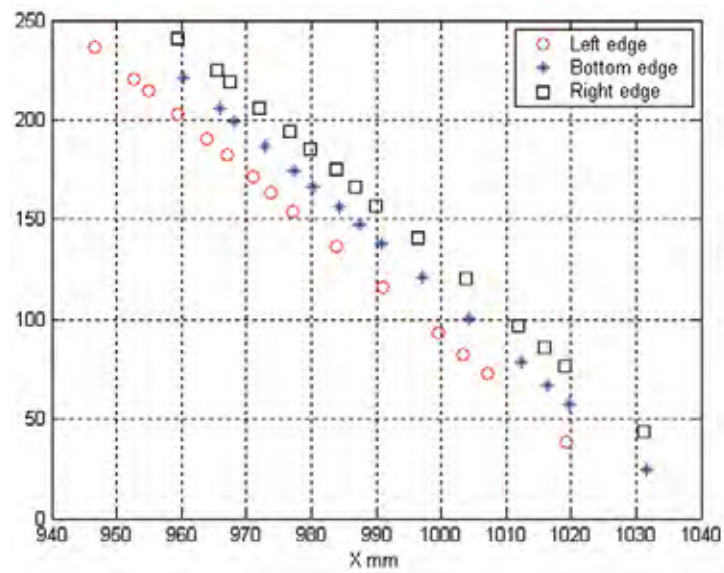
(b) XOY plane

Figure 4. The data graph of vision measurement results of a type V welding seam

4. Feature extraction based on second order difference

4.1 Image pre-processing

The gray image of laser stripe is captured via a camera with light filter. Generally, its size is large. For example, it could be as large as 576×768 pixels. Therefore, simple and efficient image pre-processing is essential to improve visual control performance in real time. The image pre-processing includes image segmentation, image enhancement and binarization (Xu et al., 2004a; 2004b).

4.1.1 Image segmentation

First, the background gray value of the image is counted. Along its horizontal and perpendicular direction, lines with constant space are drawn. Gray values of all pixels on the lines are added, and its average value is taken as the gray value of background. It is given in (18).

$$
\begin{cases}
B = \dfrac{1}{n_w n_1 + n_h n_2}\left(\displaystyle\sum_{i=1}^{n_w}\sum_{j=1}^{n_1} I(i,10j) + \sum_{i=1}^{n_2}\sum_{j=1}^{n_h} I(10i,j)\right) \\
n_1 = Int(n_h/10),\ n_2 = Int(n_w/10)
\end{cases}
\tag{18}
$$

where n_w and n_h are the image width and height respectively, n_1 is the number of horizontal lines, n_2 is the number of vertical lines, and $I(x, y)$ is the gray value of the pixel in coordinates (x, y).

Usually, laser stripe has higher brightness than the background. Along the lines drawn above, all pixels with gray value greater than $B+T_1$ are recorded. The target area on the image is confirmed according to the maximum and the minimum coordinates of pixels recorded along the horizontal and perpendicular direction respectively.

$$
\begin{cases}
X_1 = Min\{i : I(i,10j_1) - B > T_1 \text{ or } I(10i_1, j) - B > T_1\} \\
X_2 = Max\{i : I(i,10j_1) - B > T_1 \text{ or } I(10i_1, j) - B > T_1\} \\
Y_1 = Min\{j : I(i,10j_1) - B > T_1 \text{ or } I(10i_1, j) - B > T_1\} \\
Y_2 = Max\{j : I(i,10j_1) - B > T_1 \text{ or } I(10i_1, j) - B > T_1\} \\
1 \le i \le n_w, 1 \le j \le n_h, i_1 = INT(i/10), j_1 = INT(j/10)
\end{cases}
\tag{19}
$$

where T_1 is the gray threshold. The target area consists of X_1, X_2, Y_1 and Y_2.

The structured light image is suffered from arc light, splash, and acutely changed background brightness during welding. As known, the intensity of the arc light and splash changes rapidly, but the laser intensity keeps stable. According to this fact, the effect of arc light and splash can be partly eliminated via taking the least gray value between sequent images as the new gray

value of the image.

$$I(i,j) = Min\{I_k(i,j), I_{k-1}(i,j)\} \tag{20}$$

where I_k is the image captured at k-th times, and I_{k-1} is k-1-th. $X_1 \le i \le X_2$, $Y_1 \le j \le Y_2$.

4.1.2 Image enhancement and binarization

The target area is divided into several parts, and its gray values are divided into 25 levels. For every part, the appearance frequency of every gray level is calculated, as given in (21).

$$\begin{cases} F(k,h) = \sum_{i=X_1}^{X_2}\sum_{j=Y_1}^{Y_2} P(k,h) \\ P(k,h) = \begin{cases} 1 & k = Int(i/5), h = Int(I(i,j)/10) \\ 0 & others \end{cases} \end{cases} \tag{21}$$

Taking into account the different contrast between the laser stripe and background, the gray value with higher level, whose appearance reaches specified frequency, is regarded as the image enhancement threshold $T_2(k)$.

$$T_2(k) = 10K, iff\ (\sum_{h=25}^{K} F(k,h) > S_1) \vee (F(k,K) > S_2) \tag{22}$$

where S_1 is the specified sum of the frequency with the higher gray level, S_2 is the specified frequency with higher level in one child area, and K is the gray level, $1 \le K \le 25$.

(a)   (b)   (c)   (d)

(e)   (f)   (g)   (h)
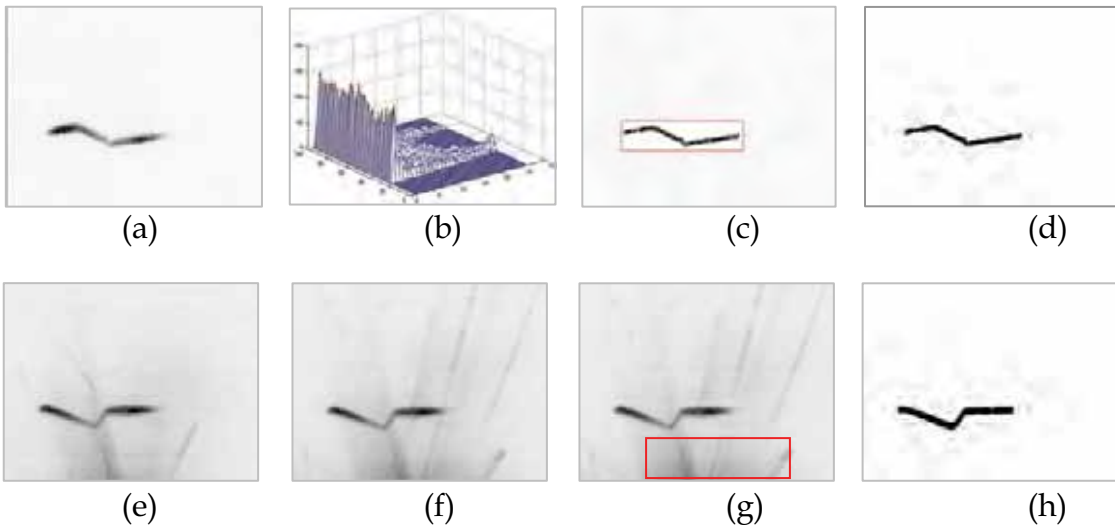
Figure 5. The primary image, frequency distribution map and object segmentation

According to the threshold of every child area, high-pass filter and image enhancement are applied to the target area, followed by Gauss filter and binary thresholding. Fig. 5 is the result of image segmentation of a welding seam. In detail, Fig. 5(a) is the original image with inverse colour, Fig. 5(b) shows its distribution of gray frequency, Fig. 5(c) is the image of the strengthened target area, and Fig. 5(d) is the binary image. Fig. 5(e) and Fig. 5(f) are two frames of original images with inverse colour in sequence during welding, and Fig. 5(g) is the processing result via taking the least gray value in the target area with (20). Fig. 5(h) is its binary image. It can be seen that the binary images of welding seams, obtained after image pre-processing with the proposed method, are satisfactory.

4.2 Features extraction

Because the turning points of the laser stripe are the brim points of the welding seam, they are selected as the feature points. To adjust the pose of the weld torch easily, some points on the weld plane are required. Therefore, the goal of features extraction is to search such turning points and weld plane points from the binary image.

To thin the binary image of welding seam, the average location between the upper edge and the lower one, which is detected from the binary image, is regarded as the middle line of laser stripe. Fig. 6(a) shows the upper, lower edge, and middle line of the laser stripe. Because of the roughness of the binary laser stripe, the middle line curve has noise with high frequency, seen in the bottom of Fig. 6(b).

(a) (b) (c) (d)
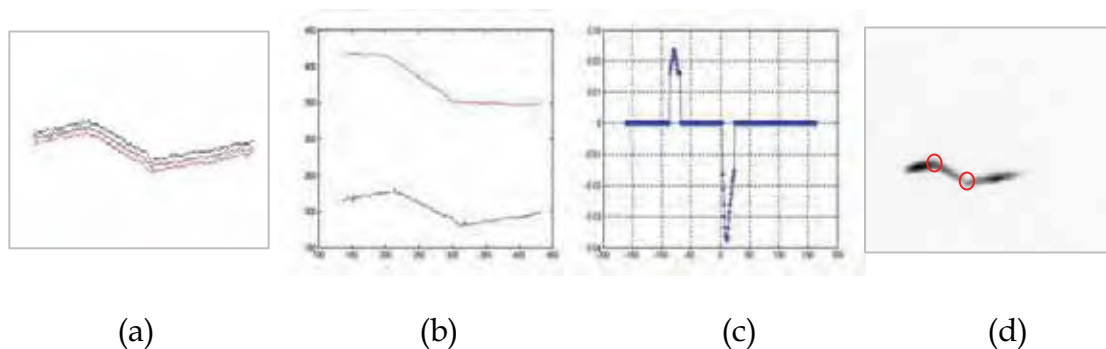
Figure 6. The procedure of features extraction

The middle line stored in an array with two dimensions is transformed via Hotelling transformation, to make its feature direction same as the x-axis. Hotelling transformation is shortly described as follows.

First, the position vector of the average of all points on the middle line is computed.

$$\overline{m}_d = \frac{1}{N} \sum_{i=1}^{N} m_d(i) \tag{23}$$

where N is the point number on the middle line, and $\overline{m}_d$ is the position vector of the average, $\overline{m}_d = \begin{bmatrix} \overline{m}_d(1) & \overline{m}_d(2) \end{bmatrix}^T$. $m_d(i,1)$ is the coordinate x of the i-th point, and $m_d(i,2)$ is the coordinate y.

Second, the position vector of each point on the middle line after Hotelling transformation is calculated.

$$C_d = \frac{1}{N} \sum_{i=1}^{N} m_d(i) m_d(i)^T - \overline{m}_d \overline{m}_d{}^T \tag{24}$$

$$m_{dh}(i) = V[m_d(i) - \overline{m}_d] \tag{25}$$

where $m_{dh}(i) = \begin{bmatrix} m_{dh}(i,1) & m_{dh}(i,2) \end{bmatrix}^T$ is the position vector of the i-th point on the middle line after Hotelling transformation. V is the eigenvector matrix of C_d, whose first row has large eigenvalue.
To clear up the effect of high frequency noise, the middle line after Hotelling transformation should be filtered. In the condition to keep the x-coordinate invariable, y-coordinate is filtered using Takagi-Sugeno fuzzy algorithm, given by equation (26) and (27).

$$\widetilde{m}_{dh}(k,2) = [\sum_{h=-5}^{5} m_{dh}(k-h,2)\mu(h)] \Big/ \sum_{h=-5}^{5} \mu(h) \tag{26}$$

where $\widetilde{m}_{dh}(k,2)$ is the y-coordinate of the k-th point on the filtered middle line. $\mu(h)$ is the membership function.

$$\mu(h) = \begin{cases} 1 & -3 \le h \le 3 \\ 2 - |h|/3 & 3 < |h| \le 5 \\ 0 & |h| > 5 \end{cases} \tag{27}$$

A line gained by Hough transform, which is the closest to the x-axis direction converted by Hotelling transformation, is viewed as the main line. Locations of points on the middle line are mapped into the parameter space $A(p, q)$ of the line function, shown in (28), and the (p, q) with the maximum value of A is the

parameter of the main line. All points on the middle line satisfied with the main line function are feature points of the weld plane.

$$\begin{cases} A(p,q) = \sum_{k=1}^{M} \sum_{p=pMin}^{pMax} B(p,q) \\ B(p,q) = \begin{cases} 1 & q = -p\widetilde{m}_{dh}(k,1) + \widetilde{m}_{dh}(k,2) \\ 0 & others \end{cases} \end{cases} \tag{28}$$

The main line is rotated an angle in order to make it parallel to the x-axis direction.

$$m_{dr}(i) = V_1 \widetilde{m}_{dh}(i) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \widetilde{m}_{dh}(i) \tag{29}$$

where $\theta = \text{atan}(p)$ is the inclination angle between the main line and x-axis, $m_{dr}(i)$ is the position vector of the i-th point on the rotated middle line, V_1 is a rotation matrix formed with $\cos\theta$ and $\sin\theta$.

The point with the maximum of the local second derivative is the turning point of the middle line. After reverse transform as given in (30), the position of the welding seam feature point in the original image is obtained.

$$m_{dm}(i) = V^{-1}V_1^{-1}m_{drm}(i) + \overline{m}_d \tag{30}$$

where $m_{drm}(i)$ is the position vector of the i-th turning point on the middle line, and $m_{dm}(i)$ is the turning point position in the original image.

The curve at the top of Fig. 6(b) shows the middle line after filtered and transformed. The second derivative of the middle line is seen in Fig. 6(c). Two feature points of the welding seam on the original image can be read from Fig. 6(d).

5. Feature extraction based on geometric centre

5.1 Algorithms for profiles extraction

Fig. 7 shows two frames of laser images of a welding seam of type V groove, in which Fig. 7(a) is an original image before welding, and Fig. 7(b) is an image with reflection of laser on the surface of the welding seam after root pass welding. It can be found that the two images are different in a very large degree. So they should be dealt with different strategies. The method proposed in Section 4 is difficult to deal with the image as given in Fig. 7(b). However, the two im-

ages have a common property, that is, the area of the welding seam is just part of the image. So the welding seam area should be detected to reduce the computation cost (Li et al., 2005).

<center>(a)   (b)</center>
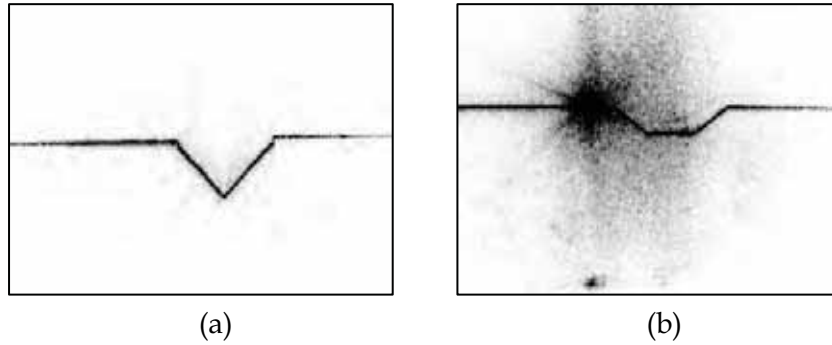
Figure 7. Images of welding seams before and after root pass welding

5.1.1 Welding seam area detection

In order to reduce the computational time required in image processing, only the image of welding seam area is processed. However, some disturbances such as reflection shown in Fig. 7(b) will be segmented in the object area with the method in Section 4, which increases the difficulty of features extraction later. Here, an intensity distribution method is presented to detect the object area. The laser stripes shown in Fig. 7, captured by another visual sensor, are horizontal; their range in column is almost from the first to end. So only the range in row needs to be detected. It can be determined by calculating the distribution of intensity of pixels in row. Apparently, the main peak of intensity is nearby to the position of the main vector of laser stripe. So the range of seams in Y-axis direction of the image plane can be detected reliably with (31).

$$\begin{cases} Y_2 = Min\{Y_p + h_w + m_w,\ n_h\} \\ Y_1 = Max\{Y_p - m_w,\ 0\} \end{cases} \qquad (31)$$

where Y_p is the Y-coordinate of main vector; h_w is the height of welding groove; and m_w is the margin remained. The target area consists of 0, n_w, Y_1 and Y_2.

5.1.2 Column based processing

Column based profiles extraction calculates the distribution of pixels' intensity with columns to get the points of profile. Some algorithms such as multi-peak method and centre of gravity (Haug & Pristrchow, 1998), gradient detection and average of upper edges and the lower edges in Section 4 are all effective

for the task. In order to get high quality profiles of seams, a method that combines smoothing filter, maximum detection and neighbourhood criteria is proposed.
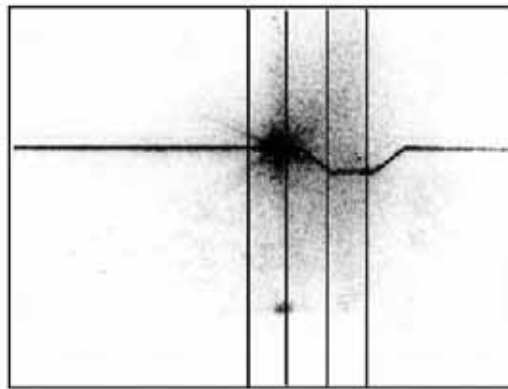
(a)
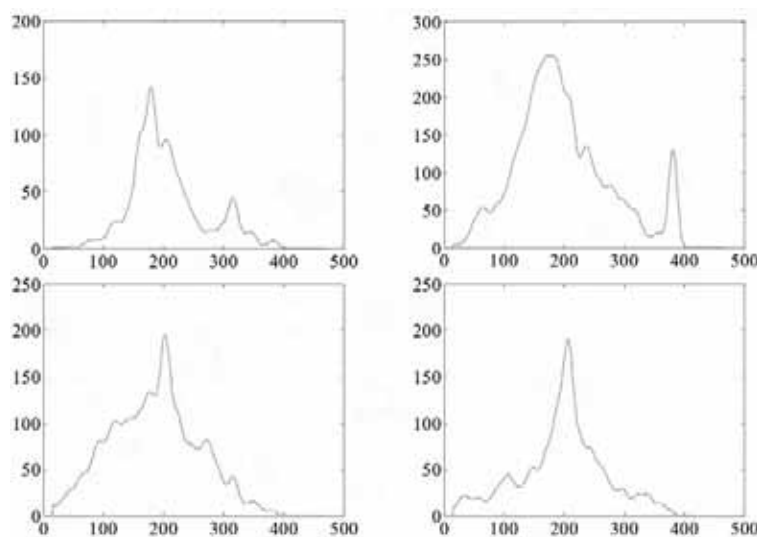
(b)

Figure 8. Intensity extraction of four columns

Firstly, a low pass filter is designed to smooth the intensity curve of column i. Usually, the pixels of profiles are in the area of main peak, and the peaks caused by disturbances are lower or thinner. After smoothing the intensity curve, the plateau is smoothed with one maximum in main peak, and the lower or thinner peaks are smoothed into hypo-peaks. Fig. 8 gives an example of intensity distribution of column 300, 350, 400, 450 of a welding seam image. Fig. 8(a) shows the original image and the positions of four example columns. Fig. 8(b) shows their intensity distribution.

Then according to the analysis of the image gray frequency, the self-adaptive thresholds of the images are calculated. Only the pixels whose intensity exceeds the thresholds are regards as valid. Thus the intensity curve is fragmented to several peaks. By calculating the area of peaks, the main peak can be gotten.

In this way, there is only one point with maximum intensity remained on the main peak in the intensity distribution curve. In other words, one point on the profile for each column is detected. Thus an array of points indexed with column is extracted through intensity distribution from the welding image.

In order to extract points of profile more properly, the criterion of neighbour is applied. Since the profiles of grooves should be a continuous curve, the points that are apparently inconsistent to neighbour points should be rejected. When the pixels whose intensity value exceeds the thresholds cannot be found, there will be no record in the array for this column. In these situations, the data in the array will be not continuous. Then the linear interpolation algorithm is used to fill up the curve between broken points, and the discrete points array is transferred to continuous profile curve.

5.2 Features extraction for seam tracking

In order to extract features of profiles for seam tracking, the first task is to select features. Usually the corner points of profiles are brim points of the welding groove, and they are often selected as features of welding seams in single pass welding (Kim et al., 1996; Wu et al., 1996). The corner detection method is only valid for images of single pass welding. But in multi-pass welding, there is distortion caused by weld bead in the bottom of groove. There are welding slag remained on the surface of welding work piece sometimes. As shown in Fig. 7(b), it is hard to judge the proper corner points by the second derivative because of the distortion. So the features extraction with corner point's detection is not reliable in this case.

The centre of gravity of groove area is selected as features because of its good stabilization relative to corner points.

Fig. 9(a) shows a profile of groove extracted with the method in Section 5.1 from a welding seam after welding root pass. Firstly, the profile is smoothed by a Hanning filter to eliminate high frequency noise, as shown in Fig. 9(b). In order to get the figure of groove area, the main vector of the profile is required. It can be extracted by Hough transform as described in Section 4.

Because the main vector is the asymptote of the profile, the main vector and the profile can form a trapeziform figure approximately. In the first step, the bottom of groove area is detected by template matching. Then from the bottom of groove, the points on the profile are searched forward and backward respectively.

(a) (b)
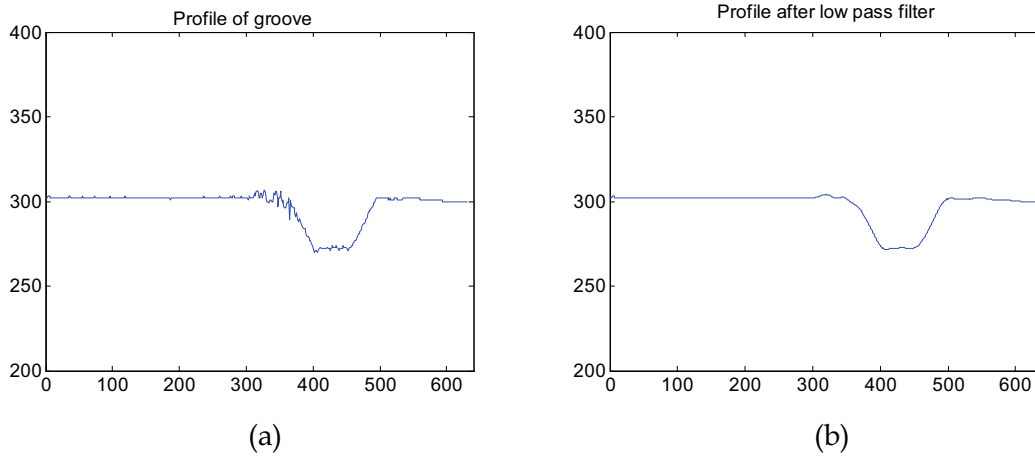
Figure 9. Profiles of the groove after root pass

The two borders (b1, b2) of the figure are gotten when the distances between the points on the profile and the main vector are less than the thresholds (5 pixels here). The trapeziform figure is defined with the border points, as shown in Fig. 10. Finally, the gravity centre of the figure is extracted as features by (32). A perpendicular to the main vector is drawn through the gravity centre. The intersection is taken as the feature point of the welding seam.

$$
\begin{cases}
F_u = \sum_{i=b1}^{b2} i \left[y_p(i) - y_v(i) \right] \Big/ \sum_{i=b1}^{b2} \left[y_p(i) - y_v(i) \right] \\[2mm]
F_v = \sum_{i=b1}^{b2} 0.5 \left[y_p(i)^2 - y_v(i)^2 \right] \Big/ \sum_{i=b1}^{b2} \left[y_p(i) - y_v(i) \right]
\end{cases}
\tag{32}
$$

where F_u, F_v are the coordinates of geometric centre; y_p and y_v are Y-coordinates of points on the profile and the main vector.
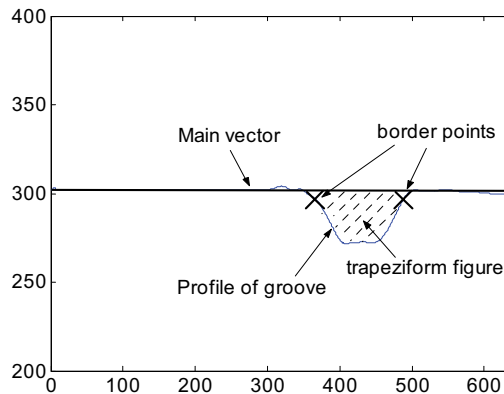
Figure 10. Main vector and border points on the groove profile

6. Hybrid visual control method for robotic welding

6.1 Jacobian matrix from the image space to the Cartesian space of the end-effector

From (3) and (4), formula (33) is deduced (Xu et al., 2004b; 2005).

$$
\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = -\left(\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x_{c1} \\ y_{c1} \\ 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} x_{c1} \\ y_{c1} \\ 1 \end{bmatrix}
\tag{33}
$$

Further more, the coordinates of a point P in Cartesian space are obtained as (34) in the end-effector frame.

$$
\begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = {}^eM_c \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} {}^em_{xc} & {}^en_{xc} & {}^eo_{xc} & {}^ep_{xc} \\ {}^em_{yc} & {}^en_{yc} & {}^eo_{yc} & {}^ep_{yc} \\ {}^em_{zc} & {}^en_{zc} & {}^eo_{zc} & {}^ep_{zc} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} {}^eR_c & {}^ep_c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}
\tag{34}
$$

in which, ${}^eM_c$ is the extrinsic parameter matrix of the camera relative to the end-effector of the robot. ${}^eR_c$ is the rotation translation matrix, and ${}^ep_c$ is the position vector.

The time derivative of (34) is as follows.

$$
\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{z}_e \\ 0 \end{bmatrix} = \begin{bmatrix} {}^eR_c & {}^ep_c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{z}_c \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{z}_e \end{bmatrix} = {}^eR_c \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{z}_c \end{bmatrix}
\tag{35}
$$

Submitting (5) into (33) and applying time derivative, then (Xu et al., 2005)

$$
\begin{aligned}
\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{z}_c \end{bmatrix} &= \frac{1}{D^2} \begin{bmatrix} -b(v-v_0)/(k_xk_y)-c/k_x & b(u-u_0)/(k_xk_y) & 0 \\ a(v-v_0)/(k_xk_y) & -a(u-u_0)/(k_xk_y)-c/k_y & 0 \\ a/k_x & b/k_y & 0 \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ 0 \end{bmatrix} \\
&= \frac{1}{D^2} \begin{bmatrix} -b(v-v_0)/(k_xk_y)-c/k_x & b(u-u_0)/(k_xk_y) \\ a(v-v_0)/(k_xk_y) & -a(u-u_0)/(k_xk_y)-c/k_y \\ a/k_x & b/k_y \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = J_c(u,v) \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix}
\end{aligned}
\tag{36}
$$

in which, $J_c(u,v)$ is the Jacobian matrix from image space to Cartesian space in the camera frame C. $D = a(u - u_0)/k_x + b(v - v_0)/k_y + c$, is a constraint of the laser plane.

Submitting (36) to (35), the Jacobian matrix from image space to Cartesian space in the end-effector frame E is obtained, as given in (37).

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{z}_e \end{bmatrix} = J(u,v)\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} \Rightarrow \begin{bmatrix} dx_e \\ dy_e \\ dz_e \end{bmatrix} = J(u,v)\begin{bmatrix} du \\ dv \end{bmatrix} \tag{37}$$

where symbol d represents derivative.

Formula (38) gives the Jacobian matrix from image space to Cartesian space in the end-effector frame, which describes the relation of the differential movements between a feature point on image plane and the end-effector. The parameters in (38), such as $[k_x, k_y]$, $[u_0, v_0]$, ^eR_c, a, b and c, can be achieved through camera and laser plane calibration.

$$J(u,v) = {}^eR_c J_c(u,v) = \frac{1}{D^2}{}^eR_c \begin{bmatrix} -b(v-v_0)/(k_x k_y) - c/k_x & b(u-u_0)/(k_x k_y) \\ a(v-v_0)/(k_x k_y) & -a(u-u_0)/(k_x k_y) - c/k_y \\ a/k_x & b/k_y \end{bmatrix} \tag{38}$$

6.2 Hybrid visual servoing control

6.2.1 The model of hybrid visual servoing control for robotic arc welding

The scheme of hybrid visual servoing control method proposed in this chapter for robotic arc welding consists of four main parts, such as the control of moving along welding seam, the control of tracking adjusting, the position control of the robot, and the image feature extraction. The block diagram is shown in Fig. 11. Position-based visual control in Cartesian space is employed in the process of moving along the welding seam. From the image of the structured light stripe at i-th sampling, the image coordinates u_i' and v_i' for feature point P_i on the stripe can be extracted. Then $[x_{ei}, y_{ei}, z_{ei}]$, the coordinates of point P_i in the end-effector frame, can be computed with (5), (33) and (34). In addition, the coordinates of point P_{i-1} in the current end-effector frame, $[x_{ei-1}, y_{ei-1}, z_{ei-1}]$, can be obtained through transformation according to the movement Δ_i of the end-effector at last times. Then the direction of welding seam is determined with $[x_{ei-1}, y_{ei-1}, z_{ei-1}]$ and $[x_{ei}, y_{ei}, z_{ei}]$. For reducing the influence of random ingredients, the coordinates of $n+1$ points P_{i-n}-P_i in the end-effector frame can be used to calculate the direction of the welding seam through fitting. The direction

vector of the welding seam is taken as movement Δ_{li} of the end-effector after multiplying with a proportion factor K. In the part of the control of moving along welding seam, the measured direction above is taken as the desired value to control the movement of the robot. It is inevitable that there exist apparent errors in the process of moving along the welding seam. Therefore the second part, that is, tracking adjusting with visual servoing control in image space, is introduced. According to the desired image coordinates $[u, v]$ and the actual ones $[u_i', v_i']$ of the feature point P_i, the errors $[du_i, dv_i]$ of the image coordinates as well as the estimated Jacobian matrix $\hat{J}(u,v)$ are calculated. Then $[d\hat{x}_e, d\hat{y}_e, d\hat{z}_e]$ is computed using (37), which is considered as the position errors of the end-effector. The differential movement Δ_{si} of the end-effect-or is generated with PID algorithm according to these errors. Δ_i, the sum of Δ_{si} and Δ_{li}, is taken as the total movement of the end-effector. The third part, the position control of the robot, controls the motion of the robot according to Δ_i. In detail, the position and pose of the end-effector in next step, in the world frame, is calculated with the current one and Δ_i. The joint angle value for each joint of the robot is calculated using inverse kinematics from the position and pose of the end-effector in next step. Then the position controller for each joint controls its motion according to the joint angle. The position control of the robot is realized with the control device attached to the robot set.
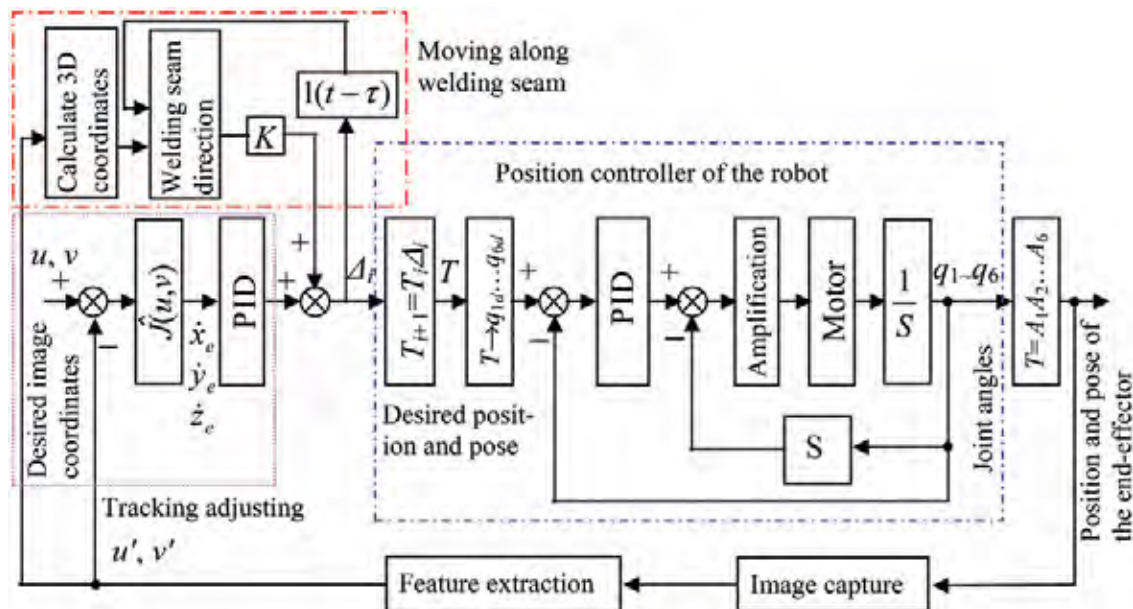
Figure 11. The block diagram of hybrid visual servoing control for robotic arc welding

The other parts such as the control of moving along welding seam, tracking adjusting and image feature extraction are completed with an additional computer (Xu et al., 2005).

6.2.2 The model simplification

In the hybrid visual servoing control system for robotic arc welding, as shown in Fig. 11, the control of moving along welding seam takes the direction of welding seam as the desired value to make the end-effector to move ahead. Its output Δ_{li} can be considered as disturbance $\xi(t)$ for the part of image-based visual servoing control. In the part of the position control of the robot, the motions of the robot are merely controlled according to the desired movements of the end-effector and the stated velocity. In the condition that the movement velocity is low, the part of the position control for the movement of the end-effector can be considered as a one-order inertia object. Therefore, the model of the hybrid visual servoing control system can be simplified to the dynamic framework as shown in Fig. 12.
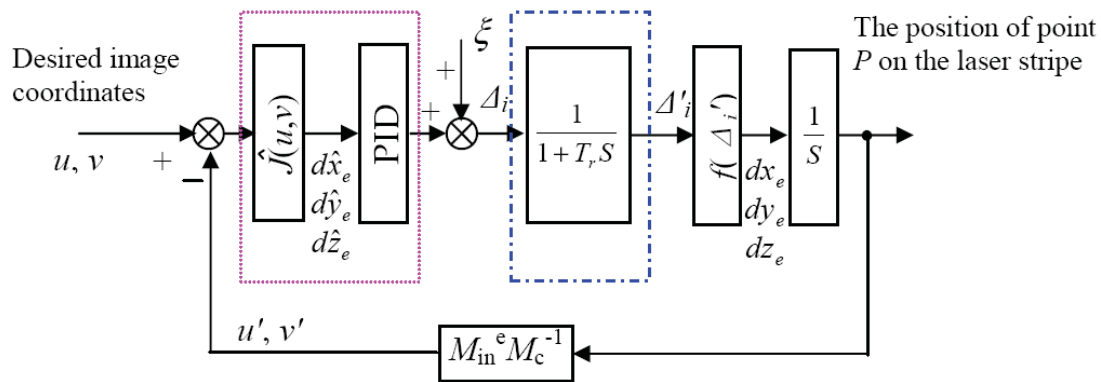
Figure 12. The simplified block diagram of hybrid visual servoing control for robotic arc welding

Although the laser stripe moves with the end-effector, the position $[x_e, y_e, z_e]$, in the end-effector frame, of the feature point P on the stripe will vary with the movement of the end-effector. The relation between the movement of the end-effector and $[x_e, y_e, z_e]$ is indicated as $f(\Delta_i')$. The model for the camera and image capture card is described as $M_{in}{}^e M_c{}^{-1}$.

6.2.3 The relation between the end-effector's movement and the feature point position

In the end-effector frame, the equation of the laser plane can be expressed as (39).

$$
\begin{cases}
a_e x + b_e y + c_e z + 1 - d_e = 0 \\
a_e = a m_x + b n_x + c o_x \\
b_e = a m_y + b n_y + c o_y \\
c_e = a m_z + b n_z + c o_z \\
d_e = a \vec{m} \cdot \vec{p} + b \vec{n} \cdot \vec{p} + c \vec{o} \cdot \vec{p}
\end{cases}
\tag{39}
$$

where $\vec{m} = \begin{bmatrix} m_x & m_y & m_z \end{bmatrix}^T$, $\vec{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$, and $\vec{o} = \begin{bmatrix} o_x & o_y & o_z \end{bmatrix}^T$ are the orientation vector of ${}^e M_c$ as (34).

Assume the equation of the welding seam, in the end-effector frame, at the i-th sampling is as given in (40).

$$
\begin{cases}
x_{li} = x_{li0} + k_x t \\
y_{li} = y_{li0} + k_y t \\
z_{li} = z_{li0} + k_z t
\end{cases}
\tag{40}
$$

From (39) and (40), the coordinates of the feature point P_i on the stripe, in the end-effector frame, at the i-th sampling is deduced, seen in (41).

$$
\begin{cases}
x_{ei} = x_{li0} + k_x t_i \\
y_{ei} = y_{li0} + k_y t_i \\
z_{ei} = z_{li0} + k_z t_i \\
t_i = \dfrac{d_e - 1 - a_e x_{li0} - b_e y_{li0} - c_e z_{li0}}{a_e k_x + b_e k_y + c_e k_z}
\end{cases}
\tag{41}
$$

After the movement of the end-effector $\Delta_i' = [\Delta'_{ix}, \Delta'_{iy}, \Delta'_{iz}]^T$, the equation of the welding seam in the end-effector frame is obtained.

$$
\begin{cases}
x_{li+1} = x_{li0} + \Delta'_{ix} + k_x t \\
y_{li+1} = y_{li0} + \Delta'_{iy} + k_y t \\
z_{li+1} = z_{li0} + \Delta'_{iz} + k_z t
\end{cases}
\tag{42}
$$

Applying (42) to (39), the resolution obtained as (43) is the coordinates of the feature point P_{i+1} on the stripe, in the end-effector frame, at the $i+1$-th sampling.

$$\begin{cases} x_{ei+1} = x_{li0} + \Delta'_{ix} + k_x t_{i+1} \\ y_{ei+1} = y_{li0} + \Delta'_{iy} + k_y t_{i+1} \\ z_{ei+1} = z_{li0} + \Delta'_{iz} + k_z t_{i+1} \\ t_{i+1} = \dfrac{d_e - 1 - a_e(x_{li0} + \Delta'_{ix}) - b_e(y_{li0} + \Delta'_{iy}) - c_e(z_{li0} + \Delta'_{iz})}{a_e k_x + b_e k_y + c_e k_z} \end{cases} \qquad (43)$$

By comparing (41) and (43), the relation of the coordinates between P_{i+1} and P_i in the end-effector frame is derived.

$$\begin{cases} x_{ei+1} = x_{ei} + \Delta'_{ix} + k_x t_{ii+1} \\ y_{ei+1} = y_{ei} + \Delta'_{iy} + k_y t_{ii+1} \\ z_{ei+1} = z_{ei} + \Delta'_{iz} + k_z t_{ii+1} \\ t_{ii+1} = \dfrac{-a_e \Delta'_{ix} - b_e \Delta'_{iy} - c_e \Delta'_{iz}}{a_e k_x + b_e k_y + c_e k_z} \end{cases} \qquad (44)$$

Formula (44) can be rewritten in the form of matrix as (45).

$$\begin{bmatrix} dx_{ei+1} \\ dy_{ei+1} \\ dz_{ei+1} \end{bmatrix} = \begin{bmatrix} x_{ei+1} - x_{ei} \\ y_{ei+1} - y_{ei} \\ z_{ei+1} - z_{ei} \end{bmatrix} = \begin{bmatrix} 1 - a_e k_x / D_e & -b_e k_x / D_e & -c_e k_x / D_e \\ -a_e k_y / D_e & 1 - b_e k_y / D_e & -c_e k_y / D_e \\ -a_e k_z / D_e & -b_e k_z / D_e & 1 - c_e k_z / D_e \end{bmatrix} \begin{bmatrix} \Delta'_{ix} \\ \Delta'_{iy} \\ \Delta'_{iz} \end{bmatrix} = F \begin{bmatrix} \Delta'_{ix} \\ \Delta'_{iy} \\ \Delta'_{iz} \end{bmatrix} \qquad (45)$$

where $D_e = a_e k_x + b_e k_y + c_e k_z$ represents the constraint of the equation of the structured light plane in the end-effector frame. D_e is a constant for line shaped welding seams with the movement of robot. F is a transformation matrix, which describes the relation of the position variations between the end-effector and the feature point P on the stripe in the end-effector frame.

6.2.4 Stability analysis

Suppose the error vector $\bar{e} = [du \quad dv]^T = [u - u' \quad v - v']^T$. The states variables are selected as $X_1 = \bar{e}$, $X_2 = \Delta_i = [\Delta_{ix} \quad \Delta_{iy} \quad \Delta_{iz}]^T$, and $X_3 = \Delta'_i = [\Delta'_{ix} \quad \Delta'_{iy} \quad \Delta'_{iz}]^T$. It is easy to establish the state equation of the system as (46) reference from Fig. 12.

$$\begin{cases} \dot{X}_1 = -M_{in} \, {}^e M_c^{-1} F X_3 \\ \dot{X}_2 = K_i \hat{J} X_1 - (1/T_r) K_d \hat{J} M_{in} \, {}^e M_c^{-1} F X_2 + (K_d / T_r - K_p) \hat{J} M_{in} \, {}^e M_c^{-1} F X_3 \\ \dot{X}_3 = (1/T_r) X_2 - (1/T_r) X_3 \end{cases} \qquad (46)$$

A positive Lyapunov function V is configured as (47). Its time derivative is seen in (48).

$$V = \frac{1}{2}(\hat{J}X_1)^T \hat{J}X_1 + \frac{1}{2}X_2^{\ T}X_2 + \frac{1}{2}X_3^{\ T}X_3 \tag{47}$$

$$\begin{aligned}
\dot{V} =& -(\hat{J}X_1)^T \hat{J}M_{in}{}^e M_c^{-1}FX_3 + X_2^T K_i \hat{J}X_1 - \frac{1}{T_r}X_2^T K_d \hat{J}M_{in}{}^e M_c^{-1}FX_2 \\
&+ X_2^T[(\frac{1}{T_r}K_d - K_p)\hat{J}M_{in}{}^e M_c^{-1}F + \frac{1}{T_r}]X_3 - \frac{1}{T_r}X_3^T X_3 \\
=& -\frac{1}{2}(\hat{J}X_1)^T \hat{J}M_{in}{}^e M_c^{-1}F\hat{J}X_1 + \frac{1}{2}(\hat{J}X_1 - X_3)^T \hat{J}M_{in}{}^e M_c^{-1}F(\hat{J}X_1 - X_3) - \frac{1}{2}X_3^T \hat{J}M_{in}{}^e M_c^{-1}FX_3 \\
&+ \frac{1}{2}(\hat{J}X_1)^T K_i \hat{J}X_1 - \frac{1}{2}(\hat{J}X_1 - X_2)^T K_i(\hat{J}X_1 - X_2) + \frac{1}{2}X_2^T K_i X_2 - \frac{1}{T_r}X_2^T K_d \hat{J}M_{in}{}^e M_c^{-1}FX_2 \\
&+ \frac{1}{2}X_2^T[(\frac{1}{T_r}K_d - K_p)\hat{J}M_{in}{}^e M_c^{-1}F + \frac{1}{T_r}]X_2 + \frac{1}{2}X_3^T[(\frac{1}{T_r}K_d - K_p)\hat{J}M_{in}{}^e M_c^{-1}F + \frac{1}{T_r}]X_3 \\
&- \frac{1}{2}(X_2 - X_3)^T[(\frac{1}{T_r}K_d - K_p)\hat{J}M_{in}{}^e M_c^{-1}F + \frac{1}{T_r}](X_2 - X_3) - \frac{1}{T_r}X_3^T X_3 \\
\leq& -\frac{1}{2}(\hat{J}X_1)^T(\hat{J}M_{in}{}^e M_c^{-1}F - K_i)\hat{J}X_1 - \frac{1}{2}X_2^T(K_p \hat{J}M_{in}{}^e M_c^{-1}F - \frac{I}{T_r} - K_i)X_2 \\
&- \frac{1}{2}X_3^T(I + K_p - \frac{1}{T_r}K_d)\hat{J}M_{in}{}^e M_c^{-1}FX_3 + o(\delta^2)
\end{aligned} \tag{48}$$

where I is a unit matrix, and $o(\delta^2) = \frac{1}{2}(\hat{J}X_1 - X_3)^T \hat{J}M_{in}{}^e M_c^{-1}F(\hat{J}X_1 - X_3)$ is a two-order infinitely small quantity term that can be ignored.

Obviously, if the condition (49) is satisfied, then $\dot{V} < 0$. According to the stability theorem of Lyapunov, the system is asymptotic stable.

$$\begin{cases}
K_i < \hat{J}M_{in}{}^e M_c^{-1}F \\
K_i < K_p \hat{J}M_{in}{}^e M_c^{-1}F - \frac{I}{T_r} \\
K_d < T_r I + T_r K_p
\end{cases} \tag{49}$$

Discussion: (1) As the ideal case, the parameters of the camera and the laser plane are accurately calibrated, that is, $\hat{J} = J$. It is easy to validate that $\hat{J}M_{in}{}^e M_c^{-1}F = I$ is satisfied. It means that the system is degenerated as a linear system.

(2) In the case that there exist errors in $\hat{J}$, if $\hat{J}M_{in}{}^e M_c^{-1}F$ is positive definite, then the PID parameters that satisfy condition (49) can make the system be asymptotic stable.

(3) If $\hat{J}M_{in}{}^e M_c^{-1}F$ is negative definite, then it is not ensured that the system is stable.

6.3 Experiment and results

The experiment system consists of a master computer, a local controller of the robot, a robot Yaskawa UP6, a camera, a laser emitter, welding power source, a welding wire supplier, a welding gun and a CO_2 gas container. The master computer is for image features extraction, the control of moving along welding seam, and the control of tracking adjusting. It outputs the relative movement value Δ_i of the end-effector to the local controller of the robot. The local controller controls the robot to move according to Δ_i. The camera, laser emitter and the welding gun are all fixed on the end-effector of the robot. The stripe formed by the laser plane projecting on the welding seam is ahead of the welding gun tip about 25mm.

Firstly, the camera and the laser plane were calibrated. The intrinsic parameters and extrinsic ones relative to the end-effector frame are as follows. Here, the image size was 768×576 in pixel.

$$M_{in} = \begin{bmatrix} 2663.8 & 0 & 445.7 \\ 0 & 2655.9 & 321.0 \\ 0 & 0 & 1 \end{bmatrix}, \; {}^eM_c = \begin{bmatrix} -0.1301 & -0.6102 & -0.7815 & 54.5258 \\ -0.0299 & 0.7903 & -0.6120 & -87.5896 \\ 0.9911 & -0.0562 & -0.1210 & 39.1279 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The laser plane equation in the camera frame is: $-0.000862x+0.004382y-0.004470z +1=0$.

It is a key factor of the visual control to select stable image features. Aiming at the structured light images as shown in Fig. 5, the points with local maximum slope variation in the stripe was selected as candidate feature points. The obvious advantage of our strategy was that the feature points selected above are on the centre line of the welding seam, such as a type V groove welding seam and a lap one. The image processing and feature extraction method in Section 4 was employed to compute the image coordinates of the feature points. The position and pose of the welding gun was adjusted adequately before welding. The images captured at this time were free from the arc light. The image coordinates of the feature point could be extracted more accurately. They were taken as the desired image coordinates $[u, v]$ for the part of tracking adjusting control. During welding, multiple candidate feature points may be obtained sometimes. In this case, the candidate feature point which image coordinates are nearest to $[u, v]$ is selected as feature point.

In the experiment of tracking and welding, the moving velocity of the robot was set to 0.003m/s. The PID parameters in tracking adjusting control were given as:

$$K_p = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}, K_d = 0, K_i = \begin{bmatrix} 0.05 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.02 \end{bmatrix}$$

The experimental welding seams were a type V groove welding seam and a lap one. The protection gas was CO_2. The transition mode of welding was short circuit. The welding experiments for a type V groove welding seam and a lap one were respectively conducted by using the methods proposed in this chapter. The results showed that the welding seam could be recognized and tracked well. And the shape of weld mark was good. Fig. 13 shows the results of welding experiment for a lap welding seam. The situation for pixel coordinate u' of feature point during tracking and welding is shown in Fig. 13(a), and v' in Fig. 13(b). Their horizontal coordinates are sample times. The pixel coordinates $[u', v']$ of feature points during tracking and welding are shown in Fig. 13(c). The weld mark after welding is in Fig. 13(d). It can be found that there existed larger errors near by the end stage. It was because of a small piece of scrap on the welding seam, which resulted in the image coordinates of the feature point with large errors.

(a)   (b)   (c)   (d)
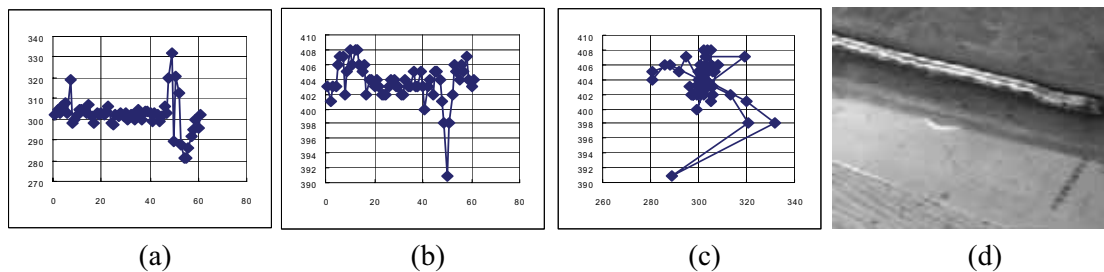
Figure 13. The experimental results

7. Conclusions

A visual control system for robotic welding is introduced in this chapter. The calibration of a vision sensor, the processing algorithms for laser stripe images, and a hybrid visual control method are discussed in detail.

Based on the robot's movement, a method of the structured light vision sensor's calibration is proposed. The laser plane can be calibrated from a group of rotation movements. The experimental results show its effectiveness. It is easy to be realized and provides the possibility to run hand-eye system calibration automatically.

The image processing algorithms proposed in this chapter include two catego-

ries such as feature extraction methods based on second order difference and geometric centre. The former can recognize welding seam of type V groove and find feature points with precise in the case of strong reflection, arc light and splash disturbance. The latter can robustly provide feature points for the welding seam after root pass welding.

A hybrid visual servoing method is proposed in this chapter, which consists of a position control inner-loop in Cartesian space and two outer-loops. One is position-based visual control in Cartesian space; another is image-based visual servoing control in image space. The former is employed for the control of moving along the direction of the welding seam. The latter is for the control of tracking adjusting. The Jacobian matrix from image space of the feature point on structured light stripe to Cartesian space is variable with the movement of the end-effector. But there exists not singular problem in the Jacobian matrix. So the target can be kept in the view field of the camera with the control method above. In the case that the errors between the estimated and real parameters of the camera and the laser plane are not very large, the asymptotic stability of the control system can be ensured through the selection of adequate PID parameters.

The experimental results show the effectiveness of the hybrid visual servoing control system for robotic arc welding. The current position and pose are not necessary for the hybrid visual servoing control system. So it can be applied to many kinds of robots, which can accept the commands of relative movement to the end-effector, to realize visual measurement and tracking.

In addition, whether the work piece is clean or not has obvious influence on the visual measurement results. The unclean surface sometimes results in gross errors of the welding seam tracking. How to eliminate the influence of the gross errors in image space is our work in near future. And the automated adjustment of the position and pose of the welding gun in the start stage is another problem to deal with in future.

Acknowledgement

8. References

Bakos, G. C.; Tsagas, N. F.; Lygouras, J. N.; Lucas J. (1993). Long distance non-contact high precision measurements. *International Journal of Electronics*, Vol. 75, No. 6, pp. 1269-1279, ISSN: 0020-7217.

Bolmsjo, G.; Olsson, M. & Cederberg, P. (2002). Robotic arc welding — trends and developments for higher autonomy. *The Industrial Robot*, Vol. 29, No. 2, pp. 98-104, ISSN: 0143-991X.

Cervera, E.; Berry, F. & Martinet, P. (2002). Image-based stereo visual servoing: 2D vs 3D features. *15th Triennial World Congress of the International Federation of Automatic Control*, pp. 1630-1635, ISBN: 0-08-044295-1, Barcelona, Spain, July 2002, Elsevier Science Press.

Chaumette, F. & Malis, E. (2000). 2 1/2 D visual servoing: a possible solution to improve image-based and position-based visual servoings. *IEEE International Conference on Robotics and Automation*, pp. 630-635, ISBN: 0-7803-5889-9, San Francisco, USA, Apr. 2000, Institute of Electrical and Electronics Engineers Inc., Piscataway, USA

Corke, P. I. & Good, M. C. (1993). Controller design for high-performance visual servoing. *Proceedings 12th World Congress International Federation of Automatic Control*, Vol. 9, pp. 395-398, ISBN: 0-08-042211-X, Sydney, Australia, July 1993, Elsevier Science Press.

Corke, P. I. & Good, M. C. (1996). Dynamic effects in visual closed-loop systems. *IEEE Transaction on Robotics and Automation*, Vol. 12, No. 5, pp. 671-683, ISSN: 1042-296X.

Faugeras, O. D. & Toscani, G. (1986). The calibration problem for stereo. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 15-20, ISBN: 0-8186-0721-1, Miami Beach, USA, 1986, IEEE Press, New York, USA.

Hager, G. D.; Hutchinson, S. & Corke, P. I. (1996). A tutorial on visual servo control. *IEEE Transaction on Robotics and Automation*, Vol. 12, No. 5, pp. 651-670, ISSN: 1042-296X.

Haug, K. & Pristrchow, G. (1998). Robust laser stripe sensor for the automated welding seam tracking in the shipbuilding industry. *Proceedings of the 24th Annual Confer-ence of the IEEE Industry Electronics Society*, pp. 1236-1241, Aachen, Germany, Sep. 1998, IEEE Comp Soc, Los Alamitos, USA.

Kim, J. S.; Son, Y. T.; Cho, H. S.; Koh, K. I. (1996). A robust visual seam tracking system for robotic arc welding. *Mechantronics*, Vol. 6, No. 2, pp. 141-163, ISSN: 0957-4158.

Li, Y.; Xu, D. & Tan, M. (2005). Robust features extraction for structured light images of welding seam in multi-pass submerged arc welding, *7th International Conference on Electronic Measurement & Instruments*, Vol. 6, pp. 1559-1563, ISBN: 7-5062-7443-4, Beijing, China, Aug. 2005, International Academic Publishers, Beijing.

Ma, S. D. (1996). A self-calibration technique for active vision system. *IEEE Transaction on Robotics and Automation*, Vol. 12, No. 1, pp. 114-120, ISSN: 1042-296X.

Malis, E.; Chaumette, F. & Boudet, S. (1999). 2D 1/2 visual servoing. *IEEE Transaction on Robotics and Automation*, Vol. 15, No. 2, pp. 234-246, ISSN: 1042-296X.

Malis, E.; Chaumette, F. & Boudet, S. (1998). Positioning a coarse-calibrated camera with respect to an unknown object by 2D 1/2 visual servoing. *IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1352-1359, ISSN: 1050-4729, Leuven, Belgium, May 1998, IEEE Press, Piscataway, USA.

Tsai, R. Y. (1987). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf cameras and lens. *IEEE Transactions on Robotics and Automation*, Vol. 3, No. 4, pp. 323-344, ISSN: 1042-296X.

Wilson, M. (2002). The role of seam tracking in robotic welding and bonding. *Industrial Robot*, Vol. 29, No. 2, pp. 132-137, ISSN: 0143-991X.

Wu, J.; Smith, J. S. & Lucas, J. (1996). Weld bead placement system for multi-pass welding. *IEE Proceedings–Science, Measurement and Technology*, Vol. 143, No. 2, pp. 85-90, ISSN: 1350-2344.

Wu, L. & Chen, S. (2000). Intelligent technologies for welding, National Defence Indus-trial Press of China, ISBN: 7-118-02299-3, Beijing.

Xu, D.; Jiang, Z.; Wang, L. & Tan M. (2004a). Features extraction for structured light image of welding seam with arc and splash disturbance. *Proceedings of 8th International Conference on Control, Automation, Robotics and Vision*, pp. 1559-1563, ISBN: 0-7803-8653-1, Kunming, China, Dec. 2004, Institute of Electrical and Electronics Engineers Inc., New York, United States.

Xu, D.; Wang, L. & Tan, M. (2004b). Image processing and visual control method for arc welding robot, *IEEE International Conference on Robotics and Biomimetics*, pp. 727-732, ISBN: 0780386418, Shenyang, China, Aug. 2004, Institute of Electrical and Electronics Engineers Computer Society, Piscataway, United States.

Xu, D. & Tan, M. (2004). A calibration method for hand-eye system of arc welding robot, *10th IEEE International Conference on Methods and Models in Automation and Robotics*, pp. 903-908, ISBN: 83-88764-8, Miedzyzdroje, Poland, Aug. 2004, Institute of Control Engineering, Technical University of Szczecin, Szczecin, Poland.

Xu, D.; Wang, L.; Tu, Z. & Tan, M. (2005). Hybrid visual servoing control for robotic arc welding based on structured light vision, *Acta Automatica Sinica*, Vol. 31, No. 4, pp. 596-605, ISSN: 0254-4156.

Yoshimi, B. H. & Allen, P. K. (1994). Active uncalibrated visual servoing. *IEEE International Conference on Robotic & Automation*, Vol. 4, pp.156-161,

ISBN: 0-8186-5332-9, San Diego, USA, May 1994, IEEE Press, Piscataway, USA.

Zhang, J. & Djordjevich, A. (1999). Study on laser stripe sensor. *Sensors and Actuators A: Physical*, Vol. 72, No. 3, pp. 224-228, ISSN: 0924-4247.

Zhang, Z. (2000) A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 11, pp. 1330-1334, ISSN 0162-8828.

Zhu, S. & Qiang X. (2000). Analysis of 3-D coordinate vision measuring methods with feature points on workpiece. *Optics and Precision Engineering*, Vol. 8, No. 2, pp. 192-197, ISSN: 1004-924X.

Zou, D.; Ye, S. & Wang, C. (1995). Structured-lighting surface sensor and its calibration. *Optical Engineering*, Vol. 34, No. 10, pp. 3040-3043, ISSN: 0091-3286.

Visual Conveyor tracking in High-speed Robotics Tasks

Theodor Borangiu

1. Introduction

The chapter presents two methods and related motion control algorithms for robots which are required to pick "on-the-fly" objects randomly moving on conveyor belts; the instantaneous location of moving objects is computed by the vision system acquiring images from a stationary, down looking camera. The algorithms for visual tracking of conveyor belts for moving object access are partitioned in two stages: (i) visual planning of the instantaneous destination of the robot, (ii) dynamic re-planning of the robot's destination while tracking the moving objects.

In the first method one assumes that conveyors are configured as external axes of the robot, which allows their modelling by means of a special class of variables called belt variables. A belt variable is here considered as a relative transformation (having a component variable in time) defining the location of a reference frame attached to the moving belt conveyor. By composing this time variable transformation (it reflects the contents of the belt's encoder) with the time – invariant instantaneous object location (estimated by vision for each object and compensated by the encoder offset value), the motion control algorithm will operate with a periodically updated destination, hence the robot will track the object moving on the belt.

In the second method the ensemble conveyor belt-actuator-sensor is configured as a $m \leq 3$-axis Cartesian robot, leading thus to a problem of cooperation between multiple robot manipulators subject to the multitasking control of a computer. Conceptually, the problem is solved by defining a number of *user tasks* which attach two types of "robots": the n – d.o.f. manipulator responsible with grasping on-the-fly objects moving on the conveyor belt, and the $m \leq 3$-axis robot emulating the conveyor belt under vision control. These user tasks run concurrently with the internal system tasks of a multitasking robot controller, mainly responsible for trajectory generation, axis servoing and system resources management.

Both methods use the concept of Conveyor Belt Window to implement fast reaction routines in response to emergency situations. The tracking algorithms

also provide collision-free object grasping by modelling the gripper's finger-prints and checking at run time whether their projections on the image plane cover only background pixels.

2. Modelling conveyors with belt variables

The problem of interest consists in building up a software environment allowing a robot controller to estimate the instantaneous position of the conveyor belt on which parts are travelling. The conveyor must be equipped with a displacement measuring device, in this case an *encoder*.
There are no constraints with respect to the position and orientation of the conveyor relative to the working area of the robot; the only requirement is that the belt's motion follows a straight line within the robot's manipulability region (Schilling, 1990). The encoder data will be interpreted by the controller as the current displacement of one of its external robot axes – in this case the tracked conveyor belt.

2.1 The special class of belt variables

The mechanism which allows specifying robot motions relative to a conveyor belt consists into modelling the belt by defining a special type of location data, named *belt variables*.

Definition 5.1: A *belt variable* is a relative homogenous transformation (having a component variable in time) which defines the location of a conveniently chosen reference frame attached to the conveyor's moving belt. The assignment of a belt variable is based on the software operation

DEFBELT %belt_variable = nominal_trans,scale_factor,

where:

- *%belt_variable* is the name of the belt variable to be defined, expressed as a 6-component homogenous transformation in minimal representation of the frame's orientation (e.g. by the Euler angles yaw, pitch and roll).
- *nominal_trans* represents the value in R^6 of the relative transformation defining the position and the orientation of the conveyor belt. The *X* axis of *nominal_trans* indicates the direction of motion of the belt, the *XY* plane defined by this transformation is parallel to the conveyor's belt surface, and the position (*X,Y,Z*) specified by the transformation points to the approximate centre of the belt relative to the base frame of the robot. The origin of *nominal_trans* is chosen in the middle of the robot's working displacement over the conveyor.

- *scale_factor* is the calibrating constant specifying the ratio between the elementary displacement of the belt and one pulse of the encoder.

Using such a belt variable, it becomes possible to describe the relationship between a belt encoder and the location and speed of the reference frame (conveniently chosen with respect to the manipulability domain of the robot accessing the belt) which maintains a fixed position and orientation relative to the belt (Borangiu, 2004). The informational model of the conveyor belt is its assigned belt variable, to which additional modelling data must be specified for robot-vision compatibility:

- *window parameters*, defining the working area of the robot over the conveyor belt;
- *encoder offset*, used to adjust the origin of the belt's reference frame (e.g. relative to the moment when image acquisition is triggered).

The current orientation data in a belt variable is invariant in time, equal to that expressed by *nominal_trans*. In order to evaluate the current location updated by the same belt variable, the following real-time computation has to be performed: multiplication of a unit vector in the direction of $X_{|\text{nominal\_trans}}$ by *belt_distance* – a distance derived from the encoder's contents (periodically read by the system), and then addition of the result to the position vector of *nominal_trans*. The symbolic representation of this computation is given in equations (1) and (2):

$$XYZ_{\text{instantaneous}} = XYZ_{\text{nominal}} + belt\_distance * \text{unit\_vect}(X_{|\text{nominal\_trans}}) \tag{1}$$

$$belt\_distance = (encoder\_count - encoder\_offset) * scale\_factor \tag{2}$$

Here, *encoder_count* is the encoder's read contents and *encoder_offset* will be used to establish the instantaneous location of the belt's reference frame (x_i, y_i) relative to its nominal location (x_n, y_n). In particular, the belt's offset can be used to nullify a displacement executed by the conveyor (by setting the value of the offset to the current value of the encoder's counter). The designed algorithm for visual robot tracking of the conveyor belt accounts for variable belt offsets which are frequently changed by software operations using mathematical expressions, like that included in the following V+ syntax: SETBELT %belt_variable = expression.
When the conveyor belt position is computed by referring to its assigned belt variable, the previously defined encoder offset will be always subtracted from the current position of the belt, i.e. from the encoder's current accumulated content. In the discussed method, setting a belt offset will use the real-valued

function BELT %belt_variable,mode to effectively reset the belt's position [encoder pulses].

Example 1:
A reaction routine continuously monitors a fast digital-input interrupt line which detects the occurrence of an external event of the type: "an object has completely entered the Conveyor Belt Window – and hence is completely visible". This event is detected by a photocell, and will determine an image acquisition of the moving object. The very moment the detected object is recognised as an object of interest and successfully located, the position of the belt is reset and consequently the belt variable will encapsulate from now on the current displacement of the belt relative to the belt's position in which the object has been successfully located. The V+ code is given below:

```
trigger = SIG(1001)         ;signal from photocell
save = PRIORITY  ;current priority of the robot-vision task
snap = 0                     ;reset event detection after image acquisition
success = 0          ;reset indication of successful part location
REACT –trigger,acquisition(snap,success,$name,belt_offset)
TIMER 1 = 0                  ;reset "timeout"-valued timer
IF TIMER(1)>timeout AND NOT snap  THEN
     GOTO l_end          ;no incoming parts, exit the task
 ELSE
LOCK PRIORITY + 2      ;raise priority to lock out any signals from the
        ;photocell until the current object is treated
IF success == 1  THEN
SETBELT %belt = belt_offset
IF $name == "PART"  THEN    ;if the object is of interest
Tracking the belt such that the robot picks on-the-fly the object (modelled
with the name "part") which was successfully located by vision in vis.loc
...
END
LOCK save ;re activate the REACT mechanism to check for on-off
;for on-off signal #1001 transitions
END
```

The interruption routine, automatically called by the REACT mechanism, has the form:

```
.PROGRAM acquisition(snap,success,$name,belt_offset)

VPICTURE (cam) –1,1    ;image acquisition and recognition of one object
snap = 1
;Locate any type of recognised object, return its name in the string
```

;var. $name and its location relative to the vision frame in vis_loc

```
VLOCATE (cam,0) $name,vis.loc
success = VFEATURE(1)          ;evaluate the success of the locating op.
belt_offset = VFEATURE(8)
```

;The real-valued function VFEATURE(8) returns the contents of the
;belt's encoder when the strobe light for image acquisition was
;triggered.

```
RETURN
.END
```

In what concerns the encoder's scale factor represented by the constant parameter *scale_factor*, it can be evaluated:

- either theoretically, knowing the mechanical coupling belt-encoder,
- or experimentally by reading the encoder's contents twice, each time when the image acquisition is triggered for a circular disk the presence of which is detected by the belt's photocell. The distance at which travel the two identical disks on the conveyor belt has been upstream set at a convenient, known value (see Fig. 1).
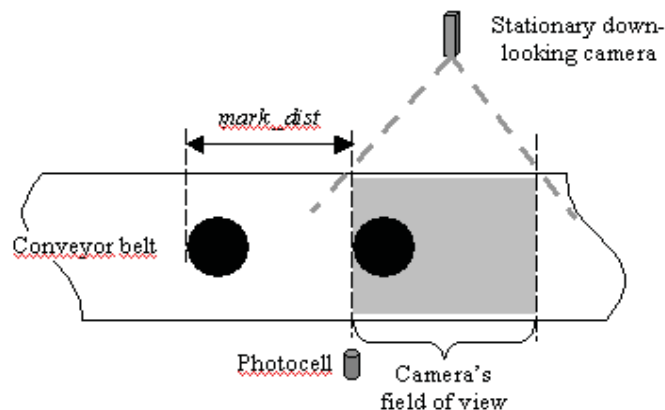
Figure 1. The experimental setup for conveyor belt calibration

2.2 The logical mechanism "Conveyor Belt Window" and emergency routines

There has been designed a logical mechanism called Conveyor Belt Window (CBW) which allows the user to check the area of the belt in which the robot will operate. A CBW defines a segment of the belt delimitated by two planes perpendicular to the direction of motion of the belt (this window is restricted only in the direction of motion of the conveyor's belt) (Borangiu & Kopacek, 2004).

Because the conveyor is modelled by a belt variable (e.g. *%belt_var*), in order to define a CBW it is necessary to refer the same belt variable and to specify two composed transformations which, together with the direction of motion of the belt, restrict the motion of the robot along a desired area of the conveyor:

WINDOW %belt_var = downstr_lim,upstr_lim,program_name,priority,

where:

- *downstr_lim* and *upstr_lim* are respectively the relative transformations defining the downstream and upstream edges of an invariant window positioned along the belt within the working space of the robot *and* the image field of the camera (it is necessary that the robot tracks and picks parts within these two limits);
- *program_name* indicates the reaction routine to be automatically called, whenever a window violation occurs while the robot tracks the conveyor belt;
- *priority* is the level of priority granted to the reaction routine. Normally, it must be greater than that of the conveyor tracking program, so that the motion of the robot can be immediately interrupted in case of window violation.

The CBW will be used not only in the stage of robot motion planning, but also at run time, during motion execution and tracking control, in order to check if the motion reference (the destination) is within the two imposed limits:

- When *a robot movement is planned*, the destination of the movement is checked against the operating CBW; if a window violation is detected, the reaction program is ignored and an error message will be issued.
- When *a robot movement relative to the conveyor belt is executed*, the destination is compared every 8 milliseconds with the window's limits; if a window violation is detected, the reaction program is automatically invoked according to its priority level and the robot will be instructed to stop tracking the belt.

There have been designed two useful CBW functions which allow the dynamic reconfiguring of programs, decisions, branching and loops during the execution of robot – vision conveyor tracking programs, function of the current value of the part-picking transformation relative to the belt, and of the current status of the belt tracking process. These functions are further introduced.
The function WINTEST(robot_transformation,time,mode) returns a value in millimetres indicating where is situated the location specified by the belt-relative composed transformation *robot_transformation*, with respect to the fixed window limits *downstr_lim* and *upstr_lim* at *time* seconds in the future, computed according to its current position and belt speed.

Finally, the argument *mode* is a real-valued expression which specifies whether the result of the WINTEST function represents a distance inside or outside the predefined conveyor belt window. For example, if *mode* is positive, the value returned by WINTEST will be interpreted as:

0: the composed, belt-relative location is inside the CBW;

<0: the location is upstream of *upstr_lim* of the CBW;

>0, the location is downstream of the *dwnstr_lim* of the CBW.

Hence, the returned value conforms to the WINDOW model shown in Fig. 2, for which the value returned by the function WINDOW increases as the belt-relative location moves downstream.
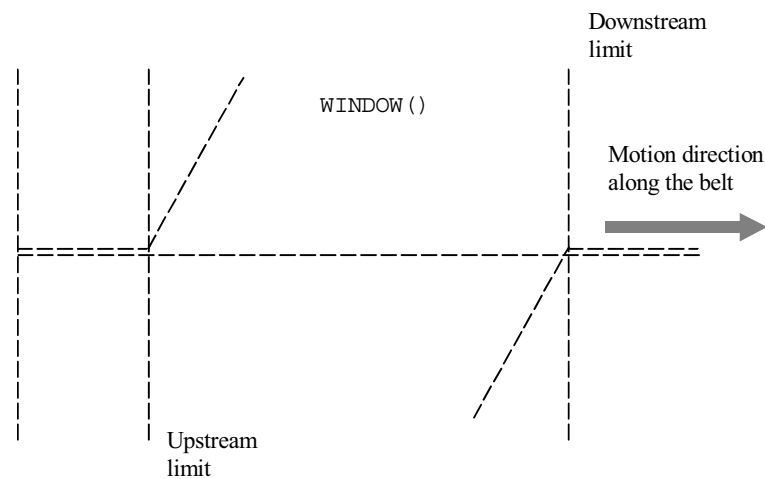
Figure 2. The WINDOW function for *mode* > 0

For robots tracking conveyor belts in order to pick moving objects recognised and located by vision, the belt-relative transformation is *%belt_var:part.loc* (variable in time), obtained by the composition of:

- *%belt_var*, models the conveyor belt,
- *part.loc*, is a composed, *time invariant* transformation expressing the gripper frame (x_g, y_g, z_g) relative to the base frame of the robot (x_0, y_0, z_0) at the moment the object was identified and located by vision.

For example, the distance WINTEST(%belt:part.loc,4,1) is positive if, in 4 seconds from the time being, the belt-relative part picking location will be outside the window defined for the conveyor belt modelled by *%belt*.

If the robot tries to move towards a belt-relative location that has not yet appeared inside the belt window (it is still upstream relative to the CBW), the motion control algorithm has been designed with two options:

- *temporarily stops the robot*, delaying thus the motion planning, until the time-variable destination enters the belt window;
- *definitively stops the robot* and generates immediately an error message.

Also, the control algorithm generates a condition of window violation anytime the vision-based robot motion planner computes a destination which is downstream the CBW, or will exit the CBW at the predicted time the robot will reach it. The function BELTSTATUS indicates the current status of the belt tracking process: *robot tracking the belt*; *destination upstream*; *destination downstream*; *window violation*, real-time information which can be used to dynamically reconfigure the robot – vision task.

2.3 Robot locations, frames and belt-relative movements planned by vision

To command the belt-relative motion of a robot with linear interpolation in the Cartesian space, i.e. to define an end-tip transformation relative to an instantaneous location of a moving frame (x_i, y_i) on the conveyor belt, the already defined belt variable (which models the conveyor belt as a relative transformation having time variable components along the *X* (and possibly *Y*) Cartesian axes) will be composed with the time-invariant end-tip transformation relative to the base of the robot (which is computed at run time by the vision part of the system).

The result will be a time-variable transformation updating the position reference for the robot. This reference or target destination tracks an object moving on the belt, to be picked by the robot's gripper. The target destination is:

- *planned once* at runtime by vision, as soon as the object is perfectly visible to the camera, either inside the manipulability area of the robot or upstream this area;

- *updated every 8 milliseconds* by the motion controller based on the current position data read from the belt's encoder, until the robot's end-point completes the necessary percentage of its motion segment towards the part's grasping location.

The research on Guidance Vision for Robots (GVR) accessing moving targets was directed to develop a *convergent* motion control algorithm for visually plan the motion of the robot as a result of object detection, recognition and locating on a moving conveyor belt, and than track the object in order to grasp it inside a conveniently defined belt window. The main idea relies on dynamically changing the visually computed destination of the robot end point by composing it with a belt-related transformation updated every 8 milliseconds from the encoder data.

If a stationary camera looks down at the conveyor belt, and supposing that its field of view covers completely a conveyor belt window defined inside the working area of the robot (after execution of a camera – robot calibration session), then the image plane can be referred by the time – invariant frame (x_{vis}, y_{vis}) as represented in Fig. 3.

It is also assumed that the X axes of the reference frame of the robot (x_0), of the conveyor's direction of motion (x_n) and of the image plane (x_{vis}) are parallel.

The conveyor belt is modelled by the belt variable %belt. Parts are circulating on the belt randomly; their *succession* (current part type entering the CBW), *distance from the central axis of the conveyor* and *orientation* are unknown. The "Look-and-Move" interlaced operating principle of the image processing section and motion control section is used (Hutchinson, 1996), (Borangiu, 2001), (West, 2001), (Adept, 2001). According to this principle, while an image of the CBW is acquired and processed for object identification and locating, no motion command is issued and reciprocally, the camera will not snap images while the robot tracks a previously located part in order to pick it "on-the-fly".
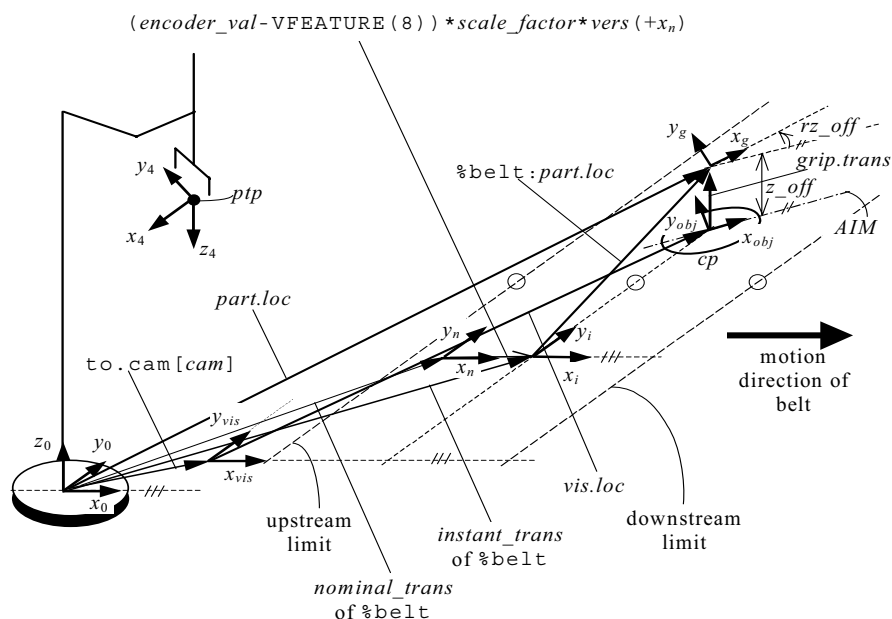
Figure 3. Robot-Vision and belt-relative transformations for conveyor tracking

The robot motion control algorithm for tracking the conveyor belt in order to pick "on-the-fly" one by one objects recognised and located by vision computing consists of the following basic steps:

1. *Triggering the strobe light* (synch./asynch. relative to the read cycle of the video camera) when *image acquisition* is requested from a fast digital-input interrupt line connected to a photocell mounted at the upstream limit of the CBW. The interrupt line signals that an object has completely entered the belt window.

2. *Recognizing a single object* that just completely entered the belt window. Object recognition is exclusively based in this approach on the match with previously learned models of all objects of interest (Lindenbaum, 1997).

3. *Locating the object* which was recognised, by computing the coordinates of its centre of mass and the angle between its minimum inertia axis (MIA) and x_{vis}. As can be seen in Fig. 3, the object-attached frame (x_{obj}, y_{obj}) has the abscissa aligned with the minimum inertia axis (MIA), and the current location of the object in the image plane is computed by the vision section and returned in *vis.loc*.

4. *Planning the instantaneous destination of the robot*. Once the object is *recognized* as the instance of a model and *located*, the related grasping transformation *grip.trans* is called. Assuming that the grasping style is such that the projection of the gripper's centre on the image plane coincides with the object's centre of mass, the gripper-attached frame (x_g, y_g) will be offset relative to the object-attached frame along z_0 by *z_off* millimetres and turned with *r_off* degrees about z_0. Now, using the relative transformation to.cam[*cam*](as output of the camera-robot calibration session) relating the vision frame $(x_{vis}, y_{vis}, z_{vis})$ to the base frame of the robot (x_0, y_0, z_0), the current destination of the robot (for a frozen conveyor belt) is computed from the vision data as a composed transformation *part.loc*, expressing the gripper frame relative to the robot base frame:

$$part.loc = to.cam[cam]:vis.loc:grip.trans$$

5. *Synchronising the encoder belt with the motion of the object* recognized in the belt window. This operation consists into setting the offset of the conveyor belt at a correct value. The operation

$$SETBELT \%belt = encoder\_val(strobe)$$

establishes the point of interest of the conveyor belt modelled with %belt as the point corresponding to the current value *encoder_val*(*strobe*) of the encoder counter at the time the strobe light was triggered. This value is available immediately after object locating. Thus, as soon as one object is recognized and located, the current belt position, identified by (x_i, y_i), will be reset since:

$$xyz_i = xyz_n + (encoder\_val - encoder\_val(strobe)) *$$
$$* scale\_factor * \text{unit\_vect}(x_n) = xyz_n \tag{3}$$

6. *Tracking and picking the object moving on the belt*. This requires issuing a linear motion command in the Cartesian space, relative to the belt. A composed relative transformation %belt:part.loc, expressing the current computed location of the gripper relative to the instantaneous moving frame (x_i, y_i), is defined. Practically, the tracking procedure begins immediately after the instantaneous position of the belt – expressed by the frame (x_i, y_i) has been initialized by the SETBELT operation, and consists into periodically updating the destination of the gripper by shifting it along the x_n axis with encoder counts accumulated during successive sampling periods $..., t_{k-1}, t_k, t_{k+1}, ...$ $\Delta t = t_{k+1} - t_k = \text{const}$:

$$\%belt : part.loc_{|t_{k+1}} = \text{SHIFT} (\%belt : part.loc_{|t_k} \text{ BY} ...$$
$$... \; encoder\_count(t_{k+1} - t_k) * scale\_factor, 0, 0) \tag{4}$$

```
MOVES %belt:part.loc        ;go towards the moving target
CLOSEI                      ;grasp "on the fly" the object
```

7. Once the robot commanded towards a destination relative to the belt, the gripper will continuously track the belt until a new command will be issued to approach a location which is not relative to the belt.

For belt-relative motions, the destination changes continuously; depending on the magnitude and the variations of the conveyor speed it is possible that the robot will not be able to attain the final positions within the default error tolerance.

In such cases, the error tolerance must be augmented. In extreme cases it will be even necessary to totally deactivate the test of final error tolerance. Fig. 4

presents the designed robot motion control algorithm for tracking the conveyor belt in order to pick "on-the-fly" an object recognized and located by vision computation inside the a priori defined belt window. A REACT mechanism continuously monitors the fast digital-input interrupt line which signals that an object has completely entered the belt window. The robot motion relative to the belt will be terminated:

- when moving the robot towards a *non belt-relative location* or
- when a *window violation* occurs.

Example 2:
The following series of instructions will move the robot end-effector towards a belt-relative location part_2 (the belt is modelled as %belt[1]), open the gripper, track the conveyor belt for 5 seconds (in fact the location part_2 on the belt), close the gripper and finally leave the belt to move towards a fixed location.

```
MOVES %belt[1]:part_2
OPENI
DELAY 5.0
CLOSEI
MOVES fixed_location
```

When defining the Conveyor Belt Window, a special high-priority routine can be specified, which will be automatically invoked to correct any window violation during the process of tracking moving objects. In such situations the robot will be accelerated (if possible) and the downstream limit temporarily shifted in the direction of motion of the conveyor belt (within a timeout depending on the belt's speed) in which the error tolerance must be reached (Espiau, 1992), (Borangiu, 2002).

```
┌─────────────────────────────────────────┐
│      Setup of virtual camera cam         │
│   Camera-Robot calibration defines:      │
│   to.cam[cam], XY_scale, pix.to.mm       │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐
│ Modelling the conveyor belt by a %belt_variable │
│  Defining nominal_trans, encoder scale_factor, │
│            belt window CBW               │
└─────────────────────────────────────────┘
```

Timeout
expired

```
┌─────────────────────────────────────────┐      No event
│ REACT to fast digital-input interrupt line │
│  from photocell: "object inside CBW"     │
└─────────────────────────────────────────┘
```

Event-driven interrupt

```
┌─────────────────────────────────────────┐
│  Triggering the strobe light and image acquisition │
└─────────────────────────────────────────┘
```

Failure or
object of no
interest

```
┌─────────────────────────────────────────┐
│  Single object recognition by prototype match │
└─────────────────────────────────────────┘
```

Instance of prototype "OBJ"

```
┌─────────────────────────────────────────┐      No
│  "OBJ" is successfully located (x_c,y_c,AIM) │
│         and vis.loc is returned          │
└─────────────────────────────────────────┘
```

Yes

```
┌─────────────────────────────────────────┐
│ Setting the offset of the conveyor belt to encoder_val(strobe) │
│              xyz_i = xyz_n               │
└─────────────────────────────────────────┘
```

```
┌─────────────────────────────────────────┐
│ Calling the grasping transformation grip.trans for "OBJ" and │
│  planning the instantaneous picking transformation part.loc: │
│   part.loc=to.cam[cam]:vis.loc:grip.trans │
└─────────────────────────────────────────┘
```

```
┌──────────────────────────┐   Destination        Destination
│  Delaying motion tracking │   inside the CBW     downstream CBW
└──────────────────────────┘
```

Destination
upstream CBW

```
            ⬭ Read encoder_count(Δt) ⬭
```

```
┌─────────────────────────────────────────┐
│ %belt:part.loc ←SHIFT(%belt:part.loc BY │
│   encoder_count(Δt)*scale_factor,0,0)    │
│  Robot moving towards time-variable destination: │
│        MOVES %belt:part.loc              │
└─────────────────────────────────────────┘
```

```
┌──────────────────────────────────┐  No  ┌──────────────┐
│ Error tolerance entered for target │─────▶│ t = t + Δt   │
└──────────────────────────────────┘      └──────────────┘
```

Yes

```
┌─────────────────────────────────────────┐
│  Pick part "on-the-fly" : CLOSEI and     │
│  move to a non belt-relative destination  │
└─────────────────────────────────────────┘
```
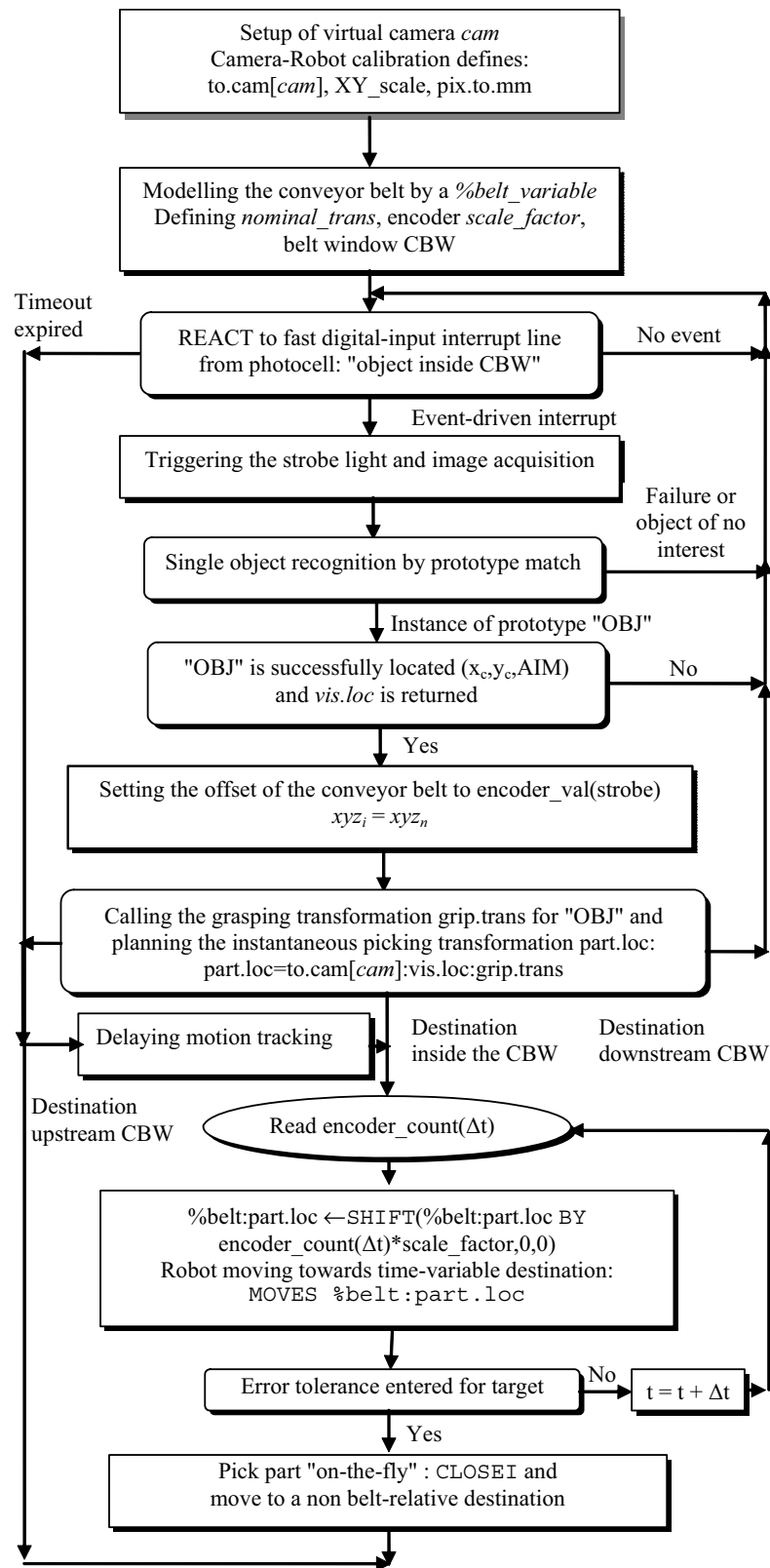
Figure 4. The robot motion algorithm for visual tracking of the conveyor belt

3. Tracking conveyors as $m{\leq}3$ Cartesian axis robots

According to the second tracking method, the ensemble conveyor belt + actuator + sensor is configured as an $m \leq 3$-axis Cartesian robot, which leads to a problem of cooperation between multiple robots subject to multitasking computer control. The V+ structured programming environment is used for exemplifying the multi tasking control of robots visually tracking moving objects on multiple belts.

3.1 Multitasking control for robot cooperation

Conceptually, the problem is solved by defining a number of *user tasks* which attach two types of "robots": the n – d.o.f. manipulator responsible with grasping on-the-fly objects moving on the conveyor belt, and the $m \leq 3$-axis robot emulating the conveyor belt under vision control. These user tasks run concurrently with the internal system tasks of a multitasking robot controller, mainly responsible for trajectory generation, axis servoing and system resource management (Adept, 2001).

In this respect, there are three tasks to be at least defined for the tracking problem:

1. Task 1: Dynamic re-planning the destination location (grasping the moving object) for the robot manipulator.
2. Task 2: Continuously moving (driving) the $m \leq 3$-axis vision belt. In the most general case, the belt moves along any 3D-direction relative to the robot base frame (x_0, y_0, z_0).
3. Task 3: Reading *once* the belt's location the very moment an object of interest has been recognised, located and its grasping estimated as collision-free, and then *continuously* until the object is effectively picked.

3.1.1 Specifying tasks, time slices and priorities

A multitasking robot control system appears to execute all these program tasks at the same time. However, this is actually achieved by rapidly switching between the tasks many times each second, each task receiving a fraction of the total time available. This is referred to as concurrent execution (Zhuang, 1992), (Borangiu, 2005).

The amount of time a particular program task receives is caused by two parameters: its *assignment* to the various time slices, and its *priority* within the time slice. One assumes that, in the multitasking operating system, each *system cycle* is divided into 16 *time slices* of one millisecond each, the slices being numbered 0 through 15. A single occurrence of all 16 time slices is referred to

as a *major cycle*. For a robot each of these cycles corresponds to one output from the trajectory generator to the servos.

A number of seven user tasks, e.g. from 0 to 6, will be used and their configuration tailored to suit the needs of specific applications. Each program task configured for use requires dedicated memory, which is not available to user programs. Therefore, the number of tasks available should be made no larger than necessary, especially if memory space for user programs is critical.

When application programs are executed, their program tasks are normally assigned default time slices and priorities according to the current system configuration. The defaults can be overridden temporarily for any user program task, by specifying the desired time slice and priority parameters of the EXECUTE initiating command.

Tasks are scheduled to run with a specified priority in one or more time slices. Tasks may have priorities from –1 to 64, and the priorities may be different in each time slice. The priority meanings are:

–1	Do not run in this slice even if no other task is ready to run.
0	Do not run in this slice unless no other task from this slice is ready to run.
1-64	Run in this slice according to specified priority. Higher priority tasks may lock lower ones. Priorities are broken into the following ranges:
1-31	Normal user task priorities;
32-62	Used by robot controller's *device drivers* and *system tasks*;
63	Used by *trajectory generator*. Do not use 63 unless you have very short task execution times, because use of these priorities may cause jerks in the robot trajectories;
64	Used by the *servo*. Do not use 64 unless you have very short task execution times, because use of these priorities may cause jerks in the robot trajectories.

The V+ operating system has a number of *internal* (system) *tasks* that compete with *application* (user) *program tasks* for time within each time slice:

- On motion systems, the V+ *trajectory generator* runs (at the highest priority task) in slice #0 and continues through as many time slices as necessary to compute the next motion device set point.

- On motion systems, the CPU running servo code runs the *servo task* (at interrupt level) every 1 or 2 milliseconds (according to the controller configuration utility).

The remaining time is allocated to user tasks, by using the controller configuration utility. For each time slice, you specify which tasks may run in the slice and what priority each task has in that slice.

3.1.2 Scheduling of program execution tasks

Vision guided robot planning ("object recognition and locating"), and dynamical re-planning of robot destination ("robot tracking the belt") should always be configured on user tasks 0 or 1 in "Look-and-Move" interlaced robot motion control, due to the continuous, high priority assignment of these two tasks, over the first 13 time slices. However, vision guidance and motion re-planning programs complete their computation in less than the 13 time slices (0-12). Consequently, in order to give the chance to conveyor-associated tasks ("drive" the vision belt, "read" the current position of the vision belt") to provide the "robot tracking" programs with the necessary position update information earlier than the slice 13, and to the high-priority trajectory generation system task to effectively use this updates, a WAIT instruction should be inserted in the loop-type vision guidance and motion re-planning programs of tasks 0 and/or 1.

A WAIT *condition* instruction with no argument will suspend then, once per loop execution, the motion re-planning program, executing on user task 1, until the start of the next major cycle (slice 0). At that time, the "vision processing and belt tracking" task becomes runnable and will execute, due to its high priority assignment.

Due to their reduced amount of computation, programs related to the management of the conveyor belt should be always assigned to tasks 2, 3, 5 or 6 if the default priority scheme is maintained for user program tasks, leaving tasks 1 and 2 for the intensive computational vision and robot motion control.

Whenever the current task becomes inactive, the multitasking OS searches for a new task to run. The search begins with the highest priority task in the current time slice and proceeds through in order of descending priority. If multiple programs wait to run in the task, they are run according to relative program priorities. If a runnable task is not found, the next higher slice is checked. All time slices are checked, wrapping around from slice 15 to slice 0 until the original slice is reached. Whenever a 1 ms interval expires, the system performs a similar search of the next time slice; if this one does not contain a runnable task, the currently executing task continues.

If more than one task in the same time slice has the same priority, they become part of a *round-robin scheduling group*. Whenever a member of a round-robin group is selected by the normal slice searching, the group is scanned to find the member of the group that run most recently. The member that follows the most recent is run instead of the one which was originally selected.

The V+ RELEASE program instruction may be used to bypass the normal scheduling process by explicitly passing control to another task. That task then goes to run in the current time slice until it is rescheduled by the 1 ms clock. A task may also RELEASE to *anyone*, which means that a normal scan is made of all other tasks to find one that is ready to run. During this scan, members of the original task's round-robin group (if any) are ignored. Therefore, a RELEASE to anyone cannot be used to pass control to a different member of the current group.

Round-robin groups are treated as a single task. If any member of the group is selected during the scan, then the group is selected. The group is scanned to find the task in the group following the one which ran most recently, and that task is run. Within each time slice, the task with highest priority can be locked out only by a servo interrupt. Tasks with lower priority, defined for driving the conveyor belt and reading position data from its encoder, can run only if the higher-priority task, defined for vision guidance of the n–d.o.f. robot and for tracking the 1–d.o.f. robot-like conveyor belt, is inactive or waiting. A user task waits whenever:

- The program issues an input or an output request that causes a wait.
- The program executes a robot motion instruction while the robot is still moving in response to a previous motion instruction.
- The program executes a WAIT or WAIT.EVENT program instruction.

If a program is executing continuously without performing any of the above operations, it locks out any lower-priority tasks in its time slice. Thus, programs that execute in continuous loops, like vision guidance and motion re-planning for belt tracking, should generally execute a WAIT (or WAIT.EVENT) instruction occasionally (for example, *once each time through the loop*).

If a program potentially has a lot of *critical processing* to perform, its task should be in *multiple slices*, and the task should have the *highest priority* in these slices. This will guarantee the task's getting all the time needed in the multiple slices, plus (if needed) additional unused time in the major cycle.

Fig. 5 shows the *task scheduler algorithm* which was designed for an n-d.o.f. robot tracking a continuously updated object grasping location, and picking the object "on-the-fly" from a conveyor belt, when motion completes. The object is recognized and located by vision, and updating of its position is provided by encoder data reads from the conveyor belt modelled as a 1-d.o.f. robot. The priority analysis and round-robin member selection are also indicated.

The problem of conveyor tracking with vision guiding for part identification and locating required definition of three user tasks, to which programs were associated:

1. <u>Task 1</u>: program "**track**" executes in this task, with robot 1 (4-d.o.f. SCARA) selected. This program has two main functions, carried out in a 2 – *stage* sequence:

   STAGE 1: Continuous *checking* whether an object travelling on the *vision belt* entered the field of view of the camera and the reachable workspace of the SCARA robot. If such an event occurs, the vision is activated to *identify* whether the object is of interest and to *locate* it. Processing on this stage terminates with the *computation of the end-effector's location* which would move the robot in the object picking location evaluated once by vision, according to a predefined grasping style, if the belt were stopped.

   STAGE 2: Continuously *re-planning* the end-effector's location, computed once by vision, by *consuming the belt position data* produced by encoder reads in program "**read**" which executes on task 3, and by *dynamically altering the robot's target* in the current motion segment.

2. <u>Task 2</u>: program "**drive**" executes in this task, with robot 2 (the 1-d.o.f. conveyor belt) selected. This program *moves the belt* in linear displacement increments, at a sufficiently high rate to provide a jerk-free, continuous belt motion. This program executes in both stages of the application, previously defined.

3. <u>Task 3</u>: program "read" executes in this task, with robot 2 selected. This program executes differently in the two stages of the application:
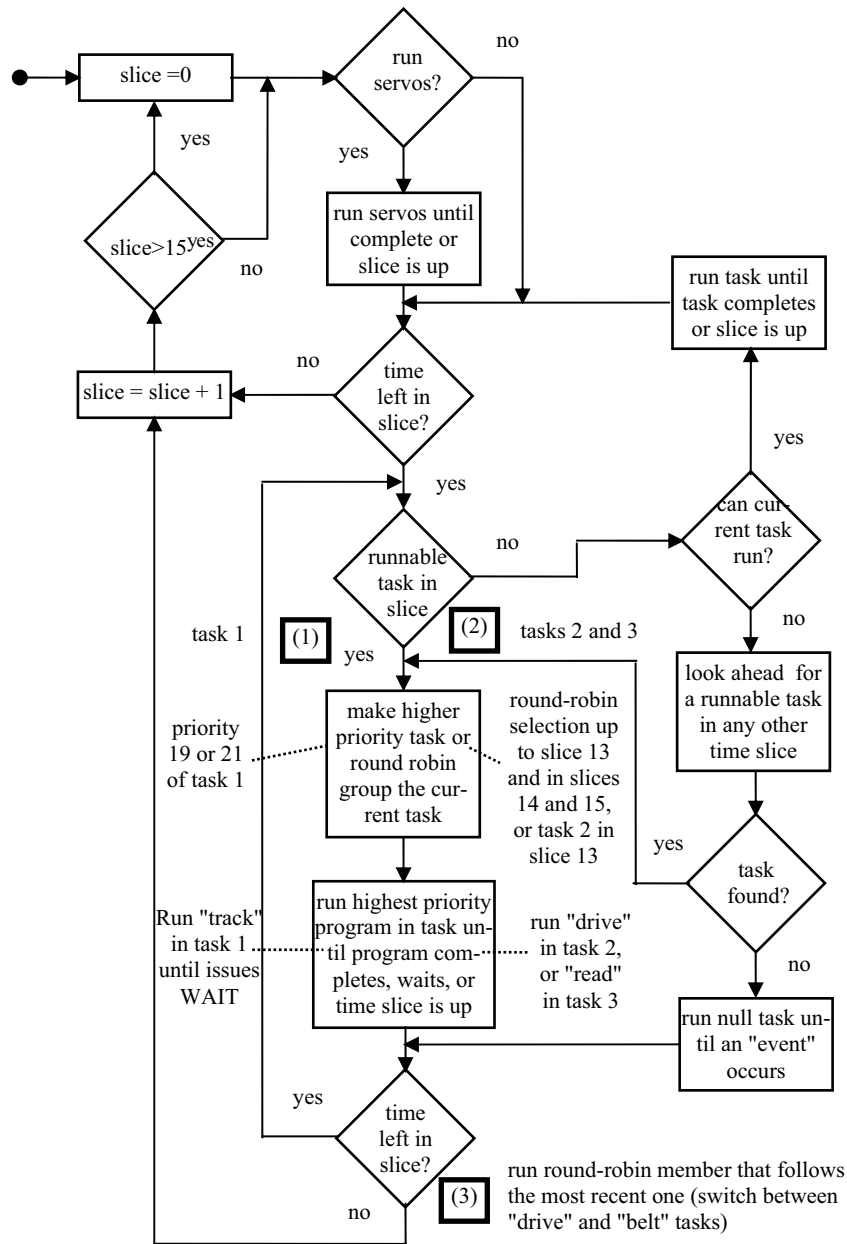
Figure 5. Task scheduler for multitasking robot control and conveyor tracking

STAGE 1: Executes *a single time* upon receiving an input signal from vision in task 1, confirming the recognition and locating of a part. In response, "**drive**" *reads the instantaneous belt position*, which from now on will be used as an offset for the position updates.

STAGE 2: Continuously *reads the belt position,* upon a request issued by "**track**" in task 1, when it starts its dynamic target re planning process.

From the three user tasks, the default priority assignment is maintained. This leads to the following priority analysis for a major cycle:

- Task 1 has the highest priority in time slices 0 – 12 (inclusively), with values of 19, 21, 9 and 11.
- Task 2 has the highest priority (20) in a single time slice: 13.
- Task 3 never detains explicitly a position of highest priority with respect to tasks 1 and 2.

The three tasks become part of a round-robin group as follows: tasks 2 and 3 in slices 0 – 12 inclusively; tasks 1, 2 and 3 in slices 14 and 15. Because tasks 2 and 3 are in more than one round-robin group on different slices, then all three tasks in the corresponding pairs of different slices appear to be in a big group. As a result of the priority scan and scheduling, the programs in the three user tasks execute as follows:

STAGE 1 – vision is processing, the robot is not moving and no WAIT is issued by task 1 (Fig. 6):

– Task 1 runs: in slices 0 – 12 (it detains the highest priority), in slice 14 (it is member of the round-robin group following task 2 that run more recently – in slice 13) only *before generating the request* for task 3 to compute the instantaneous offset belt position when vision located the object, and in slice 15 only *after generating this request* (it is member of the round-robin group following task 3 that run more recently – in slice 14).

– Task 2 runs in slice 13 (it detains the highest priority), and in slice 15 (it is member of the round-robin group following task 1 that run more recently – in slice 14) only *before* task 1 *generates the request* for task 3 to compute the instantaneous offset belt position when vision located the object.

– Task 3 runs in slice 14 (it is member of the round-robin group following task 2 that run more recently – in slice 13) only *after receiving the request* from task 1 to compute the instantaneous offset belt position when vision located the object.
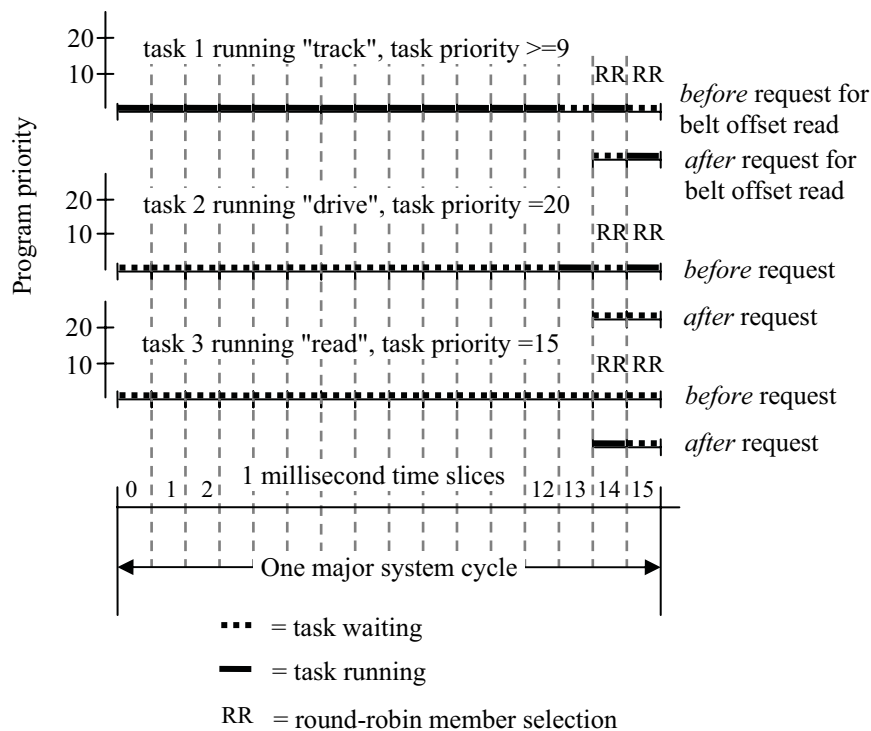
Figure 6. Priority assignment and tasks running in STAGE 1 of vision guidance for robot motion planning

STAGE 2 – *vision is not processing, the* SCARA *robot is moving and* WAIT *commands are issued in* task 1 *by the "***track***" program after each re-planning of the end-effector's target destination within one major cycle of 16 milliseconds* (Fig. 7):

– Task 1 runs in slices $i - j$, $i \le j, i \ge 0, j \le 12$, (when it detains the highest priority), i.e. starting with the moment when it is authorized to run by the highest-priority system tasks "**trajectory generation**" and "**servo**" (in slice i), and executing until it *accesses the position update* provided by task 3 *from the most recent belt encoder read, alters the last computed end-effector destination* and issues a WAIT (in slice j), to give the trajectory generator a chance to execute. From this time moment, task 1 becomes inactive for the rest of the 16 milliseconds of the current major cycle, until slice 0 of the next system cycle when it waits to be selected by the scheduler and authorized to run.

– Task 2 runs: in slices $(j + 1) - 12$ switching alternatively with task 3 whenever it is selected as the member of the round-robin group following task 3 that run most recently, in slice 13 (it detains the highest priority), and in slice 15 (it is member of the round-robin group following task 3 that run more recently – in slice 14). Task 2 runs always exactly for

1 millisecond whenever selected, so that the round-robin group scanning authorises task 3 to run always at the beginning of the next time slice.

– Task 3 runs in slices $(j + 1) - 12$ switching alternatively with task 2 whenever it is selected as the member of the round-robin group following task 2 that run most recently, and in slice 14 (it is member of the round-robin group following task 2 that run more recently – in slice 13). The task 3 runs, whenever selected, for less than 1 millisecond and issues a RELEASE "to anyone" in the current slice, allowing selection of task 2 (in the same round-robin group) for running exactly at the beginning of the next time slice.
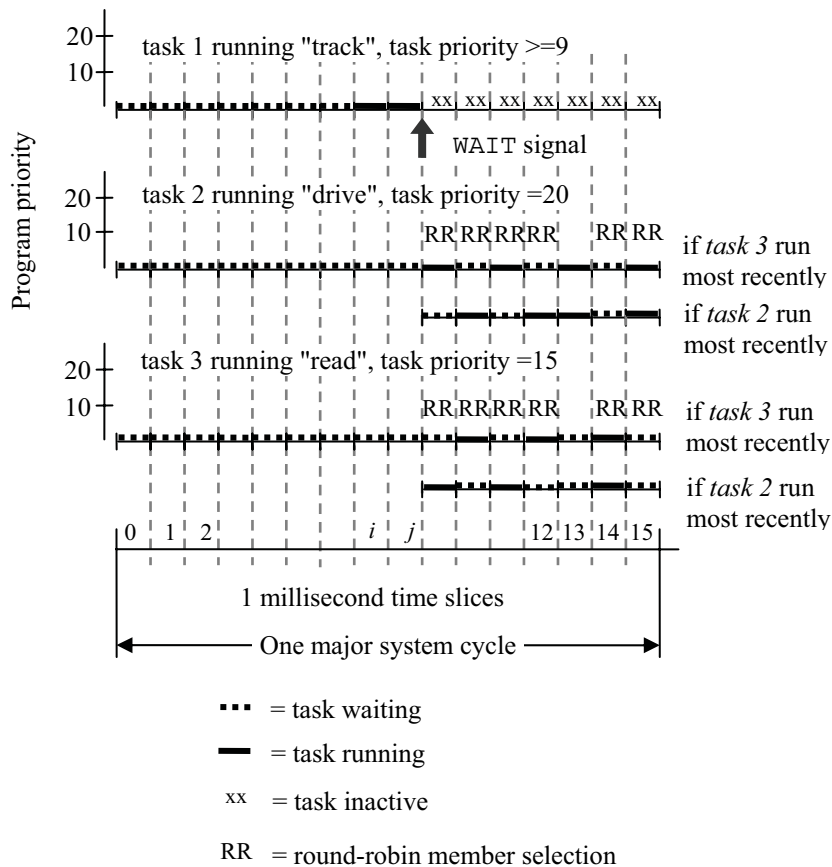

Figure 7. Priority assignment and tasks running in STAGE 2 of robot motion re-plan

3.2 Dynamically altering belt locations for collision-free object picking on-the-fly

The three previously discussed user tasks, when runnable and selected by the system's task scheduler, attach respectively the robots:

- Task 1: **robot 1** – a SCARA-type robot Cobra 600TT was considered;
- Task 2 and 3: **robot 2** – the "vision conveyor belt" of a flexible feeding system.

In multiple-robot systems like the one for conveyor tracking, SELECT robot operations select the robot with which the current task must communicate. The SELECT operation thus specifies which robot *receives motion instructions* (for example, DRIVE to move the *vision belt* in program "**drive**", or MOVES to move the SCARA in program "**track**") and *returns robot-related information* (for example, for the HERE function accessing the current vision belt location in program "**read**").

Program "**track**" executing in task 1 has two distinct timing aspects, which correspond to the partitioning of its related activities in STAGE 1 and STAGE 2. Thus, during STAGE 1, "**track**" waits first the occurrence of the on-off transition of the input signal generated by a photocell, indicating that an object passed over the sensor and will enter the field of view of the camera. Then, after waiting for a period of time (experimentally set up function of the belt's speed), "**track**" commands the vision system to acquire an image, identify an object of interest and locate it.

During STAGE 2, "**track**" alters continuously, once each major 16 millisecond system cycle, the target location of the end-effector – part.loc (computed by vision) by composing the following relative transformations:

SET part.loc = to.cam[1]:vis.loc:grip.part

where grip.part is the learned grasping transformation for the class of objects of interest. The updating of the end-effector target location for grasping one moving object uses the command ALTER()Dx,Dy,Dz,Rx,Ry,Rz, which specifies the magnitude of the real-time path modification that is to be applied to the robot path during the next trajectory computation. This operation is executed by "**track**" in task 1 that is controlling the SCARA robot in *alter mode*, enabled by the ALTON command. When *alter* mode is enabled, this instruction should be executed once during each trajectory cycle. If ALTER is executed more often, only the last set of values defined during each cycle will be used. The arguments have the meaning:
- Dx,Dy,Dz: optional real values, variables or expressions that define the translations respectively along the X, Y, Z axes;

- Rx,Ry,Rz: optional real values, variables or expressions that define the rotations respectively about the X, Y, Z axes.

The ALTON mode operation *enables real-time path-modification mode* (alter mode), and specifies the way in which ALTER coordinate information will be interpreted. The value of the argument mode is interpreted as a sequence of two bit flags:

Bit 1 (LSB):   If this bit is set, coordinate values specified by subsequent ALTER instructions are interpreted as incremental and are accumulated. If *this bit is clear*, each set of coordinate values is interpreted as the *total* (non cumulative) *correction to be applied*. The program "**read**" executing in task 3 provides at each major cycle the updated position information y_off of the robot 2 – the vision belt along its (unique) *Y* motion axis, by subtracting from the current contents *pos* the belt's offset position *offset* at the time the object was located by vision: y_off = *pos – offset*. The SCARA's target location will be altered therefore, in non cumulative mode, with y_off.

Bit 2 (MSB):   If *this bit is set*, coordinate values specified by the subsequent ALTER instructions are interpreted to be *in the World coordinate system*, to be preferred for belt tracking problems.

It is assumed that the axis of the vision belt is parallel to the Y_0 robot axis in its base. Also, it is considered that, following the belt calibrating procedure described in Section 2.1, the coefficient *pulse.to.mm*, expressing the ratio between one belt encoder pulse and one millimetre, is known.

The repeated updating of the end-effector location by altering the part.loc object-grasping location proceeds in task 1 by "**track**" execution, until motion stops at the (dynamically re-) planned grasping location, when the object will be picked-on-the-fly (Borangiu, 2006). This stopping decision is taken by "**track**" by using the STATE (select) function, which returns information about the state of the robot 1 selected by the task 1 executing the ALTER loop. The argument select defines the category of state information returned. For the present tracking software, the data interpreted is "Motion stopped at planned location", as in the example below:

Example 3:

The next example shows how the STATE function is used to stop the continuous updating of the end-effector's target location by altering every major cycle the position along the *Y* axis. The altering loop will be exit *when motion stopped at planned location*, i.e. when the robot's gripper is in the desired picking position relative to the moving part.

```
ALTON () 2                    ;Enable altering mode

MOVES part.loc               ;Robot commanded to move in grasp location
                             ;computed by vision (VLOCATE)

WHILE STATE(2)<>2 DO         ;While the robot is far from the moving
                             ;target (motion not completed at planned
                             ;location

ALTER () ,-pulse.to.mm*y_off ;Continuously alter the
                             ;target grasping location

WAIT                         ;Wait for the next major time cycle to give the
                             ;trajectory generator a chance to execute

END

ALTOFF                       ;Disable altering mode

CLOSEI                       ;Robot picks the tracked object

DEPARTS                      ;Robot exist the belt tracking mode

MOVES place                  ;Robot moves towards the fixed object-
                             placing loc
```

After alter mode terminates, the robot is left at a final location that reflects both the destination of the last robot motion and the total ALTER correction that was applied.

Program "**drive**" executing in task 2 has a unique timing aspect in both STAGES 1 and 2: when activated by the main program, it issues continuously motion commands DRIVE joint,change,speed, for the individual joint number 1 of robot 2 – the vision belt (changes in position are 100 mm; several speeds were tried).

Program "**read**" executing in task 3 evaluates the current motion of robot 2 – the vision belt along its single axis, in two different timing modes. During STAGE 1, upon receiving from task 1 the info that an object was recognised, it computes the belt's offset, reads the current robot 2 location and extracts the component along the Y axis. This invariant offset component, read when the object was successfully located and the grasping authorized as collision-free, will be further used in STAGE 2 to estimate the non cumulative updates of the y_off motion, to alter the SCARA's target location along the Y axis.
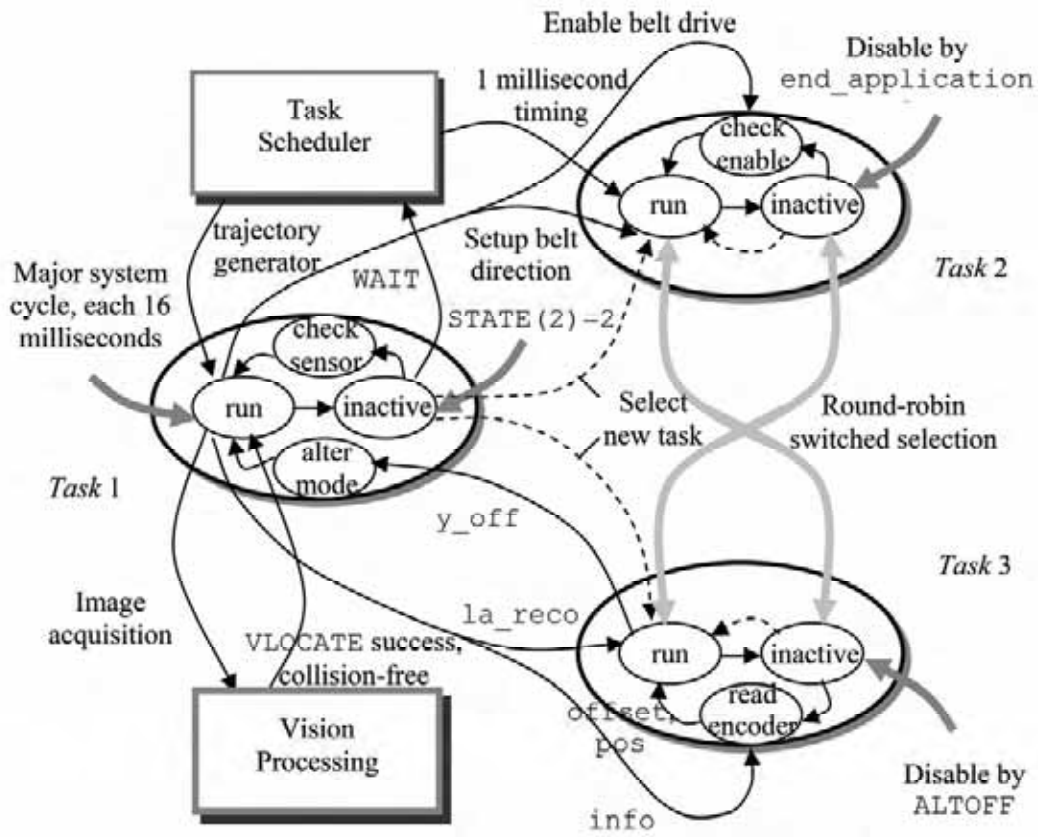 The cooperation between the tasks on which run the "track", "drive" and "read" programs is shown in Fig. 8.

Figure 8. Cooperation between the tasks of the belt tracking control problem

4. Authorizing collision-free grasping by fingerprint models

Random scene foregrounds, as the conveyor belt, may need to be faced in robotic tasks. Depending on the parts shape and on their dimension along z^{+}, grasping models Gs_m are off line trained for prototypes representing object classes. However, if there is the slightest uncertainty about the risk of collision between the gripper and parts on the belt – touching or close one relative to the other –, then *extended* grasping models $EG\_m = \{Gs\_m, FGP\_m\}$ must be created by the adding the gripper's *fingerprint model* FGP_m to effectively authorize part access only after *clear grip tests* at run time.

Definition.

A *multiple fingerprint model* $MFGP\_m(G, O) = \{FGP\_m_1(G, O), ..., FGP\_m_k(G, O)\}$ for a p-fingered gripper G and a class of objects O describes the shape, location and interpretation of k sets of p projections of the gripper's fingerprints onto the image plane x_{vis}, y_{vis} for the corresponding k grasping styles $Gs\_m_i, i = 1, ..., k$ of O-class instances. A $FGP\_m_i(G, O)$ model has the following parameter structure:

- *finger _ shape*$(G) = number, shape_i, size_i, i = 1,..., p$, expresses the shape of the gripper in terms of its number p of fingers, the shape and dimensions of each finger. *Rectangular*-shaped fingers are considered; their size is given "width" and "height".

- *fingers _ location*$(G,O) = \{x_{ci}(O), y_{ci}(O), rz_i(O)\}, i = 1,..., p$, indicates the relative location of each finger with respect to the object's *centre of mass* and *minimum inertia axis* (MIA). At training time, this description is created for the object's model, and its updating will be performed at run time by the vision system for any recognized instance of the prototype.

- *fingers _ viewing*$(G, pose \_ context_i, i = 1,...p)$ indicating the way how "invisible" fingers are to be treated; fingers are "invisible" if they are outside the field of view.

- $grip = 1,..., k$ are the k gripper-object $Gs\_m(G,O)$ distinct grasping models a priori trained, as possible alternatives to face at run time foreground context situations.

A *collision-free grasping transformation* $CF(Gs\_m_i, O)$ will be selected at run time from one of the k *grip* parameters, after checking that all pixels belonging to FGP_m_i (the projection of the gripper's fingerprints onto the image plane x_{vis}, y_{vis}, in the O-grasping location) cover only background-coloured pixels. To provide a secure, collision-free access to objects, the following <u>robot-vision sequence</u> must be executed:

1. *Training* k sets of parameters of the multiple fingerprints model $MFGP\_m(G,O)$ for G and object class O, relative to the k learned grasping styles $Gs\_m_i(G,O), i = 1,..., k$.

2. *Installing* the multiple fingerprint model $MFGP\_m(G,O)$ defining the shape, position and interpretation (viewing) of the robot gripper for clear-grip tests, by including the model parameters in a data base available at run time. This must be done at the start of application programs prior to any image acquisition and object locating.

3. *Automatically performing the clear-grip test* whenever a prototype is recognized and located at run time, and grips $FGP\_m_i, i = 1,....k$ have been a priori defined for it.

4. On line *call of the grasping parameters* trained in the $Gs\_m_i(G,O)$ model, which corresponds to the first grip FGP_m_i found to be clear.

The first step in this robot-vision sequence prepares off line the data allowing to position at run time two Windows Region of Interest (WROI) around the current object, invariant to its visually computed location, corresponding to the

two gripper fingerprints. This data refers to the size, position and orientation of the gripper's fingerprints, and is based on:

- the *number and dimensions of the gripper's fingers*: 2-parallel fingered grippers were considered, each one having a rectangular shape of dimensions wd_g, ht_g;

- the *grasping location of the fingers relative to the class model* of objects of interest.

This last information is obtained by *learning* any grasping transformation for a class of objects (e.g. "LA"), and is described by help of Fig. 9. The following frames and relative transformations are considered:

- <u>Frames</u>: (x_0, y_0): in the robot's base (world); (x_{vis}, y_{vis}): attached to the image plane; (x_g, y_g): attached to the gripper in its end-point T; (x_{loc}, y_{loc}): default object-attached frame, $x_{loc} \equiv$ MIA (the part's minimum inertia axis); (x_{obj}, y_{obj}): rotated object-attached frame, with $x_{obj} \equiv$ dir(C, G), $C(x_c, y_c)$ being the object's centre of mass and $G = \text{proj}_{(x_{vis}, y_{vis})} T$;

- <u>Relative transformations</u>: to.cam[cam]: describes, for the given camera, the location of the vision frame with respect to the robot's base frame; vis.loc: describes the location of the object-attached frame with respect to the vision frame; vis.obj: describes the location of the object-attached frame with respect to the vision frame; pt.rob: describes the location of the gripper frame with respect to the robot frame; pt.vis: describes the location of the gripper frame with respect to the vision frame.

As a result of this learning stage, which uses vision and the robot's joint encoders as measuring devices, a grasping model $\mathcal{GP}\_m(G, \text{"LA"}) = \{d.cg, alpha, z\_off, rz\_off\}$ is derived, relative to the object's centre of mass C and minimum inertia axis MIA (C and MIA are also available at runtime):

$$d.cg = \text{dist}(C, G), \ alpha = \angle(\text{MIA}, \text{dir}(C, G)), z\_off = \text{dist}(T, G), rz\_off = \angle(x_g, \text{dir}(C, G))$$

A clear grip test is executed at run time to check the collision-free grasping of a recognized and located object, by projecting the gripper's fingerprints onto the image plane, (x_{vis}, y_{vis}), and verifying whether they "cover" only *background pixels*, which means that no other object exists close to the area where the gripper's fingers will be positioned by the current robot motion command. A negative result of this test will not authorize the grasping of the object.

For the test purpose, two WROIs are placed in the image plane, exactly over the areas occupied by the projections of the gripper's fingerprints in the image plane for the desired, object-relative grasping location computed from $\mathcal{GP}\_m(G, \text{"LA"})$; the position (C) and orientation (MIA) of the recognized object must be available. From the invariant, part-related data:

$alpha, rz.off, wd_{LA}, wd_g, ht_g, d.cg$, there will be first computed at run time the current coordinates x_G, y_G of the point G, and the current orientation angle *angle.grasp* of the gripper slide axis relative to the vision frame.
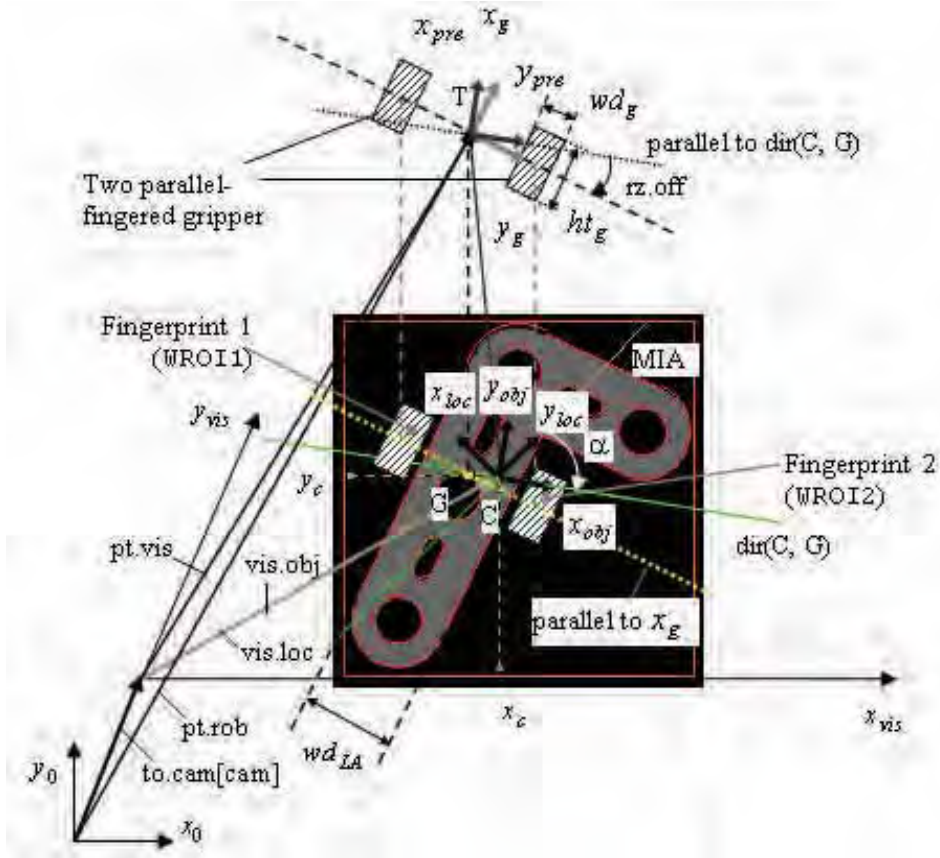
Figure 9. Frames and relative transformations used to teach the $GP\_m(G, \texttt{"LA"})$ parameters

The part's orientation *angle.aim* $= \angle(\text{MIA}, x_{vis})$ returned by vision is added to the learned *alpha*.

$$beta = \angle(\text{dir}(C,G), x_{vis}) = angle.aim + alpha \qquad (5)$$

Once the part located, the coordinates x_C, y_C of its gravity centre C are available from vision. Using them and *beta*, the coordinates x_G, y_G of the G are computed as follows:

$$x_G = x_C - d.cg \cdot \cos(beta), \quad y_G = y_C - d.cg \cdot \sin(beta) \qquad (6)$$

Now, the value of $angle.grasp = \angle(x_g, x_{vis})$, for the object's current orientation and accounting for $rz.off$ from the desired, learned grasping model, is obtained from $angle.grasp = beta + rz.off$.

Two image areas, corresponding to the projections of the two fingerprints on the image plane, are next specified using two WROI operations. Using the geometry data from Fig. 9, and denoting by dg the offset between the end-tip point projection G, and the fingerprints centres CWi, $\forall i = 1,2$, $dg = wd_{LA}/2 + wd_g/2$, the <u>positions</u> of the rectangle image areas "covered" by the fingerprints projected on the image plane in the desired part-relative grasping location are computed at run time according to (7). Their common orientation in the image plane is given by $angle.grasp$.

$$x_{cw1} = x_G - dg \cdot \cos(angle.grasp); x_{cw2} = x_G + dg \cdot \cos(angle.grasp)$$

$$(7)$$

$$y_{cw1} = y_G - dg \cdot \sin(angle.grasp); y_{cw2} = y_G + dg \cdot \sin(angle.grasp)$$

The type of image statistics is returned as the total number of non-zero (background) pixels found in each one of the two windows, superposed onto the areas covered by the fingerprints projections in the image plane, around the object. The clear grip test checks these values returned by the two WROI-generating operations, corresponding to the number of background pixels not occupied by other objects close to the current one (counted exactly in the gripper's fingerprint projection areas), against the total number of pixels corresponding to the surfaces of the rectangle fingerprints. If the difference between the compared values is less than an imposed error err for both fingerprints – windows, the grasping is authorized:

If $\left| ar1[4] - ar.fngprt \right| \le err$ AND $\left| ar2[4] - ar.fngprt \right| \le err$,

  clear grip of object is authorized; proceed object tracking by continuously
  altering its target location on the vision belt, until robot motion is completed.
Else
  another objects is too close to the current one, grasping is not authorized.

Here, $ar.fngprt = wd_g \, ht_g / [(\text{pix.to.mm})^2 \, \text{XY\_scale}]$ is the fingerprint's area [raw pixels], using the camera-robot calibration data: pix.to.mm (no. of image pixels/1 mm), and XY_scale (x/y ratio of each pixel).

5. Conclusion

The robot motion control algorithms with guidance vision for tracking and grasping objects moving on conveyor belts, modelled with belt variables and 1-d.o.f. robotic device, have been tested on a robot-vision system composed from a Cobra 600TT manipulator, a C40 robot controller equipped with EVI vision processor from Adept Technology, a parallel two-fingered RIP6.2 gripper from CCMOP, a "large-format" stationary camera (1024x1024 pixels) down looking at the conveyor belt, and a GEL-209 magnetic encoder with 1024 pulses per revolution from Leonard Bauer. The encoder's output is fed to one of the EJI cards of the robot controller, the belt conveyor being "seen" as an external device.

Image acquisition used strobe light in *synchronous* mode to avoid the acquisition of blurred images for objects moving on the conveyor belt. The strobe light is triggered each time an image acquisition and processing operation is executed at runtime. Image acquisitions are synchronised with external events of the type: "*a part has completely entered the belt window*"; because these events generate on-off photocell signals, they trigger the *fast digital-interrupt line* of the robot controller to which the photocell is physically connected. Hence, the VPICTURE operations always wait on interrupt signals, which significantly improve the response time at external events. Because a fast line was used, the most unfavourable delay between the triggering of this line and the request for image acquisition is of only 0.2 milliseconds.

The effects of this *most unfavourable 0.2 milliseconds time delay* upon the integrity of object images have been analysed and tested for two modes of strobe light triggering:

- *Asynchronous triggering* with respect to the read cycle of the video camera, i.e. as soon as an image acquisition request appears. For a 51.2 cm width of the image field, and a line resolution of 512 pixels, the pixel width is of 1 mm. For a 2.5 m/sec high-speed motion of objects on the conveyor belt the most unfavourable delay of 0.2 milliseconds corresponds to a displacement of only one pixel (and hence one object-pixel might disappear during the *dist* travel above defined), as:

$$(0.0002 \text{ sec}) * (2500 \text{ mm/sec}) / (1 \text{ mm/pixel}) = 0.5 \text{ pixels.}$$

- *Synchronous triggering* with respect to the read cycle of the camera, inducing a variable time delay between the image acquisition request and the strobe light triggering. The most unfavourable delay was in this case 16.7 milliseconds, which may cause, for the same image field and belt speed a potential disappearance of 41.75 pixels from the camera's field of view (downstream the *dwnstr_lim* limit of the belt window).

Consequently, the bigger are the dimensions of the parts travelling on the conveyor belt, the higher is the risk of disappearance of pixels situated in downstream areas. Fig. 10 shows a statistics about the sum of:

- *visual locating errors*: errors in object locating relative to the image frame (x_{vis}, y_{vis}); consequently, the request for motion planning will then not be issued;
- *motion planning errors*: errors in the robot's destinations evaluated during motion planning as being downstream *downstr_lim*, and hence not authorised,

function of the *object's dimension* (length *long_max.obj* along the minimal inertia axis) and of the *belt speed* (four high speed values have been considered: 0.5 m/sec, 1 m/sec, 2 m/sec and 3 m/sec).

As can be observed, at the very high motion speed of 3 m/sec, for parts longer than 35 cm there was registered a percentage of more than 16% of unsuccessful object locating and of more than 7% of missed planning of robot destinations (which are outside the CBW) for visually located parts, from a total number of 250 experiments.

The clear grip check method presented above was implemented in the V+ programming environment with AVI vision extension, and tested on the same robot vision platform containing an Adept Cobra 600TT SCARA-type manipulator, a 3-belt flexible feeding system Adept FlexFeeder 250 and a stationary, down looking matrix camera Panasonic GP MF650 inspecting the vision belt. The vision belt on which parts were travelling and presented to the camera was positioned for a convenient robot access within a window of 460 mm.

Experiments for collision-free part access on randomly populated conveyor belt have been carried out at several speed values of the transportation belt, in the range from 5 to 180 mm/sec. Table 1 shows the correspondence between the belt speeds and the maximum time intervals from the visual detection of a part and its collision-free grasping upon checking [#] sets of pre taught grasping models $\mathcal{G}s\_m_i(\mathcal{G}, O), i = 1, ..., \#$.
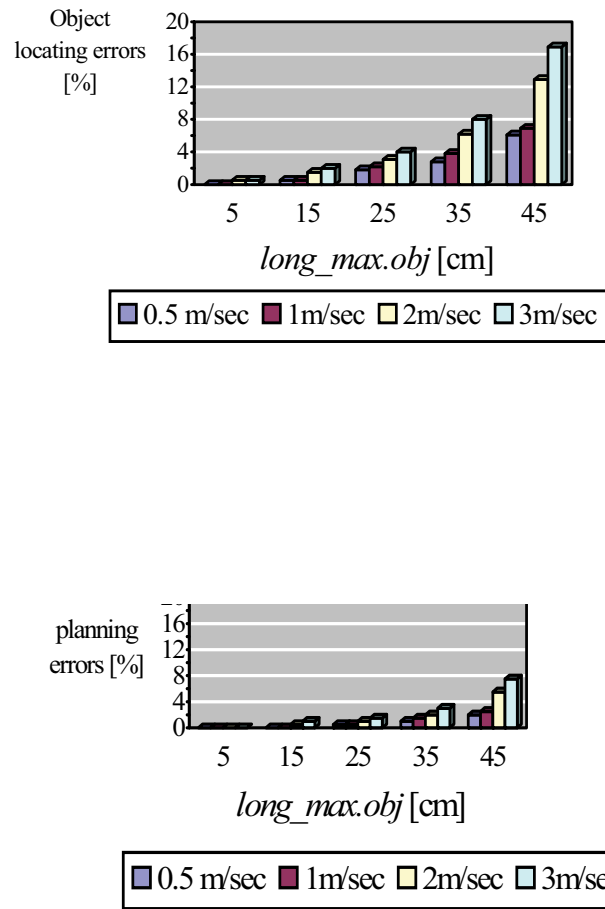
Figure 10. Error statistics for visual object locating and robot motion planning

Belt speed [mm/sec]	5	10	30	50	100	180
Grasping time (max) [sec]	1.4	1.6	1.9	2.0	2.3	2.5
Clear grips checked[#]	4	4	4	4	2	1

Table 1. Corespondance between belt speed and collision-free part grasping time

6. References

Adept Technology Inc. (2001). *AdeptVision User's Guide Version 14.0*, Technical Publications, Part Number 00964-03300, Rev. B, San Jose, CA

Borangiu, Th. & Dupas, M. (2001). *Robot – Vision. Mise en œuvre en V+*, Romanian Academy Press & AGIR Press, Bucharest

Borangiu, Th. (2002). Visual Conveyor Tracking for "Pick-On-The-Fly" Robot Motion Control, *Proc. of the IEEE Conf. Advanced Motion Control AMC'02*, pp. 317-322, Maribor

Borangiu, Th. (2004). *Intelligent Image Processing in Robotics and Manufacturing*, Romanian Academy Press, ISBN 973-27-1103-5, Bucarest

Borangiu, Th. & Kopacek, P. (2004). *Proceedings Volume from the IFAC Workshop Intelligent Assembly and Disassembly - IAD'03 Bucharest, October 9-11, 2003,* Elsevier Science, Pergamon Press, Oxford, UK

Borangiu, Th. (2005). Guidance Vision for Robots and Part Inspection, *Proceedings volume of the 14th Int. Conf. Robotics in Alpe-Adria-Danube Region RAAD'05*, pp. 27-54, ISBN 973-718-241-3, May 2005, Bucharest

Borangiu, Th.; Manu, M.; Anton, F.-D.; Tunaru, S. & Dogar, A. (2006). High-speed Robot Motion Control under Visual Guidance, *12th International Power Electronics and Motion Control Conference - EPE-PEMC 2006*, August 2006, Portoroz, SLO.

Espiau, B.; Chaumette, F. & Rives, P. (1992). A new approach to visual servoing in robotics, *IEEE Trans. Robot. Automat., vol. 8*, pp. 313-326

Lindenbaum, M. (1997). An Integrated Model for Evaluating the Amount of Data Required for Reliable Recognition, *IEEE Trans. on Pattern Analysis & Machine Intell.*

Hutchinson, S. A.; Hager, G.D. & Corke, P. (1996). A Tutorial on Visual Servo Control, *IEEE Trans. on Robotics and Automation*, vol. 12, pp. 1245-1266, October 1996

Schilling, R.J. (1990). *Fundamentals of Robotics. Analysis and Control*, Prentice-Hall, Englewood Cliffs, N.J.

Zhuang, X.; Wang, T. & Zhang, P. (1992). A Highly Robust Estimator through Partially Likelihood Function Modelling and Its Application in Computer Vision, *IEEE Trans. on Pattern Analysis and Machine Intelligence*

West, P. (2001). High Speed, Real-Time Machine Vision, *CyberOptics – Imagenation*, pp. 1-38 Portland, Oregon

Visual Feedback Control of a Robot in an Unknown Environment (Learning Control Using Neural Networks)

Xiao Nan-Feng and Saeid Nahavandi

1. Introduction

When a robot has no transcendental knowledge about an object to be traced and an operation environment, a vision sensor is needed to attach to the robot in order to recognize the object and the environment. On the other hand, it is also desirable that the robot has learning ability in order to improve effectively the trace operation in the unknown environment.

Many methods[1]-[11] have been so far proposed to control a robot with a camera to trace an object so as to complete a non-contact operation in an unknown environment. e.g., in order to automate a sealing operation by a robot, Hosoda, K.[1] proposed a method to perform the sealing operation by the robot through off-line teaching beforehand. This method used a CCD camera and slit lasers to detect the sealing line taught beforehand and to correct on line the joint angles of the robot during the sealing operation.

However, in those methods[1]-[3], only one or two image feature points of the sealing were searched per image processing period and the goal trajectory of the robot was generated using an interpolation. Moreover, those methods must perform the tedious CCD camera calibration and the complicated coordinate transformations. Furthermore, the synchronization problem between the image processing system and the robot control system, and the influences of the disturbances caused by the joint friction and the gravity of the robot need to be solved.

In this chapter, a visual feedback control method is presented for a robot to trace a curved line in an unknown environment. Firstly, the necessary conditions are derived for one-to-one mapping from the image feature domain of the curved line to the joint angle domain of the robot, and a multilayer neural network which will be abbreviated to NN hereafter is introduced to learn the mapping. Secondly, a method is proposed to generate on line the goal trajectory through computing the image feature parameters of the curved line. Thirdly, a multilayer neural network-based on-line learning algorithm is developed for the present visual feedback control. Lastly, the present approach is applied to trace a curved line using a 6 DOF industrial robot with a CCD cam-

era installed in its end-effecter. The main advantage of the present approach is that it does not necessitate the tedious CCD camera calibration and the complicated coordinate transformations.
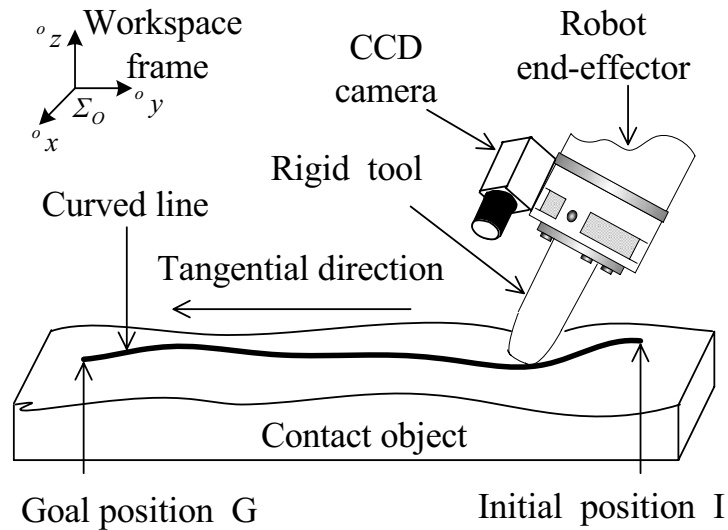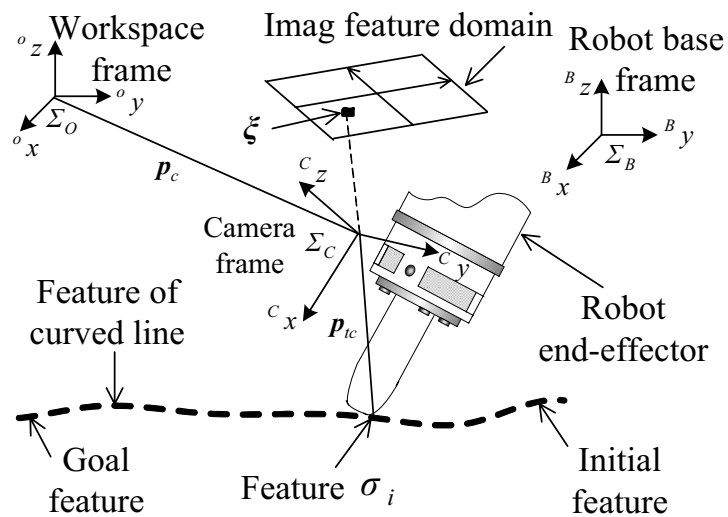

Figure 1. Vision-based trace operation

Figure 2. Image features and mapping relation

2. A Trace Operation by a Robot

When a robot traces an object in an unknown environment, visual feedback is necessary for controlling the position and orientation of the robot in the tangential and normal directions of the operation environment. Figure 1 shows a robot with a CCD camera installed in its end-effecter. The CCD camera guides the end-effecter to trace a curved line from the initial position I to the target position G. Since the CCD camera is being fixed at the end-effecter, the CCD camera and the end-effecter always move together.

3. Mapping from Image Feature Domain to Joint Angle Domain

3.1 Visual feedback control

For the visual feedback control shown in Fig. 1, the trace error of the robot in the image feature domain needs to be mapped to the joint angle domain of the robot. That is, the end-effecter should trace the curved line according to a_j, a_{j+1} in the image domain of the features A_j, A_{j+1} shown in Fig. 2.

Let $\xi, \xi_d \in R^{6\times1}$ be the image feature parameter vectors of a_j, a_{j+1} in the image feature domain shown in Fig. 2, respectively. The visual feedback control shown in Fig. 1 can be expressed as

$$e = ||\xi_d - \xi||, \tag{1}$$

where $||\cdot||$ is a norm, and e should be made into a minimum.

From the projection relation shown in Fig. 2, we know

$$\xi = \varphi(p_{tc}), \tag{2}$$

where $\varphi \in R^{6\times1}$ is a nonlinear mapping function which realizes the projection transformation from the workspace coordinate frame $\sum_O$ to the image feature domain shown in Fig. 2.
It is assumed that $p_{tc} \in R^{6\times1}$ is a position/orientation vector from the origin of the CCD camera coordinate frame $\sum_C$ to the gravity center of A_j. Linearizing Eq.(2) at a minute domain of p_{tc} yields

$$\delta\xi = J_f \cdot \delta p_{tc}, \tag{3}$$

where $\delta\xi$ and δp_{tc} are minute increments of ξ and p_{tc}, respectively, and $J_f = \partial\varphi/\partial p_{tc} \in R^{6\times6}$ is a feature sensitivity matrix.

Furthermore, let θ and $\delta\theta \in R^{6\times1}$ are a joint angle vector of the robot and its minute increment in the robot base coordinate frame $\sum_B$. If we map $\delta\theta$ from $\sum_B$ to δp_{tc} in $\sum_O$ using Jacobian matrix of the CCD camera $J_c \in R^{6\times6}$, we can get

$$\delta p_{tc} = J_c \cdot \delta\theta.$$ (4)

From Eqs.(3) and (4), we have

$$\delta\theta = (J_f J_c)^{-1} \cdot \delta\xi.$$ (5)

Therefore, the necessary condition for realizing the mapping expressed by Eq.(5) is that $(J_f J_c)^{-1}$ must exist. Moreover, the number of the independent image feature parameters in the image feature domain (or the element numbers of ξ) must be equal to the degrees of freedom of the visual feedback control system.

3.2 Mapping relations between image features and joint angles

Because J_f and J_c are respectively linearized in the minute domains of p_{tc} and θ, the motion of the robot is restricted to a minute joint angle domain, and Eq.(5) is not correct for large $\delta\theta$. Simultaneously, the mapping is weak to the change of $(J_f J_c)^{-1}$. In addition, it is very difficult to calculate $(J_f J_c)^{-1}$ on line during the trace operation. Therefore, NN is introduced to learn such mapping.

Firstly, we consider the mapping from Δp_{tc} in $\sum_O$ to $\Delta\xi$ in the image feature domain. $\Delta\xi$ and Δp_{tc} are increments of ξ and p_{tc} for the end-effecter to move from A_j to A_{j+1}, respectively. As shown in Fig. 2, the mapping is depend on p_{tc}, and the mapping can be expressed as

$$\Delta\xi = \varphi_1\left(\Delta p_{tc}, p_{tc}\right),$$ (6)

where $\varphi_1(\ ) \in R^{6\times1}$ is a continuous nonlinear mapping function. We have from Eq.(6),

$$\Delta p_{tc} = \varphi_2\left(\Delta\xi, p_{tc}\right),$$ (7)

where $\varphi_2(\ ) \in R^{6\times1}$ is a continuous nonlinear mapping function. When ξ is uniquely specified in the image feature domain, there is a one-to-one mapping relationship between p_{tc} and ξ. Therefore, Eq.(7) is expressed as

$$\Delta \boldsymbol{p}_{tc} = \varphi_3 \left(\Delta \xi , \xi \right), \tag{8}$$

where $\varphi_3 ( ) \in R^{6 \times 1}$ is a continuous nonlinear mapping function.

Secondly, we consider the mapping from $\Delta \theta$ in $\sum_B$ to $\Delta \boldsymbol{p}_c$ in $\sum_O$. Let $\boldsymbol{p}_c \in R^{6 \times 1}$ be a position/ orientation vector of the origin of $\sum_C$ with respect to the origin of $\sum_O$, $\Delta \theta$ and $\Delta \boldsymbol{p}_c$ be increments of θ and $\boldsymbol{p}_c$ for the end-effecter to move from A_j to A_{j+1}, respectively. $\Delta \boldsymbol{p}_c$ is dependent on θ as shown in Fig. 2, and we obtain from the forward kinematics of the CCD camera

$$\Delta \boldsymbol{p}_c = \varphi_3^{'} \left(\Delta \theta , \theta \right), \tag{9}$$

where $\varphi_3^{'} ( ) \in R^{6 \times 1}$ is a continuous nonlinear mapping function.

Since the relative position and orientation between the end-effecter and the CCD camera are fixed, the mapping from $\Delta \boldsymbol{p}_c$ to $\Delta \boldsymbol{p}_{tc}$ is also one-to-one. We get

$$\Delta \boldsymbol{p}_{tc} = \varphi_4 \left(\Delta \boldsymbol{p}_c \right), \tag{10}$$

where $\varphi_4 ( ) \in R^{6 \times 1}$ is a continuous nonlinear mapping function. Combining Eq.(9) and Eq.(10) gives

$$\Delta \boldsymbol{p}_{tc} = \varphi_4^{'} \left(\Delta \theta , \theta \right), \tag{11.a}$$

and we have from Eq.(11.a)

$$\Delta \theta = \varphi_5 \left(\Delta \boldsymbol{p}_{tc} , \theta \right), \tag{11.b}$$

where $\varphi_5 ( ) \in R^{6 \times 1}$ is a continuous nonlinear mapping function. It is known from Eq.(11.b) that if the CCD camera moves from A_j to A_{j+1}, the robot has an unique $\Delta \theta$.

Combing Eq.(8) and Eq.(11.b) yields

$$\Delta \theta = \varphi_6 \left(\Delta \xi , \xi , \theta \right), \tag{12}$$

where $\varphi_6 ( ) \in R^{6 \times 1}$ is a continuous nonlinear mapping function. In this paper, NN is used to learn $\varphi_6 ( )$.

3.2 Computation of image feature parameters

For 6 DOF visual feedback control, 6 independent image feature parameters are chosen to correspond to the 6 joint angles of the robot. An image feature parameter vector $\xi^{(j)} = [\xi_1^{(j)}, \xi_2^{(j)}, \cdots, \xi_6^{(j)}]^T$ is defined at the window j shown in Fig. 3. L and W are length and height of the window j, respectively.

Defining $g_{qr}^{(j)}$ at the window j by

$$g_{qr}^{(j)} = \begin{cases} 0 & (\text{white pixel}) \\ 1 & (\text{black pixel}) \end{cases} , \tag{13.a}$$

the elements of $\xi^{(j)}$ are defined and calculated by the following equations:

(1) The gravity center coordinates

$$\xi_1^{(j)} = \frac{\sum\limits_{q=1}^{L}\sum\limits_{r=1}^{W} g_{qr}^{(j)} \cdot q}{\sum\limits_{q=1}^{L}\sum\limits_{r=1}^{W} g_{qr}^{(j)}} , \tag{13.b}$$
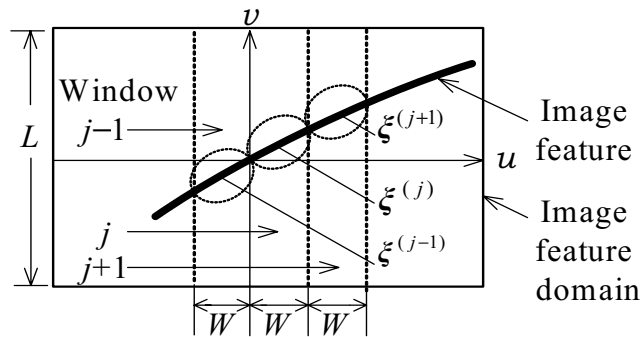

Figure 3.Definition of image feature parameter vector

$$\xi_2^{(j)} = \frac{\sum\limits_{q=1}^{L}\sum\limits_{r=1}^{W} g_{qr}^{(j)} \cdot r}{\sum\limits_{q=1}^{L}\sum\limits_{r=1}^{W} g_{qr}^{(j)}} , \tag{13.c}$$

(2) The area

$$\xi_3^{(j)} = \sum\limits_{q=1}^{L}\sum\limits_{r=1}^{W} g_{qr}^{(j)} , \tag{13.d}$$

(3) The lengths of the main and minor axes of the equivalent ellipse

$$\xi_4^{(j)} = \frac{\lambda_{20}^{(j)} + \lambda_{02}^{(j)} + \sqrt{(\lambda_{20}^{(j)} - \lambda_{02}^{(j)})^2 + 4(\lambda_{11}^{(j)})^2}}{2\xi_3^{(j)}},\tag{13.e}$$

$$\xi_5^{(j)} = \frac{\lambda_{20}^{(j)} + \lambda_{02}^{(j)} - \sqrt{(\lambda_{20}^{(j)} - \lambda_{02}^{(j)})^2 + 4(\lambda_{11}^{(j)})^2}}{2\xi_3^{(j)}},\tag{13.f}$$

Where

$$\lambda_{11}^{(j)} = \sum_{q=1}^{L}\sum_{r=1}^{W} g_{qr}^{(j)} \cdot [q - \xi_1^{(j)}][r - \xi_2^{(j)}],\tag{13.g}$$

$$\lambda_{20}^{(j)} = \sum_{q=1}^{L}\sum_{r=1}^{W} g_{qr}^{(j)} \cdot [q - \xi_1^{(j)}]2,\tag{13.h}$$

$$\lambda_{02}^{(j)} = \sum_{q=1}^{L}\sum_{r=1}^{W} g_{qr}^{(j)} \cdot [r - \xi_2^{(j)}]2,\tag{13.i}$$

(4) The orientation

$$\xi_6^{(j)} = \frac{1}{2}\tan^{-1}\frac{\lambda_{11}^{(j)}}{\lambda_{20}^{(j)} - \lambda_{02}^{(j)}}.\tag{13.j}$$

At time $t=imT$, ξ and $\varDelta\xi$ in Eq.(12) are given by

$$\xi = \xi^{(j)}(im),\tag{14.a}$$

$$\varDelta\xi = \varDelta\xi(im),\tag{14.b}$$

where $\xi^{(j)}(im)$ and $\xi^{(j+1)}(im)$ are image feature parameter vectors in the window j and $j+1$ shown in Fig. 3. $\xi^{(0)}(im)$, $\xi^{(1)}(im)$ and $\xi^{(2)}(im)$ can be calculated for $j=0,1,2$ in Eq.(13.a)~(13.j).

4. Goal Trajectory Generation Using Image Feature Parameters

In this paper, a CCD camera is used to detect the image feature parameters of the curved line, which are used to generate on line the goal trajectory. The se-

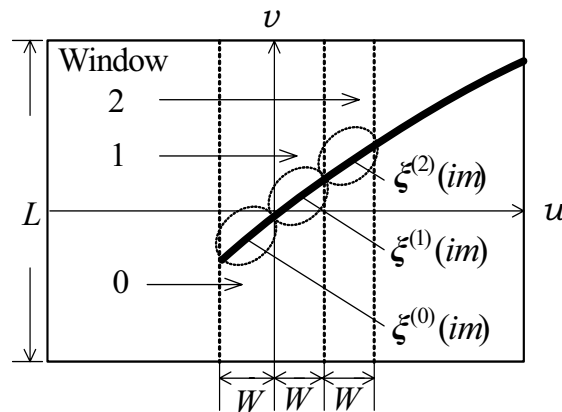quences of trajectory generation are shown in Fig. 4. Firstly, the end-effecter is set to the central point of the window 0 in Fig. 4(a). At time $t=0$, the first image of the curved line is grasped and processed, and the image feature parameter vectors $\xi^{(0)}(0), \xi^{(1)}(0)$ and $\xi^{(2)}(0)$ in the windows 0,1,2 are computed respectively. From time $t=mT$ to $t=2mT$, the end-effecter is only moved by $\varDelta\xi(0) = \xi^{(1)}(0) - \xi^{(0)}(0)$. At time $t=mT$, the second image of the curved line is grasped and processed, the image feature parameter vector $\xi^{(2)}(m)$ shown in Fig. 4(b) is computed. From $t=2mT$ to $t=3mT$, the end-effecter is only moved by $\varDelta\xi(m) = \xi^{(2)}(0) - \xi^{(1)}(0)$. At time $t=imT$, $(i+1)$th image is grasped and processed, the image feature parameter vector $\xi^{(2)}(im)$ shown in Fig. 4(c) is computed. From $t=imT$ to $t=(i+1)mT$, the end-effecter is only moved by $\varDelta\xi(im) = \xi^{(1)}[(i-1)m] - \xi^{(0)}[(i-1)m]$.
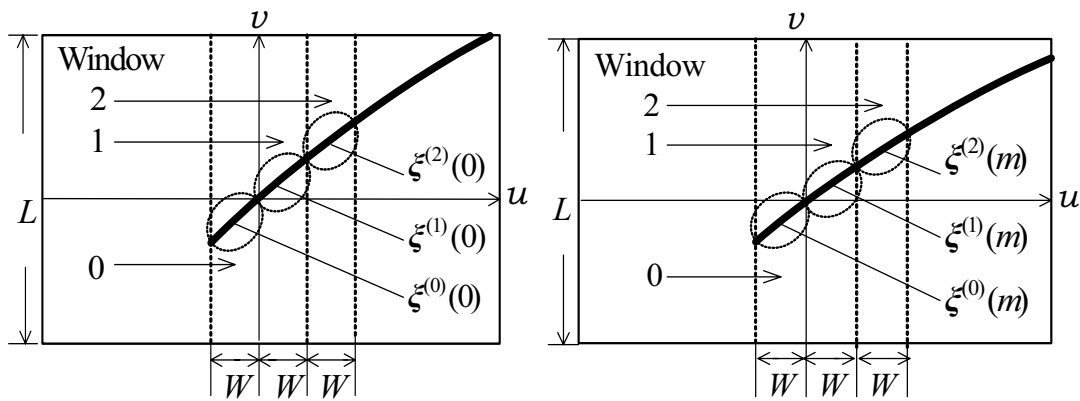
(a) Image feature domain at $t=0$ b) Image feature domain at $t=mT$

c) Image feature domain $t=imT$

Figure 4.Image trajectory generation sequences

5. Neural Network-Based Learning Control System

5.1 Off-line learning algorithm

In this section, a multilayer NN is introduced to learn $\varphi_6( )$. Figure 5 shows the structure of NN which includes the input level A, the hidden level B, C and the output level D. Let M, P, U, N be the neuron number of the levels A, B, C, D, respectively, $g=1,2,\cdots,M$, $l=1,2,\cdots,P$, $j=1,2,\cdots,U$ and $i=1,2,\cdots,N$. Therefore, NN can be expressed as
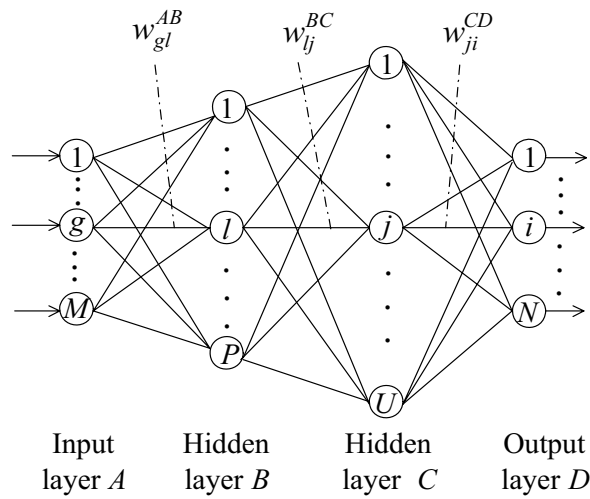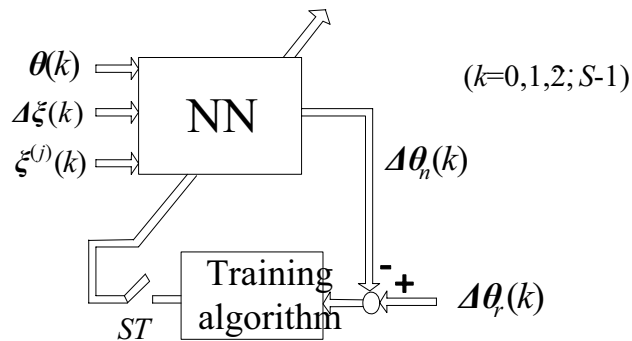

Figure 5. Neural network (NN) structure

Figure 6. Off-line learning of NN

$$
\left.
\begin{array}{llll}
x_l^B = \sum\limits_{g=1}^{M} w_{gl}^{AB} \cdot y_g^A + z_l^B, & x_j^C = \sum\limits_{l=1}^{P} w_{lj}^{BC} \cdot y_l^B + z_j^C, & x_i^D = \sum\limits_{j=1}^{U} w_{ji}^{CD} \cdot y_j^C + z_i^D \\
y_g^A = x_g^A, & y_l^B = f(x_l^B), & y_j^C = f(x_j^C), & y_i^D = f(x_i^D)
\end{array}
\right\},
\qquad (15)
$$

where x_m^R, y_m^R and z_m^R (m=g, l, j, i; R=A, B, C, D) are the input, the output and the bias of a neuron, respectively. w_{gl}^{AB}, w_{lj}^{BC} and w_{lj}^{CD} are the weights between y_g^A and x_l^B, y_l^B and x_j^C, y_j^C and x_i^D, respectively. The sigmoid function of NN is defined by

$$
f(x) = \frac{1 - e^{-\beta x}}{1 + e^{-\beta x}}, \qquad (16)
$$

where β is a real number which specifies the characteristics of $f(x)$. The learning method of NN is shown in Fig. 6, and its evaluation function is defined as

$$
E_f = \frac{1}{2S} \sum_{k=0}^{S-1} [\boldsymbol{\Delta\theta}_r(k) - \boldsymbol{\Delta\theta}_n(k)]^T [\boldsymbol{\Delta\theta}_r(k) - \boldsymbol{\Delta\theta}_n(k)], \qquad (17)
$$

where $\boldsymbol{\Delta\theta}_r(k)$ and $\boldsymbol{\Delta\theta}_n(k) \in R^{6\times1}$ are the increments of the robot joint angle vector and the output vector of the neural network, respectively. The positive integer S is the number of the learning samples $\theta(k)$, $\boldsymbol{\Delta\theta}_r(k)$, $\xi^{(j)}(k)$, $\Delta\xi(k)$ for the end-effecter to move along the curved line from the initial position I to the goal position G. $\theta(k), \boldsymbol{\Delta\theta}_r(k), \xi^{(j)}(k)$ and $\Delta\xi(k)$ are off-line measured in advance, respectively. $\theta(k)$, $\xi^{(j)}(k)$ and $\Delta\xi(k)$ are divided by their maximums before inputting to NN, respectively. w_{ji}^{CD} of NN is changed by

$$
\Delta w_{ji}^{CD} = -\eta_f \frac{\partial E_f}{\partial w_{ji}^{CD}}, \quad (j=1,2,\cdots,U; i=1,2,\cdots,N), \qquad (18.a)
$$

where Δw_{ji}^{CD} is an increment of w_{ji}^{CD}, η_f is a learning rate of NN. From Eqs.(15)~(17), we obtain

$$
\frac{\partial E_f}{\partial w_{ji}^{CD}} = -[\Delta\theta_{ri}(k) - \Delta\theta_{ni}(k)] f'(x_i^D) y_j^C, \quad (j=1,2,\cdots,U; i=1,2,\cdots,N), \qquad (18.b)
$$

where $\Delta\theta_{ri}(k)$ and $\Delta\theta_{ni}(k)$ are the *i*th element of $\boldsymbol{\Delta\theta}_r(k)$ and $\boldsymbol{\Delta\theta}_n(k)$, respectively. w_{gl}^{AB} and w_{lj}^{BC} of NN are changed by the error back-propagation algorithm. Here, the detailed process of error back propagation is omitted.

The present learning control system based on NN is shown Fig. 7. The initial w_{gl}^{AB}, w_{lj}^{BC} and w_{ji}^{CD} of NN are given by random real numbers between –0.5 and 0.5. When NN finishes learning, the reference joint angle $\theta_n(k+1)$ of the robot is obtained by

$$\left.\begin{array}{l} \boldsymbol{\theta}_n(k+1) = \boldsymbol{\theta}_n(k) + \varDelta\boldsymbol{\theta}_n(k), \\ \boldsymbol{\theta}_n(0) = \boldsymbol{\theta}(0), \quad \varDelta\boldsymbol{\theta}_n(0) = \mathbf{d}, \quad (k=0,1,\cdots,S-1) \end{array}\right\} , \qquad (19)$$

where $\mathrm{d} \in R^{6\times1}$ is an output of NN when $\theta(0), \xi^{(0)}(0)$ and $\varDelta\xi(0)$ are used as initial inputs of NN. The PID controller $G_c(z)$ is used to control the joint angles of the robot.

5.2 On-line learning algorithm

For the visual feedback control system, a multilayer neural network NN_c is introduced to compensate the nonlinear dynamics of the robot. The structure and definition of NN_c is the same as NN, and its evaluation function is defined as

$$E_c(k) = e^T(k)We(k) \qquad (20)$$

where $e(k) \in R^{6\times1}$ is an input vector of $G_c(z)$, and $W \in R^{6\times6}$ is a diagonal weighting matrix. w_{ji}^{CD} of NN_c is changed by

$$\Delta w_{ji}^{CD} = -\eta_c \frac{\partial E_c}{\partial w_{ji}^{CD}}, \qquad (j=1,2,\cdots,U; i=1,2,\cdots,N) , \qquad (21.\mathrm{a})$$

where Δw_{ji}^{CD} is an increment of w_{ji}^{CD}, and η_c is a learning rate. From Eqs.(15)~(17), we have

$$\frac{\partial E_c}{\partial w_{ji}^{CD}} = -e_i w_i f'(x_i^D) y_j^C \quad (j=1,2,\cdots,U; i=1,2,\cdots,N), \qquad (21.\mathrm{b})$$

where e_i is the ith element of $e(k)$, and w_i is the ith diagonal element of W. w_{gl}^{AB} and w_{lj}^{BC} of NN_c are changed by the error back-propagation algorithm, the process is omitted.

The initial w_{gl}^{AB}, w_{lj}^{BC} and w_{ji}^{CD} of NN_c are given by random number between –0.5 and 0.5. $\theta_n(k+1)$, $\varDelta\theta_n(k)$ and $\varDelta\theta_n(k-1)$ are divided by their maximums before inputting to NN_c, respectively. K is a scalar constant which is specified by the experiments. While NN_c is learning, the elements of $e(k)$ will become smaller and smaller.
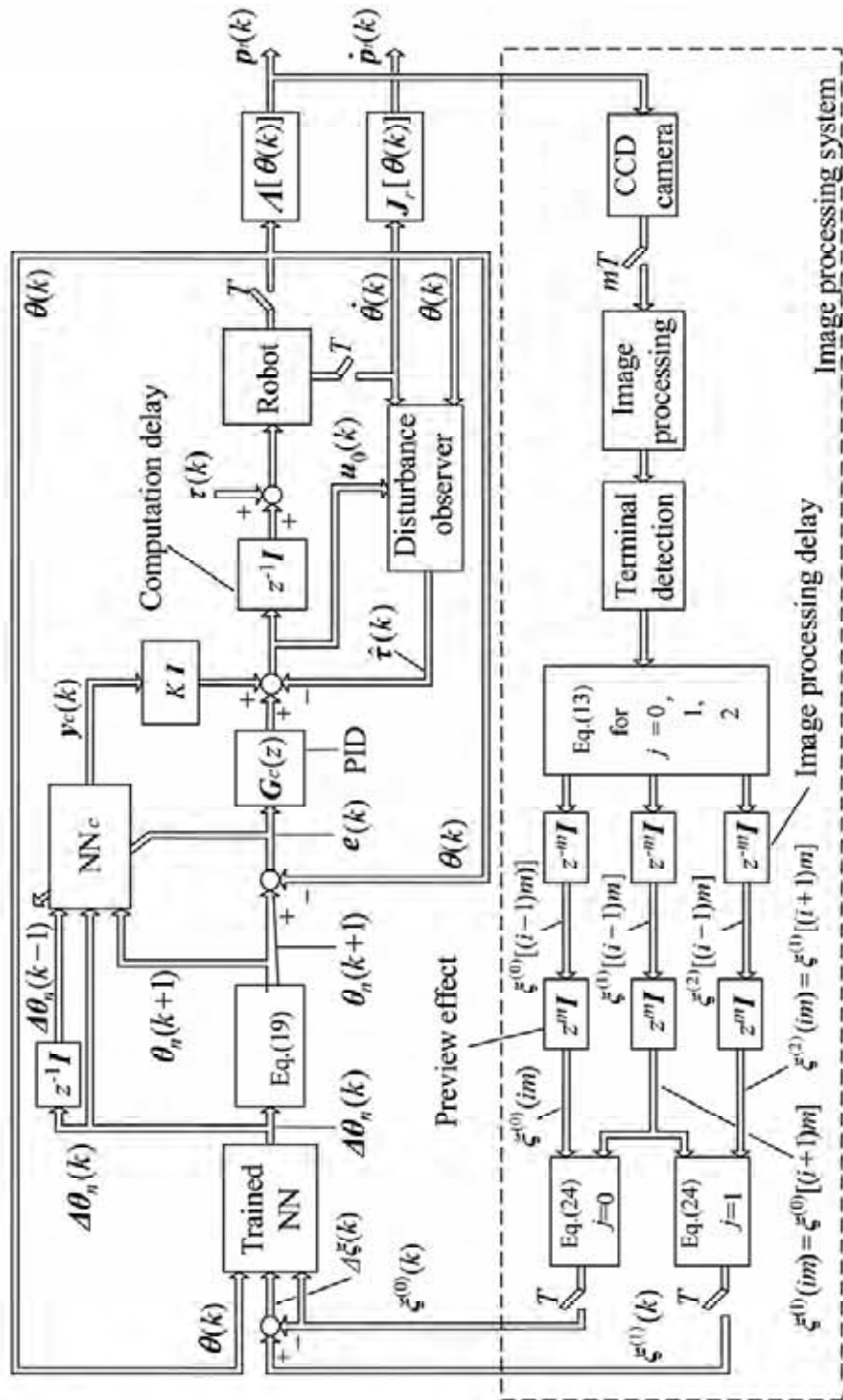
Figure 7. Block diagram of learning control system for a robot with visual feedback

In Fig. 7, I is a 6×6 unity matrix. $\varLambda[\theta(k)] \in R^{6\times1}$ and $\boldsymbol{J}_r[\theta(k)] = \partial\varLambda[\theta(k)]/\partial\theta(k) \in R^{6\times6}$ are the forward kinemics and Jacobian matrix of the end-effecter (or the rigid tool), respectively. Let $\boldsymbol{p}_t(k) = [p_{t1}, p_{t2}, \cdots, p_{t6}]^T$ be a position/orientation vector which is defined in $\sum_O$ and corresponds to the tip of the rigid tool, we have

$$p_t(k) = \varLambda[\theta(k)], \tag{22.a}$$

$$\dot{\boldsymbol{p}}_t(k) = \boldsymbol{J}_r[\theta(k)] \cdot \dot{\theta}(k). \tag{22.b}$$

The disturbance observer is used to compensate the disturbance torque vector $\tau(k) \in R^{6\times1}$ produced by the joint friction and gravity of the robot. The PID controller $\boldsymbol{G}_c(z)$ is given by

$$\boldsymbol{G}_c(z) = \boldsymbol{K}_P + \boldsymbol{K}_I \frac{z}{1-z} + \boldsymbol{K}_D (1 - z^{-1}), \tag{23}$$

where $\boldsymbol{K}_P, \boldsymbol{K}_I$ and $\boldsymbol{K}_D \in R^{6\times6}$ are diagonal matrices which are empirically determined.

5.3 Work sequence of the image processing system

The part circled by the dot line shown in Fig. 7 is an image processing system. The work sequence of the image processing system is shown in Fig. 8. At time $t=imT$, the CCD camera grasps a 256-grade gray image of the curved line, the image is binarized, and the left and right terminals of the curved line are detected. Afterward, the image parameters $\xi^{(0)}(0), \xi^{(1)}(0)$ and $\xi^{(2)}(0)$ (or $\xi^{(2)}(im)$) are computed using Eqs. (13.a)~(13.j) in the windows 0,1,2 shown in Figs. 4(a)~(c). Furthermore, in order to synchronize the image processing system and the robot joint control system, the 2nd-order holder $G_{h2}(z)$ in Section 5.4 is introduced. $\xi^{(0)}(im), \xi^{(1)}(im)$ and $\varDelta\xi(im)$ are processed by $G_{h2}(z)$, and their discrete time values $\xi^{(0)}(k), \xi^{(1)}(k)$ and $\varDelta\xi(k)$ are solved at time $t=kT$.

5.4 Synchronization of image processing system and robot control system

Generally, the sampling period of the image processing system is much longer than that of the robot control system. Because the sampling period of $\xi^{(0)}(im), \xi^{(1)}(im)$ and $\varDelta\xi(im)$ is m times of the sampling period of $\theta(k)$, it is difficult to synchronize $\xi^{(0)}(im), \xi^{(1)}(im), \varDelta\xi(im)$ and $\theta(k)$ by zero-order holder or 1st-order holder. Otherwise, the robot will drastically accelerate or decelerate during the visual feedback control.

In this section, $\xi^{(0)}(im)$ and $\xi^{(1)}(im)$ are processed by the 2nd-order holder $G_{h2}(z)$. For instance, $\xi_l^{(j)}$ is the lth element of $\xi^{(j)}$, and $\xi_l^{(j)}$ is compensated by the 2nd-order curved line from $t=imT$ to $t=(i+1)mT$. At time $t=kT$, $\xi^{(j)}(k)$ $(j=0,1)$ is calculated by

$$\xi^{(j)}(k)=\xi^{(j)}(im)+\frac{2(k-im)^2}{m^2}\left\{\xi^{(j)}[(i+1)m]-\xi^{(j)}(im)\right\}, \quad (0\square k-im<m/2), \tag{24.a}$$

$$\xi^{(j)}(k)=\xi^{(j)}(im)+\left\{1-\frac{2[k-(i+1)m]^2}{m^2}\right\}\left\{\xi^{(j)}[(i+1)m]-\xi^{(j)}(im)\right\}, \quad (m/2\square k-im<m). \tag{24.b}$$
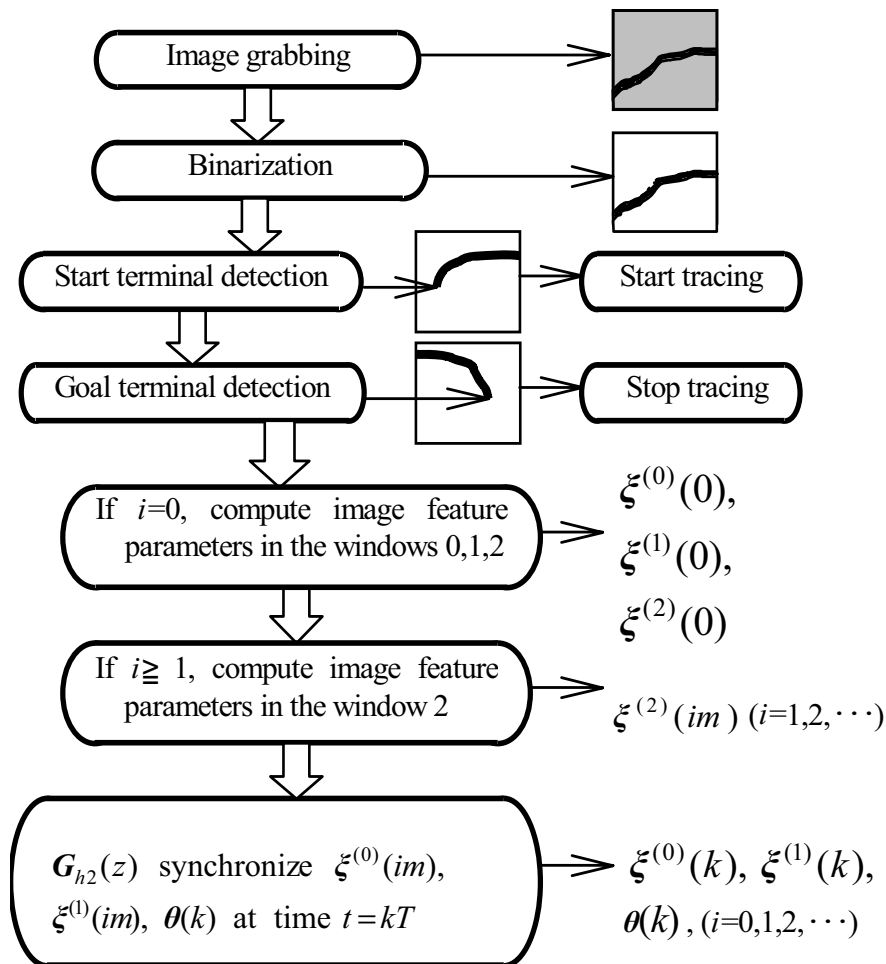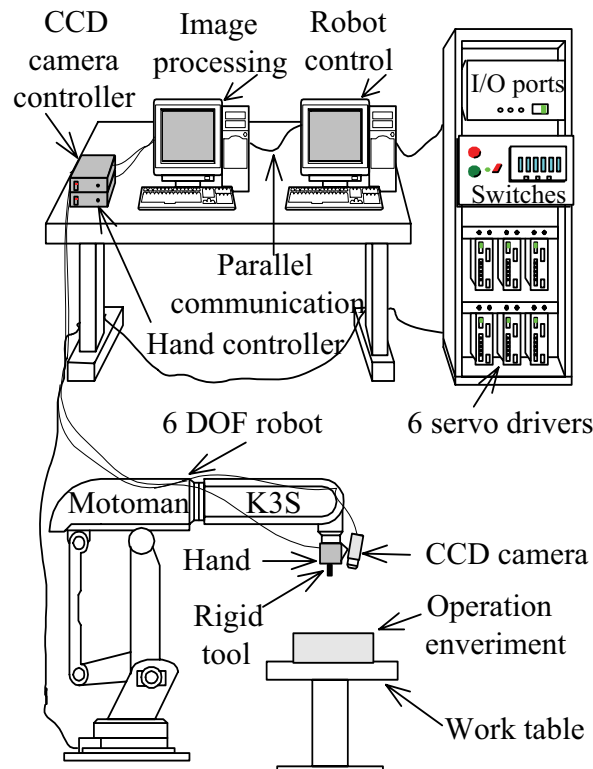

Figure 8. Work sequence of image processing

Figure 9.Trace operation by a robot

6. Experiments

In order to verify the effectiveness of the present approach, the 6 DOF industrial robot is used to trace a curved line. In the experiment, the end-effecter (or the rigid tool) does not contact the curved line. Figure 9 shows the experimental setup. The PC-9821 Xv21 computer realizes the joint control of the robot. The Dell XPS R400 computer processes the images from the CCD camera. The robot hand grasps the rigid tool, which is regarded as the end-effecter of the robot. The diagonal elements of K_P, K_I and K_D are shown in Table 1, the control parameters used in the experiments are listed in Table 2, and the weighting matrix W is set to be an unity matrix I.

Figures 10 and 11 show the position responses (or the learning results of NN and NN_c) $p_{t1}(k)$, $p_{t2}(k)$ and $p_{t3}(k)$ in the directions of x, y and z axes of $\sum O$. $p_{t1}(k)$, $p_{t2}(k)$ and $p_{t3}(k)$ shown in Fig. 10 are teaching data. Figure 11 show the trace responses after the learning of NN_c. Figure 12 shows the learning processes of NN and NN_c as well as the trace errors. E_f converges on 10^{-9} rad², the learning error E^* shown in Fig. 12(b) is given by $E^* = [\sum_{k=0}^{N-1} E_c(k)]/N$, where N=1000. After the 10 trials (10000 iterations) using NN_c, E^* converges on 7.6×10^{-6} rad², and the end-

effecter can correctly trace the curved line. Figure 12(c) shows the trace errors of the end-effecter in x, y, z axis directions of $\sum_O$, and the maximum error is lower than 2 mm.

i	K_P N·m/rad	K_I N·m/rad	K_D N·m/rad
1	25473	0.000039	0.0235
2	8748	0.000114	0.0296
3	11759	0.000085	0.0235
4	228	0.004380	0.0157
5	2664	0.000250	0.0112
6	795	0.001260	0.0107

Table 1. Diagonal ith element of K_P , K_I and K_D

Neuron numbers of NN and NN$_c$	M=18, P=36 U=72, N=6
Sampling period	T =5 ms, mT=160 ms
Control Parameters	η_f =0.9, η_c =0.9, β =1, K=100, S=732
Window size	L=256, W=10

Table 2. Parameters used in the experiment
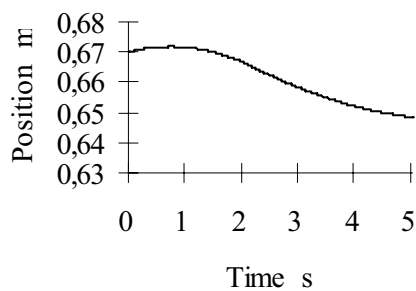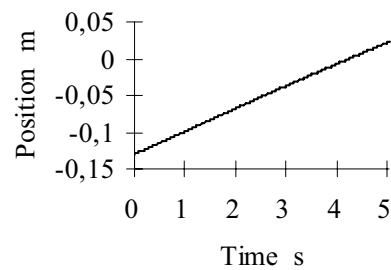
(a)Position p_{t1} in x direction   (b)Position p_{t2} in y direction
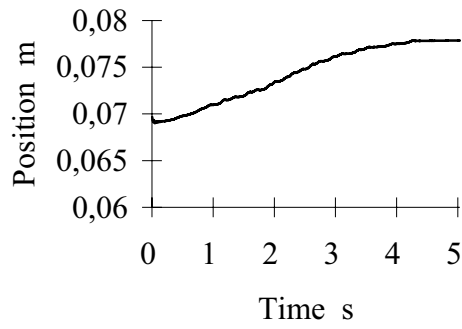
(c)Position p_{t3} in z direction

Figure 10. The teaching data for the learning of NN
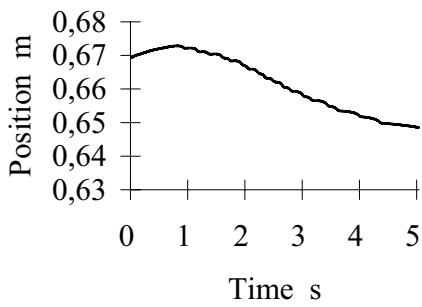
(a)Response p_{t1} in x direction
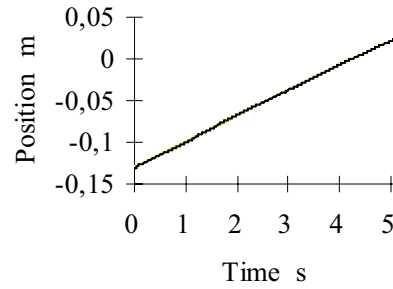
(b)Response p_{t2} in y direction
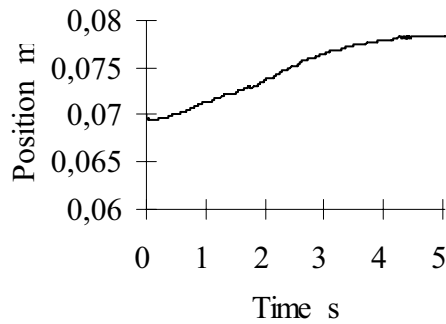
(c)Response p_{t3} in z direction

Figure 11. The trace responses after the learning of NN_c (after 10th trials)

| (a)Learning of NN | (b)Learning of NN$_c$ |

(c)Trace errors in x,y,z directions
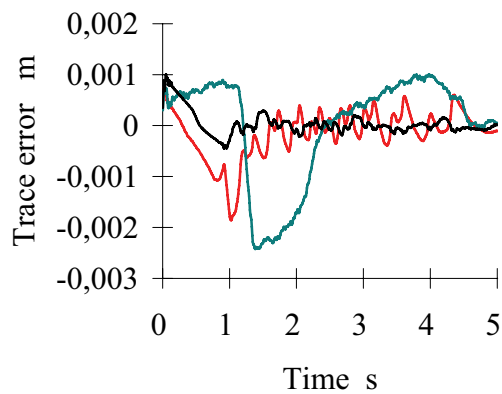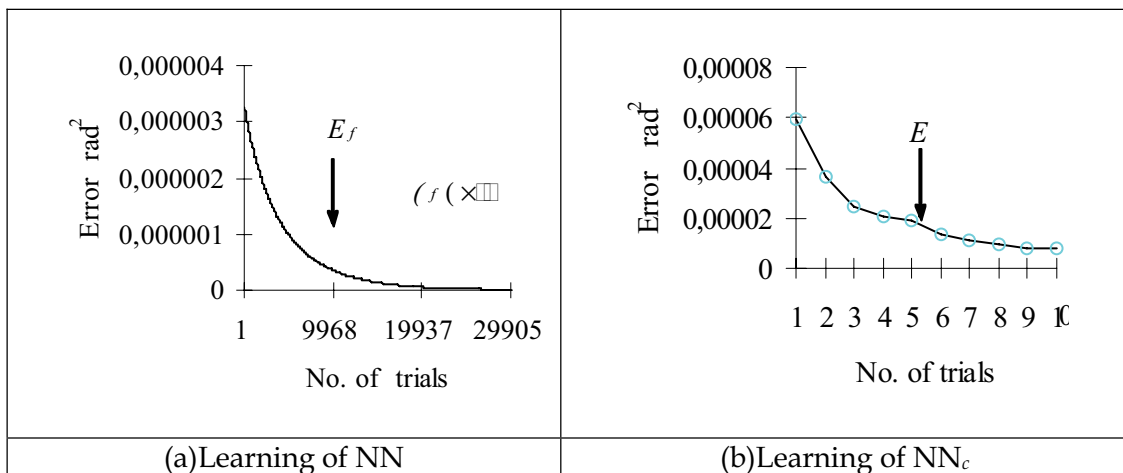
Figure 12. Learning processes and trace errors

7. Conclusions

In this chapter, a visual feedback control approach based on neural network is presented for a robot with a camera installed on its end-effecter to trace an object in an unknown environment. Firstly, the necessary conditions for mapping the image features of the object to be traced to the joint angles of the robot are derived. Secondly, a method is proposed to generate a goal trajectory of the robot by measuring the image feature parameters of the object to be traced. Thirdly, a multilayer neural network is used to learn off-line the mapping in order to produce on line the reference inputs for controlling the robot. Fourthly, a multilayer neural network-based learning controller is designed for the compensation the nonlinear robotic dynamics. Lastly, the effectiveness of

the present approach is verified by tracing a curved line using a 6 DOF industrial robot with a CCD camera installed on its end-effecter.

Through the above research, the following conclusions are obtained:

1. If the mapping relations between the image feature domain of the object and the joint angle domain of the robot are satisfied, NN can learn the mapping relations.
2. By computing the image feature parameters of the object, the goal trajectory for the end-effecter to trace the object can be generated.
3. The approach does not necessitate the tedious CCD camera calibration and the complicated coordinate transformation.
4. Using the 2nd-order holder and the disturbance observer, the synchronization problem and the influences of the disturbances can be solved.

8. References

Hosoda, K. & Asada, M., (1996). Adaptive Visual Servoing Controller with Feed-forward Compensator without Knowledge of True Jacobian, *Journal of Robot Society of Japan*, (in Japanese), Vol. 14, No. 2, pp. 313-319.

Hosoda, K., Igarashi, K. & Asada, M., (1997). Adaptive Hybrid Visual/Force Servoing Control in Unknown Environment, *Journal of Robot Society of Japan*, (in Japanese), Vol. 15, No. 4, pp. 642-647.

Weiss, L.E. & Sanderson, A.C., (1987). Dynamic Sensor-Based Control of Robots with Visual Feedback, *IEEE Journal of Robotics and Automation*, Vol.3, No. 5, pp. 404-417.

Nikolaos, P. & Pradeep, K., (1999). Visual Tracking of a Moving Target by a Camera Mounted on a Robot: A Combination of Control and Vision, *IEEE Journal of Robotics and Automation*, Vol. 9, No. 1, pp. 14-35.

Bernardino, A. & Santos-Victor J, (1999). Binocular Tracking: Integrating Perception and Control, *IEEE Journal Robotics and Automation*, Vol. 15, No. 6, pp. 1080-1093.

Malis, E., Chaumette, F. and Boudet, S., (1999). 2-1/2-D Visual Servoing, *IEEE Journal of Robotics and Automation*, Vol. 15, No. 2, pp. 238-250.

Hashimoto, K., Ebine, T. & Kimura, H., (1996). Visual Servoing with Hand-Eye Manipulator-Optimal Control Approach, *IEEE Journal of Robotics and Automation*, Vol. 12, No. 5, pp. 766-774.

Wilson, J.W., Williams, H. & Bell, G.S., (1996). Relative End-Effecter Control Using Cartesian Position Based Visual Servoing, *IEEE Trans. Robotics and Automation*, Vol. 12, No. 5, pp. 684-696.

Ishikawa, J., Kosuge, K. & Furuta, K., Intelligent Control of Assembling Robot Using Vision Sensor, 13-18 May 1990, Cincinnati, OH, USA, Proceedings 1990 IEEE International Conference on Robotics and Automation (Cat. No.90CH2876-1), Vol. 3, pp. 1904-1909.

Yamada, T. & Yabuta, T., (1991). Some Remarks on Characteristics of Direct Neuro-Controller with Regard to Adaptive Control, *Trans. Soc. Inst. Contr. Eng.*, (in Japanese), Vol. 27, No. 7, pp. 784-791.

Verma, B., (1997). Fast Training of Multilayer Perceptrons, *IEEE Trans. Neural Networks*, Vol. 8, No. 6, pp. 1314-1319.

Joystick Teaching System for Industrial Robots Using Fuzzy Compliance Control

Fusaomi Nagata, Keigo Watanabe and Kazuo Kiguchi

1. Introduction

Industrial robots have been applied to several tasks, such as handling, assembling, painting, deburring and so on (Ferretti et al., 2000), (Her & Kazerooni, 1991), (Liu, 1995), (Takeuchi et al., 1993), so that they have been spread to various fields of manufacturing industries. However, as for the user interface of the robots, conventional teaching systems using a teaching pendant are only provided. For example, in the manufacturing industry of wooden furniture, the operator has to manually input a large mount of teaching points in the case where a workpiece with curved surface is sanded by a robot sander. This task is complicated and time-consuming. To efficiently obtain a desired trajectory along curved surface, we have already considered a novel teaching method assisted by a joystick (Nagata et al., 2000), (Nagata et al., 2001). In teaching mode, the operator can directly control the orientation of the sanding tool attached to the tip of the robot arm by using the joystick. In this case, since the contact force and translational trajectory are controlled automatically, the operator has only to instruct the orientation with no anxiety about overload and non-contact state. However, it is not practical to acquire sequential teaching points with normal directions, adjusting the tool's orientation only with operator's eyes.

When handy air-driven tools are used in robotic sanding, keeping contact with the curved surface of the workpiece along the normal direction is very important to obtain a good surface quality. If the orientation of the sanding tool largely deviates from normal direction, then the kinetic friction force tends to become unstable. Consequently, smooth and uniform surface quality can't be achieved. That is the reason why a novel teaching system that assists the operator is now being expected in the manufacturing field of furniture.

In this paper, an impedance model following force control is first proposed for an industrial robot with an open architecture servo controller. The control law allows the robot to follow a desired contact force through an impedance model in Cartesian space. And, a fuzzy compliance control is also presented for an advanced joystick teaching system, which can provide the friction force acting

between the sanding tool and workpiece to the operator (Nagata et al., 2001). The joystick has a virtual spring-damper system, in which the component of stiffness is suitably varied according to the undesirable friction force, by using a simple fuzzy reasoning method. If an undesirable friction force occurs in teaching process, the joystick is controlled with low compliance. Thus, the operator can feel the friction force thorough the variation of joystick's compliance and recover the orientation of the sanding tool. We apply the joystick teaching using the fuzzy compliance control to a teaching task in which an industrial robot FS-20 with an open architecture servo controller profiles the curved surface of a wooden workpiece. Teaching experimental results demonstrate the effectiveness and promise of the proposed teaching system.

2. Impedance Model Following Force Control

More than two decades ago, two representative force control methods were proposed (Raibert, 1981), (Hogan, 1985) ; controllers using such methods have been advanced and further applied to various types of robots. However, in order to realize a satisfactory robotic sanding system based on an industrial robot, deeper considerations and novel designs are needed. Regarding the force control, we use the impedance model following force control that can be easily applied to industrial robots with an open architecture servo controller (Nagata et al., 2002). The desired impedance equation for Cartesian-based control of a robot manipulator is designed by

$$M_d(\ddot{x} - \ddot{x}_d) + B_d(\dot{x} - \dot{x}_d) + SK_d(x - x_d) = SF + (I - S)K_f(F - F_d) \qquad (1)$$

where $x \in \mathfrak{R}^3$, $\dot{x} \in \mathfrak{R}^3$ and $\ddot{x} \in \mathfrak{R}^3$ are the position, velocity and acceleration vectors, respectively. $M_d \in \mathfrak{R}^{3\times3}$, $B_d \in \mathfrak{R}^{3\times3}$ and $K_d \in \mathfrak{R}^{3\times3}$ called impedance parameters are the coefficient matrices of the desired mass, damping and stiffness, respectively. $F \in \mathfrak{R}^3$ is the force vector acting between the end-effector and its environment. $K_f \in \mathfrak{R}^{3\times3}$ is the force feedback gain matrix. x_d, $\dot{x}_d$, $\ddot{x}_d$ and F_d^T are the desired position, velocity, acceleration and force vector; S and I are the switch matrix diag(S_1, S_2, S_3) and identity matrix. It is assumed that M_d, B_d, K_d and K_f are positive definite diagonal matrices. Note that if $S = I$, then Eq. (1) becomes an impedance control system in all directions; whereas if S is the zero matrix, it becomes a force control system in all directions. If the force control is used in all direction, $X = \dot{x} - \dot{x}_d$ gives

$$\ddot{X} = -M_d^{-1}B_d X + M_d^{-1}K_f(F - F_d) \qquad (2)$$
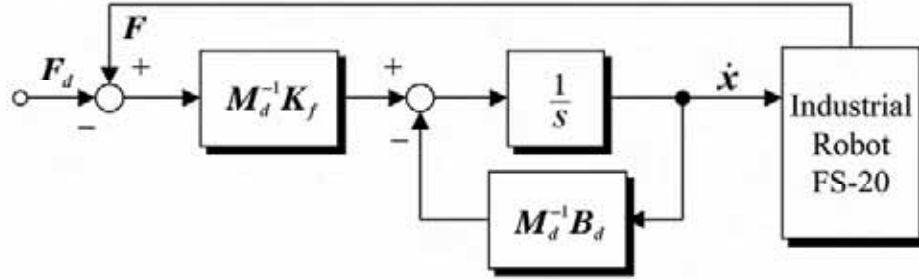
Figure 1. Block diagram of the impedance model following force control.

In general, Eq. (2) is solved as

$$X = \exp\left(- M_d^{-1} B_d t\right) X(0) + \int_0^t \exp\left\{- M_d^{-1} B_d (t - \tau)\right\} M_d^{-1} K_f (F - F_d) d\tau \qquad (3)$$

Here, we will consider the form in the discrete time k using a sampling width Δt. It is assumed that F is constant within $\Delta t(k-1) \le t < \Delta tk$ and diagonal components of M_d, B_d, K_d and K_f are given constant values. Defining $X(k) = X(t)|_{t=\Delta tk}$, it follows that

$$X(k) = \exp\left(- M_d^{-1} B_d \Delta t\right) X(k-1) - \left\{\exp(-M_d^{-1} B_d \Delta t) - I\right\} B_d^{-1} K_f \{F(k) - F_d\} \qquad (4)$$

Remembering $X(k) = \dot{x}(k) - \dot{x}_d(k)$ and setting $\dot{x}_d(k) = 0$ in the direction of force control, a recursive equation of velocity command in Cartesian space is derived by

$$\dot{x}(k) = \exp\left(- M_d^{-1} B_d \Delta t\right) \dot{x}(k-1) - \left\{\exp(-M_d^{-1} B_d \Delta t) - I\right\} B_d^{-1} K_f \{F(k) - F_d\} \qquad (5)$$

where $x(k)$ is composed of position vector $[x(k)\, y(k)\, z(k)]^T$. The manipulated variable $\dot{x}(k)$ is given to the normal direction to a workpiece. Figure 1 shows the block diagram of the impedance model following force control in *s*-domain.

Profiling control is the basic strategy for sanding or polishing, and it is performed by both force control and position/orientation control. However, it is very difficult to realize stable profiling control under such environments that have unknown dynamics or shape. Undesirable oscillations and non-contact state tend to occur. To reduce such undesirable influences, an integral action is added to Eq. (5), which yields

$$v_f(k) = exp\left(-M_d^{-1}B_d\Delta t\right)\dot{x}(k-1) - \left\{ exp(-M_d^{-1}B_d\Delta t) - I \right\}$$
$$B_d^{-1}K_f\left\{F(k) - F_d\right\} + K_i\sum_{n=1}^{k}\left\{F(n) - F_d\right\}$$
(6)

where $K_i = \text{diag}(K_{i1}, K_{i2}, K_{i3})$ is the integral gain. The manipulated variable $v_f(k)$ given by Eq. (6) is also substituted into the reference of the Cartesian based servo controller incorporated in an industrial robot, so that the contact force $F(k)$ can track the reference F_d through the impedance model.
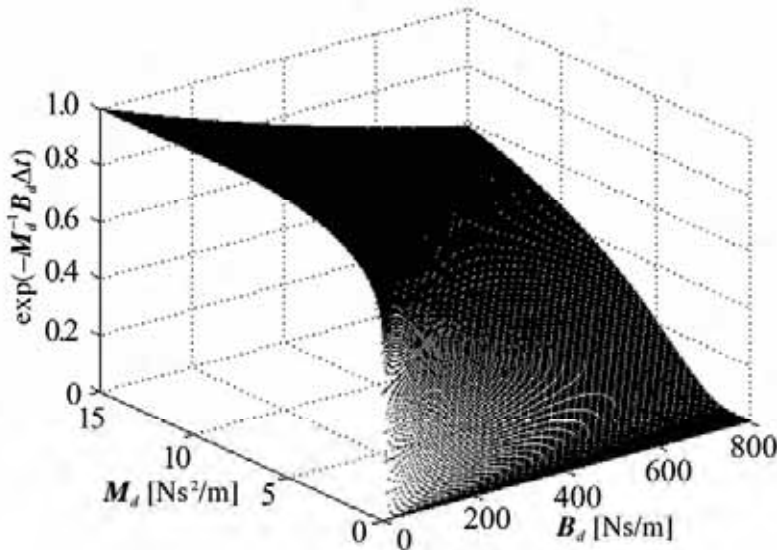
Figure 2. Relation among desired mass M_{di}, damping B_{di} and $\exp(-M_{di}^{-1}B_{di}\Delta t)$

From Eq. (6), the following characteristics are seen. Among the impedance parameters, desired damping has much influence on force control response as well as the force feedback gain. The larger B_d becomes, the smaller the effectiveness of force feedback becomes. Figure 2 shows the relation among M_{di}, B_{di} and diagonal elements of transition matrix $\exp\left(-M_{di}^{-1}B_{di}\Delta t\right)$ in the case that Δt is set to 0.01 [s]. i denotes the i-th (i =1, 2, 3) diagonal element□As can be seen, for example, if B_{di} is smaller than about 100, then appropriate M_{di} is limited. M_{di} over 15 leads $\exp\left(-M_{di}^{-1}B_{di}\Delta t\right)$ to almost 1. In selecting the impedance parameters, their combinations should be noted.

3. Fuzzy Compliance Control of a Joystick Device

3.1 Fuzzy Compliance Control

In our proposed teaching system, the joystick is used to control the orientation of the sanding tool attached to the top of the robot arm. The rotational velocity of the orientation is generated based on the values of the encoder in x- and y-rotational directions as shown in Fig. 3. Also, the compliance of the joystick is varied according to the kinetic friction force acting between a sanding tool and workpiece. As the friction force becomes large, the joint of the joystick is controlled more stiffly. Therefore, the operator can perform teaching tasks having the change of the friction force with the joystick's compliance.

The desired compliance equation for the joint-based control of a joystick is designed by

$$\boldsymbol{\tau}_J = \boldsymbol{B}_J \dot{\boldsymbol{\theta}}_J + \widetilde{\boldsymbol{K}}_J \boldsymbol{\theta}_J \tag{7}$$

where $\boldsymbol{\tau}_J \in \Re^2$ is the joint driving torque vector of the joystick. $\boldsymbol{\theta}_J \in \Re^2$ and $\dot{\boldsymbol{\theta}}_J \in \Re^2$ are the inclination angle and the angular velocity vectors, respectively. $\boldsymbol{B}_J = \mathrm{diag}(B_{Jx}, B_{Jy})$ and $\widetilde{\boldsymbol{K}}_J = \mathrm{diag}(\widetilde{K}_{Jx}, \widetilde{K}_{Jy})$ are the virtual damper and stiffness matrices of the joystick joints. The subscripts x, y denotes x- and y-directional components in Fig. 3, respectively.
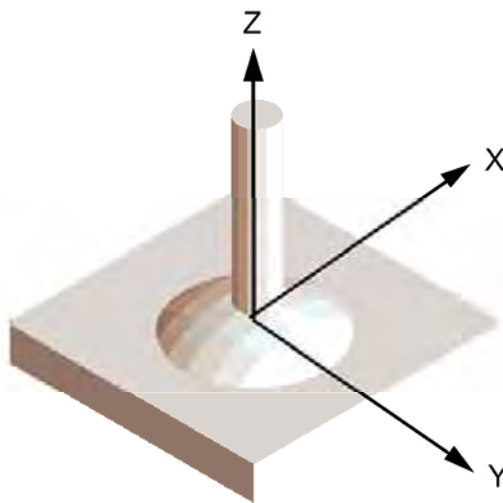

Figure 3. Coordinate system of a joystick

Further, to adjust the compliance of the joystick according to the friction force, $\widetilde{\boldsymbol{K}}_J$ is defined as

$$\begin{pmatrix} \widetilde{K}_{Jx} & 0 \\ 0 & \widetilde{K}_{Jy} \end{pmatrix} = \begin{pmatrix} K_{Jx} & 0 \\ 0 & K_{Jy} \end{pmatrix} + \begin{pmatrix} \Delta K_{Jx} & 0 \\ 0 & \Delta K_{Jy} \end{pmatrix} \tag{8}$$

where $\boldsymbol{K}_J = \mathrm{diag}(K_{Jx}, K_{Jy})$ is the base stiffness matrix, $\Delta \boldsymbol{K}_J = \mathrm{diag}(\Delta K_{Jx}, \Delta K_{Jy})$ is the compensated stiffness matrix whose diagonal elements are suitably given from the following fuzzy reasoning part.

3.2 Generation of Compensated Stiffness Using Simple Fuzzy Reasoning

In this section, we discuss how to suitably generate the compensated stiffness according to the undesirable friction force. The compensated stiffness is adjusted by using a simple fuzzy reasoning method, so that the teaching operator can conduct the teaching task delicately feeling the friction force acting between the sanding tool and workpiece through the compliance of the joystick. In teaching, x- and y-directional frictions F_x and F_y in the base coordinate system are used as fuzzy inputs for the fuzzy reasoning, and they are used to estimate y- and x-rotational compliance of the joystick joints. The present fuzzy rules are described as follows:

Rule 1:   If $|F_x|$ is $\widetilde{A}_{x1}$ and $|F_y|$ is $\widetilde{A}_{y1}$, Then $\Delta K_{Jx} = B_{x1}$ and $\Delta K_{Jy} = B_{y1}$

Rule 2:   If $|F_x|$ is $\widetilde{A}_{x2}$ and $|F_y|$ is $\widetilde{A}_{y2}$, Then $\Delta K_{Jx} = B_{x2}$ and $\Delta K_{Jy} = B_{y2}$

Rule 3:   If $|F_x|$ is $\widetilde{A}_{x3}$ and $|F_y|$ is $\widetilde{A}_{y3}$, Then $\Delta K_{Jx} = B_{x3}$ and $\Delta K_{Jy} = B_{y3}$

$$\vdots$$

Rule L: If $|F_x|$ is $\widetilde{A}_{xL}$ and $|F_y|$ is $\widetilde{A}_{yL}$, Then $\Delta K_{Jx} = B_{xL}$ and $\Delta K_{Jy} = B_{yL}$

Where $\widetilde{A}_{xi}$ and $\widetilde{A}_{yi}$ are i-th ($i=1,\dots,L$) antecedent fuzzy sets for $|F_x|$ and $|F_y|$, respectively. L is the number of the fuzzy rules. B_{xi} and B_{yi} are the consequent constant values which represent i-th x- and y-rotational compensated stiffness, respectively. In this case, the antecedent confidence calculated by i-th fuzzy rule is given by

$$\omega_i = \mu_{Axi}(|F_x|) \wedge \mu_{Ayi}(|F_y|) \tag{9}$$

where $\mu_X(\cdot)$ is the Gaussian type membership function for a fuzzy set represented by

$$\mu_X(x) = \exp\left\{\log\frac{(x-\alpha)^2\beta^2}{2}\right\} \tag{10}$$

where α and β are the center of membership function and reciprocal value of standard deviation, respectively.
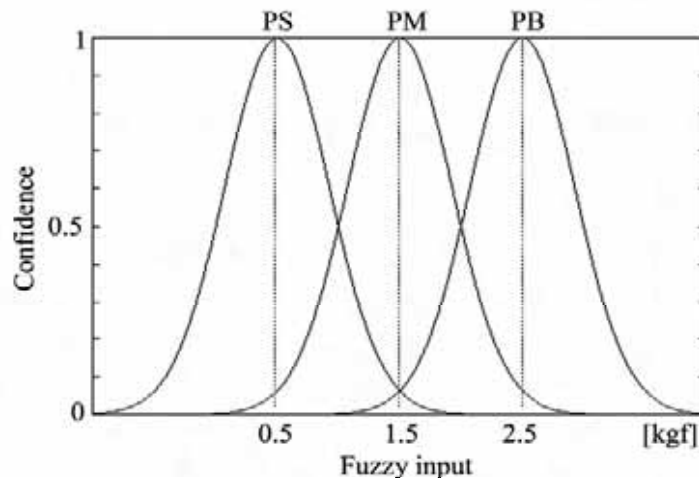
Figure 4, Antecedent membership function for $|F_x|$ and

$|F_x|$	$|F_y|$		
	PB	PM	PS
PB	$(4.0K_{Jx}, 4.0K_{Jy})$	$(3.0K_{Jx}, 4.0K_{Jy})$	$(0.4K_{Jx}, 4.0K_{Jy})$
PM	$(4.0K_{Jx}, 3.0K_{Jy})$	$(3.0K_{Jx}, 3.0K_{Jy})$	$(0.4K_{Jx}, 3.0K_{Jy})$
PS	$(4.0K_{Jx}, 0.4K_{Jy})$	$(3.0K_{Jx}, 0.4K_{Jy})$	$(0.4K_{Jx}, 0.4K_{Jy})$

Table 1. Constant values in the consequent part.

In the sequal, the compensated stiffness matrix ΔK_J is obtained from the weighted mean method given by

$$\Delta K_J = \text{diag}\left\{\frac{\sum_{i=1}^{L} B_{xi}\omega_i}{\sum_{i=1}^{L}\omega_i}, \frac{\sum_{i=1}^{L} B_{yi}\omega_i}{\sum_{i=1}^{L}\omega_i}\right\} \tag{11}$$

Figure 4 shows the designed antecedent membership functions. On the other hand, the designed consequent constants, which represent the compensated values of the stiffness, are tabulated in Table 1. In teaching experiments, the friction force more than 3 kgf is regarded as an overload. If such an overload is detected, then the teaching task is automatically stopped and the polishing tool is immediately removed from the workpiece. Therefore, the support set of range [0, 3] in Fig. 3 is used for the antecedent part.

4. Teaching Experiment

4.1 Sanding Robot System

Throughout the remainder of this paper, the effectiveness of the proposed teaching method is proved by teaching experiments.

Photo 1. Robotic sanding system.
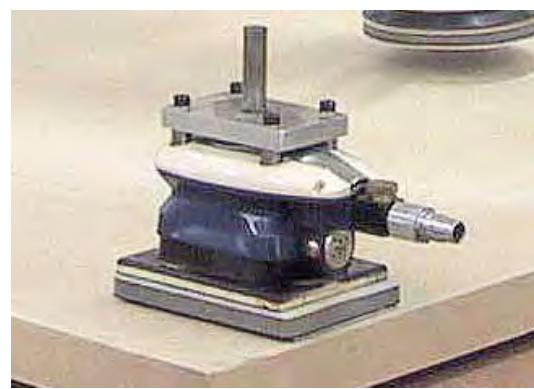
Photo 2 Air-driven sanding tool.

Photo 3. Joystick system used in teaching experiments (Impulse Engine2000).
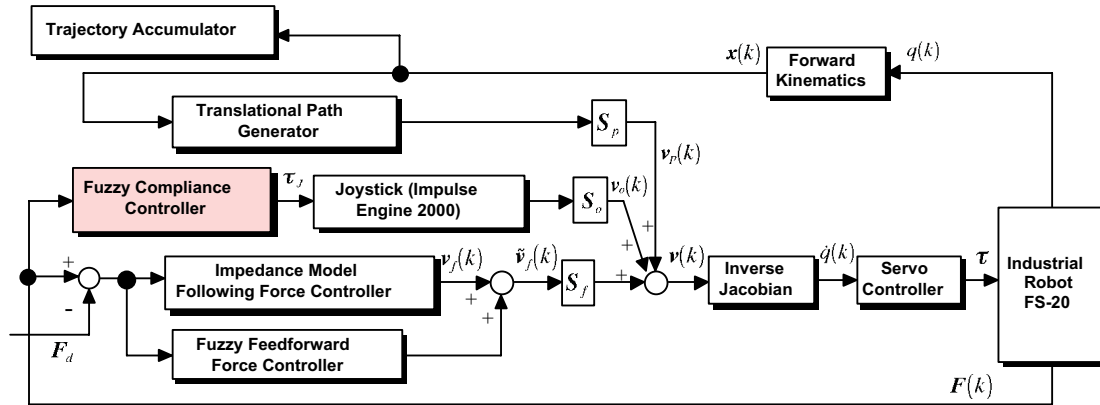
Figure 5. Block diagram of the sanding robot in teaching mode.

Photo 1 shows the overview of the sanding robot used in the teaching experiments. The base 6-DOF industrial robot with an open architecture servo controller is the model FS-20 provided by Kawasaki Heavy Industries, whose tip of the arm has an air-driven sanding tool as shown in Photo 2 via a 6-DOF force/torque sensor 67M25A provided by Nitta corporation. The permitted weight of workpieceis under 20 kgf. The size of the sanding tool is 60 × 100 mm² and its paper roughness is #120. Since this type of tool tends to cause not only high frequency but also large magnitude vibrations, we use the force sensor's filter whose cutoff frequency is set to 30 Hz. Photo 3 shows the 2-DOF joystick Impulse Engine2000 provided by Immersion corporation. This joystick can perform a maximum force of 8.9 N by controlling the joint torque with 2048 steps. In teaching experiments, we apply the fuzzy compliance control given by Eq. (7) to this joystick.

Figure 5 shows the block diagram of the sanding robot in teaching mode. The proposed teaching process is as follows: in the direction of position control, the translational trajectory generator yields a base trajectory such as a zigzag path and whirl path with a velocity command $v_p(k)$. In the direction of orientation control, a rotational velocity $v_o(k)$ is generated using the compensated angle of inclination $\widetilde{\boldsymbol{\theta}}_J = \left[\widetilde{\theta}_{Jx}\ \widetilde{\theta}_{Jy}\right]^T$ with a valocity transformation gain $\boldsymbol{K}_v$. $\widetilde{\theta}_{Ji}$ ($i = x$, y) is obtained by

$$
\widetilde{\theta}_{Ji} = \begin{cases} 0 & \text{if} \quad -500 \le \theta_{Ji} \le 500 \\ \theta_{Ji} - 500 & \text{if} \quad \theta_{Ji} > 500 \\ \theta_{Ji} + 500 & \text{if} \quad \theta_{Ji} < -500 \end{cases}
\tag{12}
$$

Note that $\boldsymbol{\theta}_J = \begin{bmatrix} \theta_{Jx} \ \theta_{Jy} \end{bmatrix}^T$ in Eqs. (12) and (7) are the same variable. In this case, the teaching operator can conduct the teaching task feeling the friction force acting between the sanding tool and workpiece with the compliance of the joystick. In the direction of force control, the impedance model following force controller given by Eq. (6) yields $\boldsymbol{v}_f(k)$, in which $\boldsymbol{v}_f(k)$ is added to the output from the already proposed fuzzy feedforward force controller (Nagata et al., 1999) to generate $\tilde{\boldsymbol{v}}_f(k)$. After switched by S_p, S_o and S_f, each directional velocity command is summed up to compose a velocity vector $\boldsymbol{v}(k)$. $\boldsymbol{v}(k)$ is transformed into a joint angle velocity $\dot{\boldsymbol{q}}(k)$ with the inverse Jacobian to give to the servo controller.

4.2 Teaching Experiments

In order to examine the effectiveness of the proposed teaching system, an experiment as shown in Photo 1 was conducted using a workpiece machined by a 5-axis NC machine tool. Figure 6 shows the CAD model of the workpiece. The teaching was carried out under the following conditions: the air power of the sanding tool is switched off; the profiling velocity in the tangent direction is set to 20 mm/s; the desired contact force in the normal direction is set to 1 kgf; and the sanding tool moves from the point A to the point B in Photo 1.
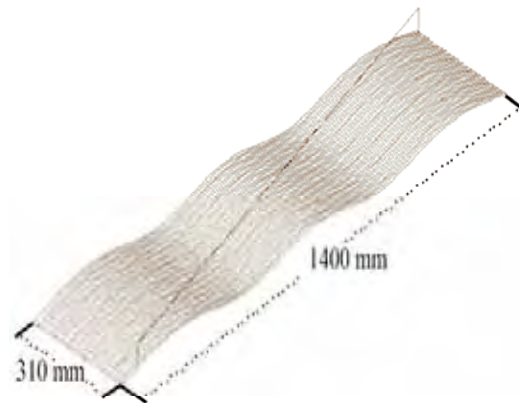
Figure 6. CAD Model of a workpiece

Photo 1. Teaching scene by using the proposed system
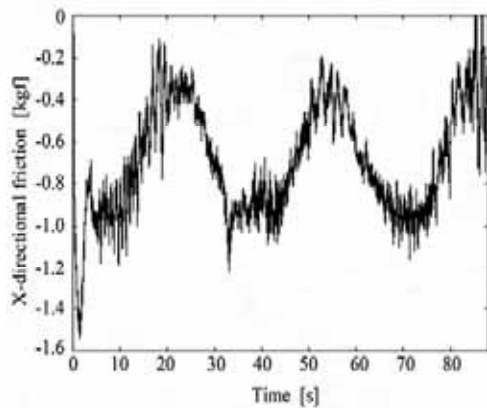
Figure 7. X-directional friction force
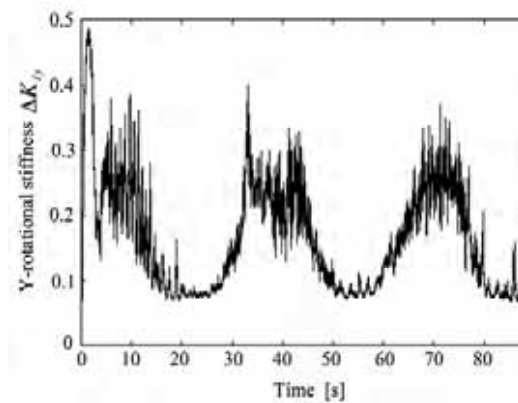
Figure 8. Y-rotational stiffnes generated by fuzzy resoning

The base compliance of the joystick is set to $K_{Jx} = K_{Jy} = 0.167$, $B_{Jx} = B_{Jy} = 0.5$. Table 2 shows the control parameters given in the experiment. After these preparations, an experiment on the proposed joystick teaching was done. Photo 4 shows the teaching scene by using the proposed teaching system. Figures 7 and 8 show x-directional friction force f_x and y-rotational component ΔK_{Jy} of the compensated stiffness matrix ΔK_J, respectively.

Desired contact force $\sqrt{(F_x)^2 + (F_z)^2}$	1 [kgf]
Desired mass coefficient M_{d1}, M_{d3}	0.01 [kgf·s²/mm]
Desired damping coeficinet B_{d1}, B_{d3}	10 [kgf·s/mm]
Force feedback gain K_{f1}, K_{f3}	1
Angle velocity gain K_{vx}, K_{vy}	0.03
Profiling velocity $|v_p|$	20 [mm/s]
Sampling width Δt	10 [msec]

Table 2. Designed control parameters in teaching mode
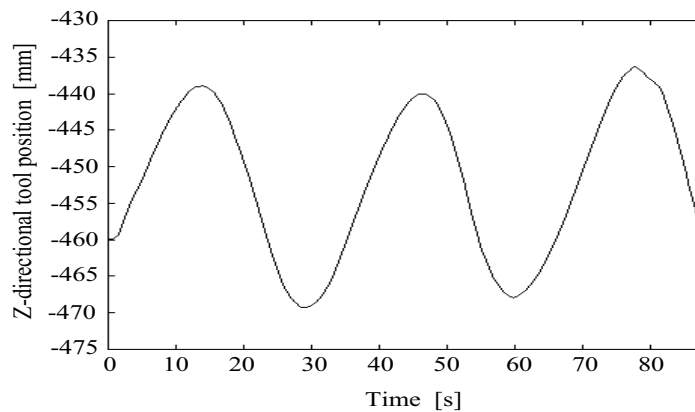
Figure 9. Obtained tip position in the *z*-direction

It is observed from the result that the compliance of the joystick changes according to the friction acting between the sanding tool and workpiece. Thus, the operator could execute the teaching task feeling the friction force with the compliance of the joystick. In teaching, the time series data of both the position and orientation were stored into the trajectory accumulator as shown in Fig. 5. Figure 9 shows the *z*-directional position obtained by this teaching.

4.3 Sanding Task Using the Acquired Trajectory

Figure 10 shows the block diagram of the sandinging robot in playback mode. An experiment on polishing task was carried out using the acquired trajectory. In this case, although the tangent profiling velocity was set to 40 mm/s which was two times as fast as that in teaching mode, the polishing task could be stably practiced. The *z*-directional force control result is plotted in Fig. 11.
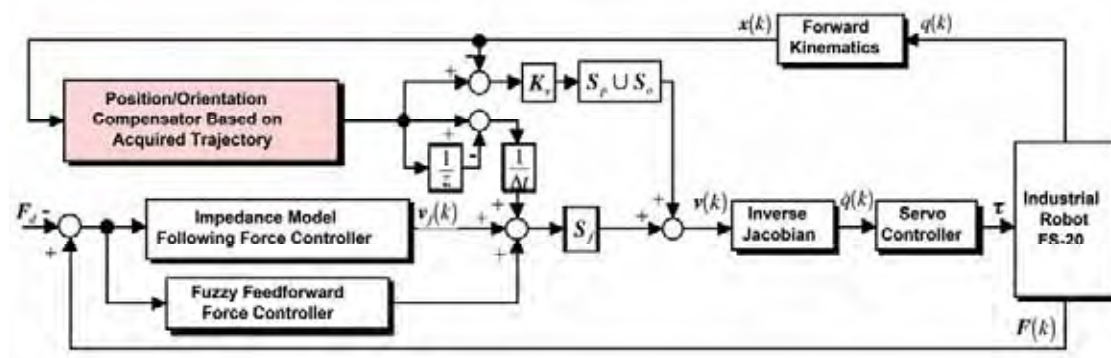
Figure 10. Block diagram of sending robot in diagram in playback mode using joystick taught data.

It has been observed that a desirable response is obtained in spite of tool's large vibrations. Furthermore, the surface accuracy of the workpiece was so good condition as well as polished by skilled workers. The measurements evaluated by arithmetical mean roughness method were less than 2µm.
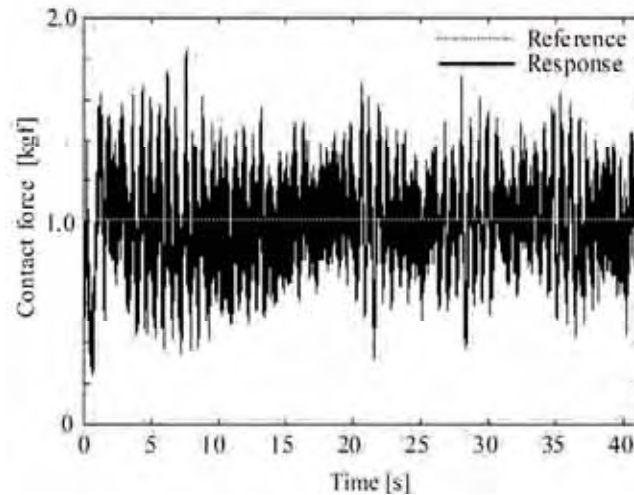
Figure 11. Force control result in playback mode

5. Conclusion

In this paper, a joystick teaching system using a fuzzy compliance control has been proposed for industrial robots. We have applied the proposed teaching system to a teaching task of a furniture sanding robot. Experimentally, it was demonstrated that the operator could safely carry out the teaching task feeling the friction force acting between a sanding tool and workpiece through the compliance of the joystick.

The proposed teaching process is as follows: first, a zigzag path considered according to both sizes of each work and sanding tool is prepared; next, the sanding robot, in which an impedance model following force control method is incorporated, profiles the surface of the workpiece along the zigzag path. The operator has only to control the orientation of the sanding tool using the fuzzy compliance controlled joystick so that the tool and workpiece can be in contact each other keeping the desired relation of position and orientation. Since the force controller keeps the contact force a desired value, the operator has to give no attention to a sudden over-load or non-contact state. The desired trajectory is automatically obtained as the data including continuous information of the position and orientation along the zigzag path on the workpiece surface. In playback mode, the robot can finally achieve the sanding task without any assists of the operator by referring the acquired trajectory.

6. References

Ferretti, G.; Magnani, G. & Rocco, P. (2000). Triangular Force/Position Control with Application to Robotic Deburring, *Machine Intelligence & Robotic Control*, Vol. 2, No. 2, pp. 83-91.

Her, M. & Kazerooni, H. (1991). Automated Robotic Deburring of Parts Using Compliance Control, *ASME Journal of Dynamic systems, Measurement, and Control*, Vol. 113, pp. 60-66.

Hogan, N. (1985). Impedance Control: An Approach to Manipulation, *Trans. ASME, J. Dyn. Syst. Measure. Contr.*, Vol. 107, pp. 1–24.

Liu, M. H. (1995). Force-Controlled Fuzzy-Logic-Based Robotic Deburring, *Control Engineering Practice*, Vol. 3, No. 2, pp. 189-201.

Nagata, F.; Watanabe, K. & Izumi, K. (1999). Position-Based Impedance Control Using a Fuzzy Compensator, *Procs. of the 3rd International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES'99)*, pp. 125-128.

Nagata, F.; Watanabe, K.; Hashino, S.; Tanaka, H.; Matsuyama, T. & Hara, K. (2000). Polishing Robot Using a Joystick Controlled Teaching System, *CD-ROM Procs. of the IEEE International Conference on Industrial Electronics, Control and Instrumentation (IECON-2000)*, pp. 632-637.

Nagata, F.; Watanabe, K.; Hashino, S.; Tanaka, H.; Matsuyama, T. & Hara, K. (2001). Polishing Robot Using Joystick Controlled Teaching, *Journal of Robotics and Mechatronics*, Vol. 13, No. 5, pp. 517-525.

Nagata, F.; Watanabe, K.; Kiguchi, K.; Tsuda, K.; Kawaguchi, S.; Noda, Y. & Komino, M. (2001). Joystick Teaching System for Polishing Robots Using Fuzzy Compliance Control, *Procs. of 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 362-367.

Nagata, F.; Watanabe, K.; Fujimoto, Y.; Kiguchi, K. & Murase, Y. (2002). 3D Machining and Finishing System for New Designed Furniture, *Procs. of 2002 Japan-USA Symposium on Flexible Automation*, pp. 1239-1245.

Raibert, M. H. & Craig, J. J. (1981). Hybrid Position/Force Control of Manipulators, *Trans. ASME, J. Dyn. Syst. Measure. Contr.*, Vol. 102, pp. 126–133.

Takeuchi, Y.; Ge, D. & Asakawa, N. (1993). Automated Polishing Process with a Human-like Dexterous Robot, *Procs. of IEEE International Conference Robotics and Automation*, pp. 950-956.

Forcefree Control for Flexible Motion of Industrial Articulated Robot Arms

Satoru Goto

1. Introduction

Many industrial robot arms are operated in industry, and some robotic applications in the industry, such as a pulling-out of products made by die casting, require the flexible motion excited by an external force. Here, the flexible motion means that the robot arm moves passively according to the external force. Industrial robot arms, however, are difficult to be moved by the external force because the servo controller of the industrial robot arm controls the motion of the robot arm excited by an input signal responsible for the motion. The torque generated by the external force is a kind of disturbance for the robot control system and it can be compensated by the servo controller.

Impedance control (Hogan 1985; Scivicco & Siciliano, 2000) and compliance control (Mason, 1981; Micheal et al., 1982) were proposed in order to achieve the flexible motion, and these methods have been applied to industrial robots (Ciro et al., 2000). Most of these control methods impose desired dynamic characteristics between an end-effector and an environment by setting inertia, friction and stiffness. Usually an elastic spring behavior is introduced in order to achieve the flexible motion of the robot arm. The potential force of the elastic spring behavior is a conservative force, and it is impossible to achieve the passive motion away from the environment caused by the external force is impossible to be achieved by using these control methods.

In this research, the forcefree control, which achieves the passive motion of the robot arm according to the external force, is proposed. Moreover, the forcefree control is extended to the forcefree control with independent compensation, the forcefree control with assigned locus and the position information based forcefree control. The effectiveness of the proposed forcefree control is assured by comparing the experimental results with simulation results. Comparison between the forcefree control and other force control methods such as impedance control are also described. Finally, applications of the forcefree control of pull-out work, direct teaching and rehabilitation robot are demonstrated.

2. Forcefree Control

2.1 Necessity of Forcefree Control

Operations such as cutting and welding can be easily achieved by using industrial robot arms. These operations are carried out through contour control in that the tip of the robot arm moves along a given path, and point-to-point (PTP) control in that the tip moves between previously assigned points. These operations are tractable as the rigidity of the robot arm advances. Therefore, the industrial robot arms are designed with high rigidity.

Recently, control of a contact force is required in order to carry out assembling, handling, inlaying, pull-out work and grinder operations. On the contrary, it is difficult to control the contact force if the robot arm possesses high rigidity. In view of this, the property of low rigidity for the industrial robot arms is required to control the contact force.

On the other hand, flexible motion is also required for the safety operation such as contact between the robot and human operator. Generally, an emergency shutdown switch is built in to the servo controller. When the operator is sandwiched between the tip of the robot arm and the environment, the emergency halt becomes more dangerous. If the robot arm can be actuated with flexibility, the operator is released from the sandwiched situation.

Some of the problems have been solved by particularly designed robots or by remodeling of robots. However, a number of industrial robot arms are manufactured for general purpose, and such robot arms have a lower cost compared with that of special purpose robots. In this view, flexible motion by general purpose industrial robot arms is required in industry.

A servo controller of industrial robot arm includes a position loop and a velocity loop. Input to the industrial robot arm is usually the joint position of each link. Hence, the industrial robot arms should be considered as the combination of the mechanism of the robot arm and the servo controller.

Recently, a study of force control of industrial robots has developed rapidly and the achievement of such force control has been the major concern. A number of force control methods for the change of rigidity of robot arms such as impedance control (Hogan 1985; Scivicco & Siciliano, 2000) and compliance control (Mason, 1981; Mecheal et al., 1982) have been proposed. These methods are apparently good enough to achieve the requirement of force control. However, to apply these methods in industry, there are difficult problems to be solved. For general purpose robots including the servo controller, these methods require changes in the control strategy in the servo controller.

Modification of the servo controller is almost impossible from the user side, and modification by the maker is very expensive even upon request from the user side. Presently, available methods for the achievement of flexible contact

between the tip of the robot arm and the environment are almost achieved by attaching a flexible device on the tip of the robot arm.

The forcefree control can achieve the flexible motion of the industrial robot arms under virtual circumstances of non-gravity and non-friction without any change of the built-in controller. By use of the forcefree control, the robot arm moves passively according to the external force directly as if it were under the circumstances of non-friction and non-gravity. The mathematical explanation of the forcefree control is described below.

2.2 Derivation of Forcefree Control

Dynamics of an articulated robot arm is expressed by

$$H(q)\ddot{q} + D\dot{q} + N_{\mu}\text{sgn}(\dot{q}) + h(q,\dot{q}) + g(q) = \tau_s + \tau_f \tag{1}$$

where $H(q)$ is the inertia matrix, $D\dot{q} + N_{\mu}\text{sgn}(\dot{q})$ is the friction term, $h(q,\dot{q})$ is the coupling nonlinear term, $g(q)$ is the gravity term, q is the output of joint angle, τ_s is the torque input to the robot arm and τ_f is the joint torque corresponding to the external force f acting on the tip of the robot arm (Fu et al., 1987).

Figure 1. Block diagram of forcefree control

In industrial robot arms, the servo controller is adapted to control the motion of the robot arm. The control loop of the servo controller is shown on the right side of Fig. 1, where K_p, K_v and K_τ are position loop gain, velocity loop gain and torque constant, respectively (Nakamura et al., 2004; Kyura, 1996). The

servo controller adopts P and PI type cascade control, the P controller is used for the position loop control and the PI controller is used for the velocity loop control. The velocity control loop has the role of the derivative action. The servo controller generates the torque input to the robot arm as described by

$$\tau_s = K_\tau \left(K_v \left(K_p \left(q_d - q \right) - \dot{q} \right) \right) + \tau_d + \tau_g - \tau_f \tag{2}$$

where q_d is the input of joint angle, τ_d is the friction compensation torque and τ_g is the gravity compensation torque. As expressed in (2), the servo controller includes the friction compensation and the gravity compensation through integral action of PI control. The friction and the gravity are assumed to be exactly compensated by the servo controller as

$$\tau_d = D\dot{q} + N_\mu \mathrm{sgn}(\dot{q}) \tag{3}$$

$$\tau_g = g(q). \tag{4}$$

The torque caused by an external force τ_f is also compensated by the servo controller because the servo controller of an industrial robot arm is designed such that the stiffness of the robot arm is high enough and the robot arm will never be moved by the external force.
The total dynamic equation of an industrial articulated robot arm including the servo controller is given by substituting (2), (3) and (4) for (1) as

$$H(q)\ddot{q} + h(q,\dot{q}) = K_\tau \left(K_v \left(K_p \left(q_d - q \right) - \dot{q} \right) \right). \tag{5}$$

Forcefree control means that the influences of friction and gravity on the robot arm motion can be compensated. The entire dynamics of the industrial robot arms controlled by the forcefree control is described by

$$H(q)\ddot{q} + h(q,\dot{q}) = \tau_f \tag{6}$$

where τ_f is obtained by substituting (2) for (5) as

$$\tau_f = -\left(\tau_s - \tau_d - \tau_g - \left(H(q)\ddot{q} + h(q,\dot{q}) \right) \right). \tag{7}$$

Generally, the speed of the flexible motion of an industrial robot arm is relatively slow, usually less than 1/5 of the rated speed. Hence, the inertia and nonlinear terms of the robot arm is negligibly small $\left(H(q)\ddot{q} + h(q,\dot{q}) \approx 0 \right)$, and the external torque is approximately given by

$$\tau_f = -\left(\tau_s - \tau_d - \tau_g\right). \tag{8}$$

Finally, the control law of the forcefree control with independent compensation is obtained by substituting (6) and (8) for (5) and by solving for q_d as

$$q_d = K_p^{-1}\left(K_v^{-1}K_\tau^{-1}\left(-\tau_s + \tau_d + \tau_g\right) + \dot{q}\right) + q. \tag{9}$$

Here, τ_s is measured by the torque monitor which is usually attached to the servo controller of the industrial robot arm and is used to check the value of the torque.

2.2.1 Estimation of Friction Term

Friction term τ_d consists of Viscous friction $D\dot{q}$ and Coulomb friction $N_\mu \mathrm{sgn}(\dot{q})$ as in (3). The friction effect to the motion of the robot arm is estimated by the torque output under constant velocity motion. The friction term is obtained through the following procedure;

1. To cancel the effect of the gravity, the robot arm sets around its vertical position;
2. Various constant velocity inputs are applied to each link of the robot arm;
3. Respective torque outputs corresponding to the applied velocities are measured by using the torque monitor;
4. The torque output vs. applied velocities are plotted;
5. Viscous friction coefficient D and magnitude of Coulomb friction N_μ in (3) are estimated by using the least squares method from the collected data.

In order to smoothen the Coulomb friction effect, the Sigmoid function is introduced in the friction term as

$$\tau_d = D\dot{q} + N_\mu f_d(\dot{q}) \tag{10}$$

where

$$f_d(\dot{q}) = \begin{pmatrix} \left(1 - e^{-\gamma\dot{q}_1}\right)/\left(1 + e^{-\gamma\dot{q}_1}\right) \\ \left(1 - e^{-\gamma\dot{q}_2}\right)/\left(1 + e^{-\gamma\dot{q}_2}\right) \\ \vdots \\ \left(1 - e^{-\gamma\dot{q}_n}\right)/\left(1 + e^{-\gamma\dot{q}_n}\right) \end{pmatrix} \tag{11}$$

and the parameter γ is introduced for adjusting the effect of smoothness.

2.2.2 Estimation of Gravity Term

The gravity term is a function of the robot arm position q as (4). Here, the gravity term is modelled by

$$g(q) = U(q)a + V(\dot{q})b \tag{12}$$

where the function

$$V(\dot{q}) = \begin{pmatrix} e^{-\lambda(\dot{q}_1)^2} & 0 & \cdots & 0 \\ 0 & e^{-\lambda(\dot{q}_2)^2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & e^{-\lambda(\dot{q}_n)^2} \end{pmatrix} \tag{13}$$

is introduced in order to smoothen the effect of static friction b. In (13), the parameter λ is introduced for adjusting the effect of smoothness.

The parameters a and b are estimated by using the least squares method from the experimental data of the steady-state torque monitor outputs for various postures of the robot arm. For the estimation of the parameters a and b in (12), the steady-state torque monitor outputs are used because the torque monitor output contains a transient component, which is caused by the integral action of the servo controller. Hence, the gravity compensation torque can be represented by

$$\tau_g = \left(I - e^{At}\right)g(q) \tag{14}$$

where

$$e^{At} = \begin{pmatrix} e^{-t/T_1} & 0 & \cdots & 0 \\ 0 & e^{-t/T_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & e^{-t/T_n} \end{pmatrix}. \tag{15}$$

and $T_i \, (i = 1, \cdots, n)$ are the time constants.

2.3 Alogrithm of Forcefree Control

The algorithm of the forcefree control is explained here. Initial setting of the forcefree control is expressed in the following first 3 items.

1. Servo parameters K_p, K_v and K_τ are obtained from the servo controller;
2. Friction term (10) is estimated as explained in the above section;
3. Gravity term (12) is estimated as explained in the above section.

The execution of the forcefree control has summarized by the following 6 items.

1. External force f is added to the robot arm;
2. Torque monitor detects the external force f;
3. The friction torque τ_d is estimated by (10);
4. The gravity torque τ_g is estimated by (12);
5. External torque τ_f is calculated by (8);
6. The position input q_d is generated by (9).

Finally, the reference position input q_d is given to the servo controller according to the above algorithm and the forcefree control is achieved.

2.4 Verification

Robot arm motion by using the forcefree control was verified by a simulation study and experiments. The simulation study shows an ideal motion of the forcefree control. An industrial articulated robot arm (Performer-MK3S, YAHATA Electric Machinery Mfg., Co., Ltd) was used for the experiment on the forcefree control with independent compensation. Two links of Performer-MK3S were used for the experiment. The link lengths of the robot arm are $l_1 = 0.25\,[\text{m}], l_2 = 0.215\,[\text{m}]$, and masses of the links are $m_1 = 2.86\,[\text{kg}]$, $m_2 = 2.19\,[\text{kg}]$, respectively. The position loop gain was $K_p = \text{diag}(25,25)\,[1/\text{s}]$, the velocity loop gain was $K_v = \text{diag}(150,150)\,[1/\text{s}]$, and the torque constant was $K_\tau = \text{diag}(0.017426, 0.036952)\,[\,\text{Nm}/(\text{rad}/\text{s}^2)]$.
Fig. 2 shows the experimental results of the estimation of friction term. The bold lines show the results of (10) for $\gamma = 120$ and the dotted lines show the results of $D\dot{q} + N_\mu \text{sgn}(\dot{q})$. The step change of the results of $D\dot{q} + N_\mu \text{sgn}(\dot{q})$ can be smoothened using the sigmoid function.
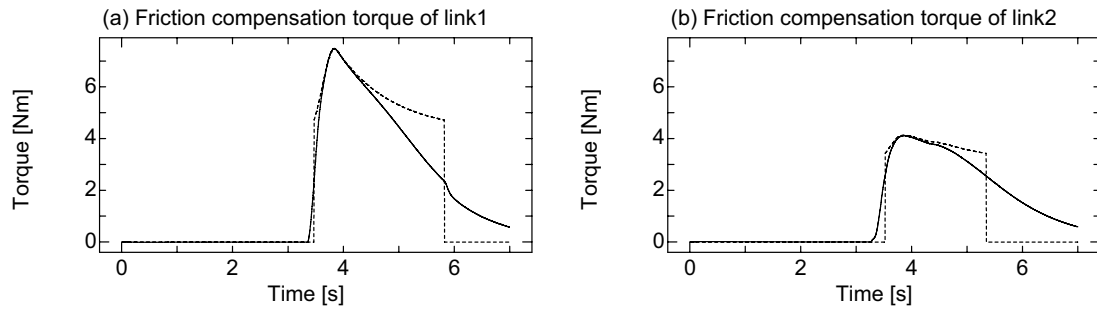
Figure 2. Estimation of friction term

Fig. 3 shows the experimental results of the estimation of gravity term. The bold lines show the results of (12) for $\lambda = 150$ and the dotted lines show the results of $U(q)a + b$. The alternate long and short lines show the result of $U(q)a$. The estimated friction approximately gives the static friction effect.
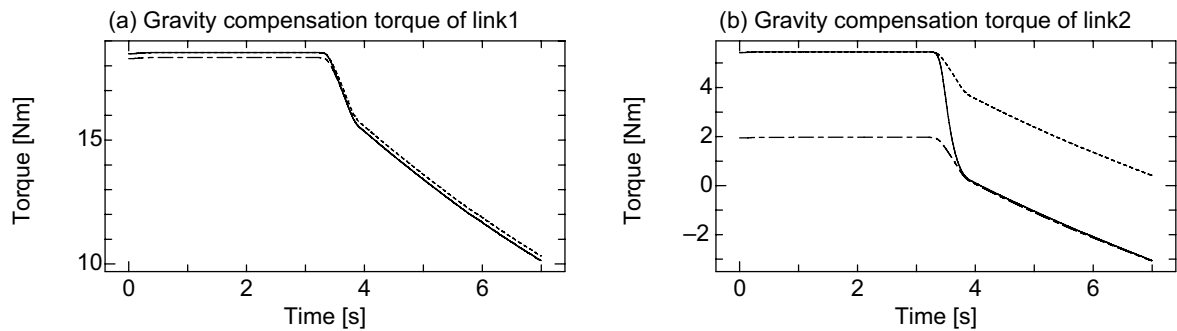
Figure 3. Estimation of gravity term

Fig. 4 shows the simulation results and the experimental results where the external force f is $(-88.9, 6.1)$ [N]. The dotted lines show the simulation results and the bold lines show the experimental results. As in Fig. 4, the experimental results and theoretical response are almost the same and thereby shows that the exact forcefree control can be achieved in practice. The result shows that the forcefree control was achieved with an actual industrial robot arm.

Figure 4. Simulation and experimental results of forcefree control

3. Expansion of Forcefree Control

3.1 Forcefree Control with Independent Compensation

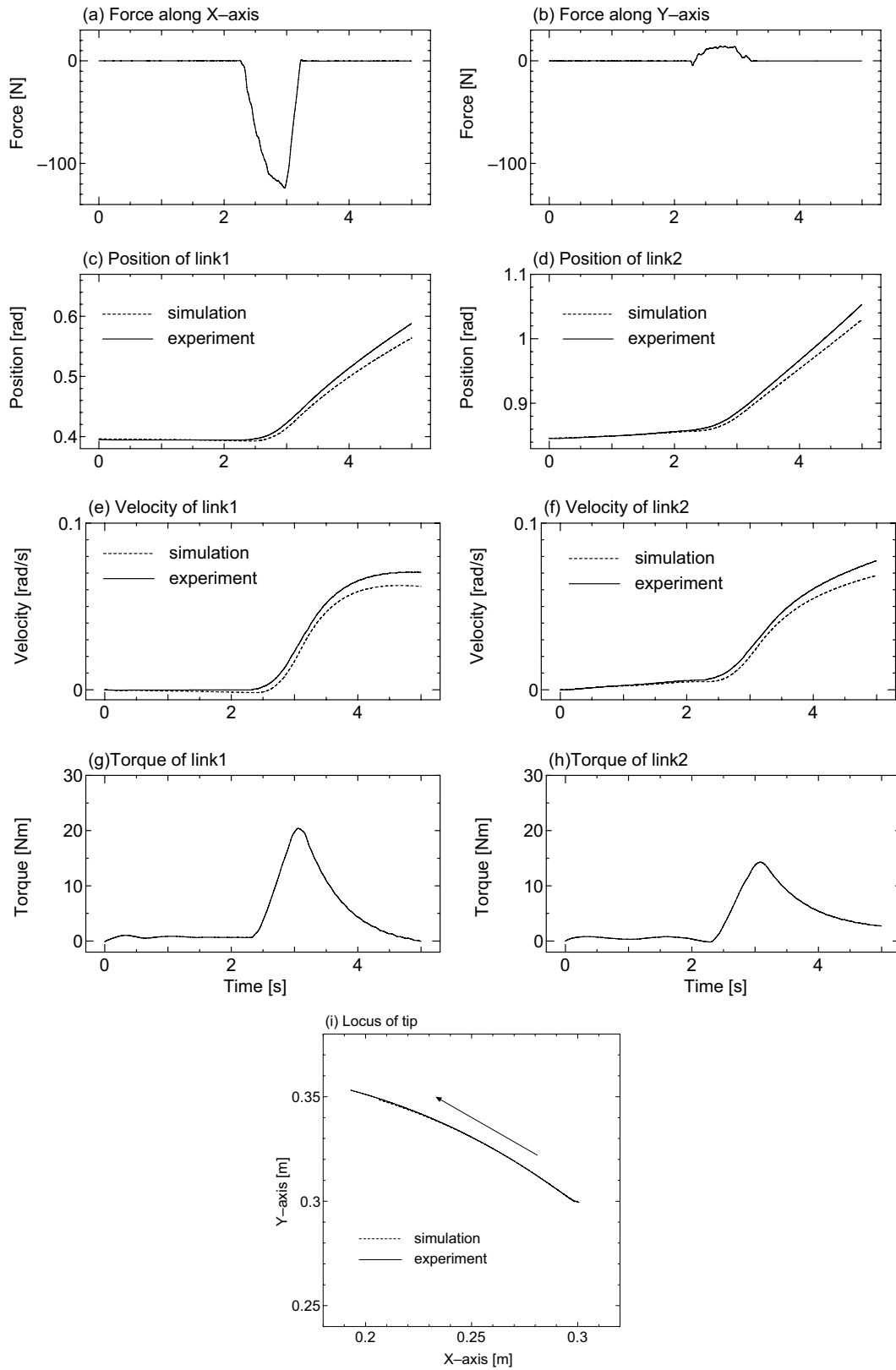The forcefree control is achieved in the non-friction and non-gravity condition. Nevertheless, in some operation of industrial robot arms, friction and/or gravity of the robot arm is useful. Moreover, a large force is required in order to achieve flexible motion of huge robot arms because of their huge inertia, even if the forcefree control is applied. The forcefree control is extended to realize flexible motion emulating the operational circumstances of arbitrary inertia, arbitrary friction and arbitrary gravity through independent compensation of inertia, friction and gravity.

3.1.1 Derivation of Forcefree Control with Independent Compensation

Forcefree control with independent compensation means that the influences of inertia, friction and gravity to the robot arm motion can be assigned arbitrarily. The entire dynamics of an industrial robot arm working on the forcefree control with independent compensation is described by

$$H(q)\ddot{q} + h(q,\dot{q}) = C_f \tau_f - C_d \tau_d - C_g \tau_g \tag{16}$$

where C_f, C_d and C_g are the coefficients of the inertia, friction and gravity terms, respectively. They can be tuned to adjust the effect of the inertia, friction and gravity, independently. For instance, $C_f = E$, $C_d = 0$ and $C_g = 0$, corresponds to the forcefree control and $C_f \to \infty$, $C_d = C_g = 0$ corresponds to the perfect compensation of the inertia, friction and gravity.

The block diagram of the forcefree control with independent compensation is shown in Fig. 5. The inputs of joint angle q_d for the forcefree control with independent compensation is obtained by substituting (14) for (5) and by solving for q_d as

$$q_d = K_p^{-1}\left(K_v^{-1}K_\tau^{-1}\left(C_f \tau_f - C_d \tau_d - C_g \tau_g\right) + \dot{q}\right) + q. \tag{17}$$

Finally, the control law of the forcefree control with independent compensation is obtained by substituting (8) for (17) as

$$q_d = K_p^{-1}\left(K_v^{-1}K_\tau^{-1}\left(-C_f \tau_s + \left(C_f - C_d\right)\tau_d + \left(C_f - C_g\right)\tau_g\right) + \dot{q}\right) + q \tag{18}$$
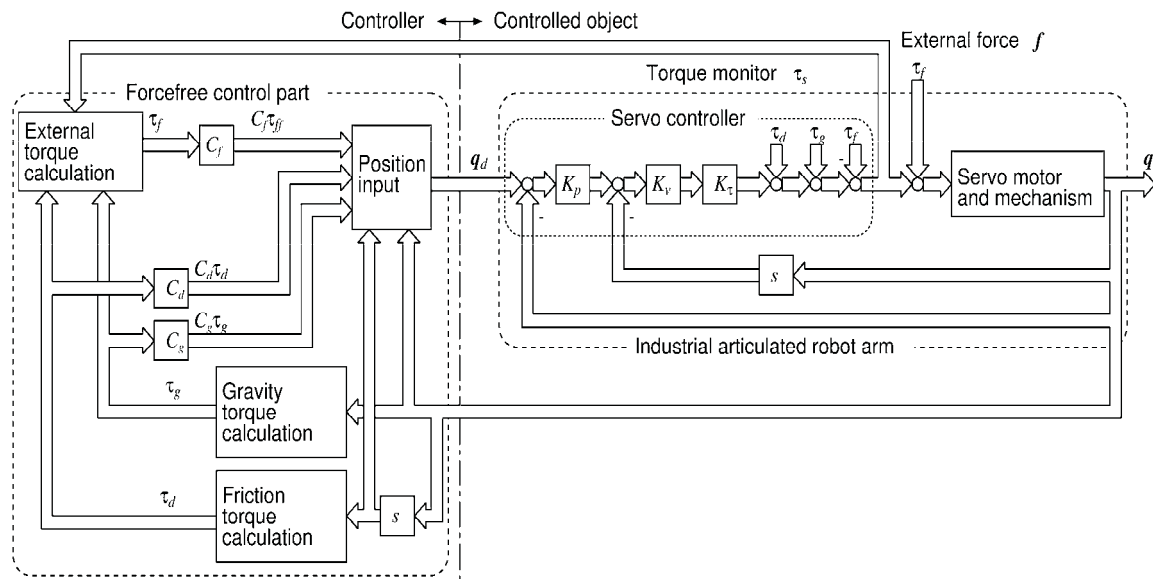
Figure 5. Block diagram of forcefree control with independent compensation

3.1.2 Verification

Step input of 10[N] to X-axis direction was applied to the tip of the robot arm. A force sensor was used to measure the value of external force. The initial end-effector position of the robot arm was at (0.3, 0.3)[m].

Experimental results of the forcefree control with independent compensation are shown in Fig. 6, where the coefficients of compensation are $c^f = 2E$, $C^d = E$, $C^g = 0$. In Fig. 6, the dotted lines show the theoretical responses obtained through simulation and the bold lines show the experimental results.

As in Fig. 6, the experimental results and theoretical response are almost the same and thereby shows that the exact forcefree control with independent compensation can be achieved in practice. The result shows that the forcefree control with independent compensation was realized with an actual industrial robot arm.
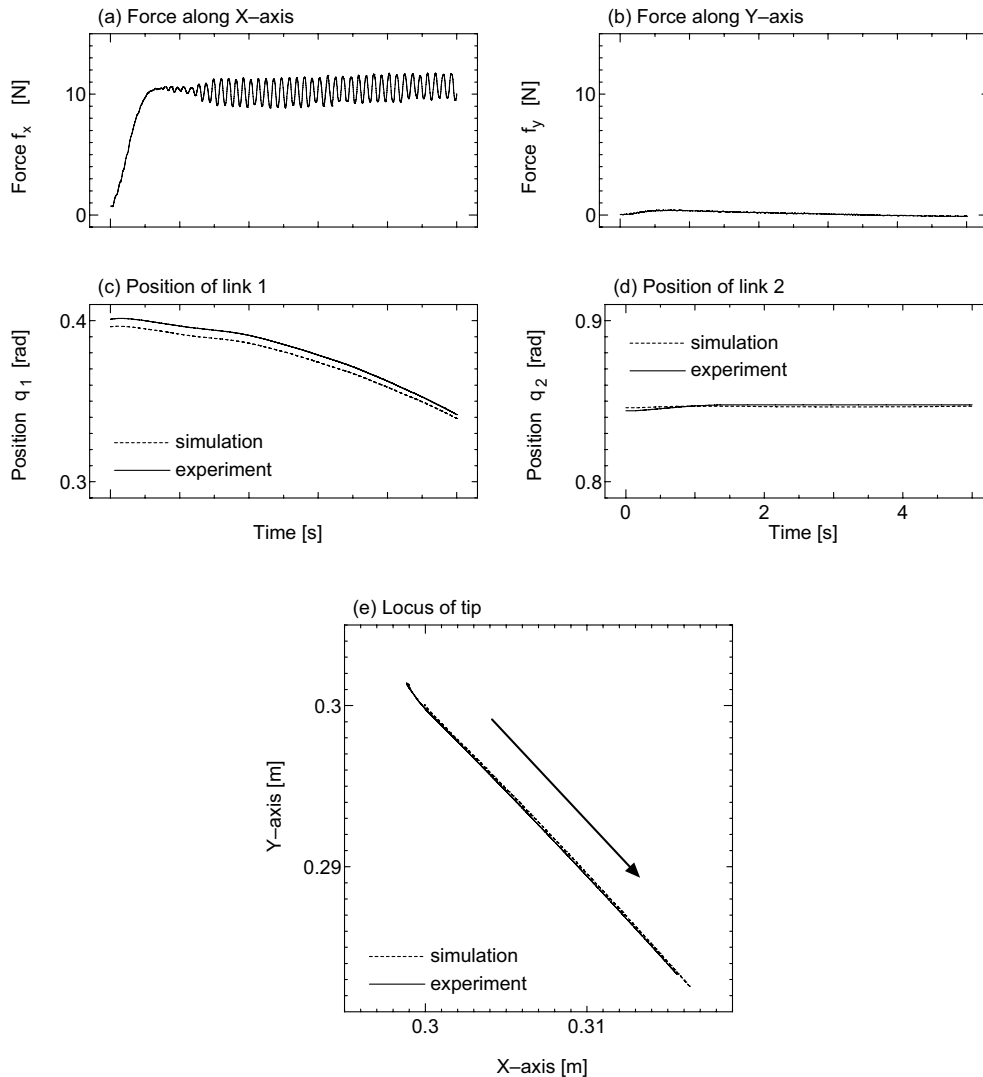
Figure 6. Simulation and experimental results of forcefree control with independent compensation

3.2 Forcefree Control with Assigned Locus

3.2.1 Necessity of Forcefree Control with Assigned Locus

By use of the forcefree control, the robot arm moves according to the external force. The direction of the motion depends on the direction of the external force. When the motion of the robot arm was restricted, the original forcefree control can not be applied. In such a case, the flexible motion with assigned locus is required. In this section, the forcefree control with assigned locus is introduced. The forcefree control with assigned locus makes the tip of the robot arm to follow the assigned locus, and the tip velocity depends on the external force.

3.2.2 Derivation of Forcefree Control with Assigned Locus

The forcefree control with assigned locus is based on mass point type forcefree control. Mass point type forcefree control is constructed from a mass point which is assumed to be the tip of the robot arm. Therefore, the motion of the mass point and the tip of the robot arm are the same. The mass point moves according to the velocity v caused by the external force f, when an external force f is applied on the mass point.

The direction of the motion is the same as the external force f. Besides, the absolute value of the velocity v depends on the external force f. Therefore, the mass point could not follow the assigned locus. In order to follow the assigned locus, the direction of the generated velocity v has to be changed to the tangential direction of the assigned locus. By continuing the above processes, the direction of the velocity v of the mass point is always the same as the tangential direction on the assigned locus. Hence, the tip of the robot arm follows the assigned locus with the velocity which is determined by the external force f.

Fig. 7 shows the block diagram of the forcefree control with assigned locus. Under the non-gravity condition, the equation of the motion of the mass point by the external force f

$$m\ddot{r} + d\dot{r} = f \tag{19}$$

where r is the position of the mass point, m is the mass of the mass point and d is the friction coefficient.

In order to realize the flexible motion with assigned locus according to the external force f, the mass point obeys the equation (19) only for the component in (x, y, z) which gives the maximal amplitude of the external force f.
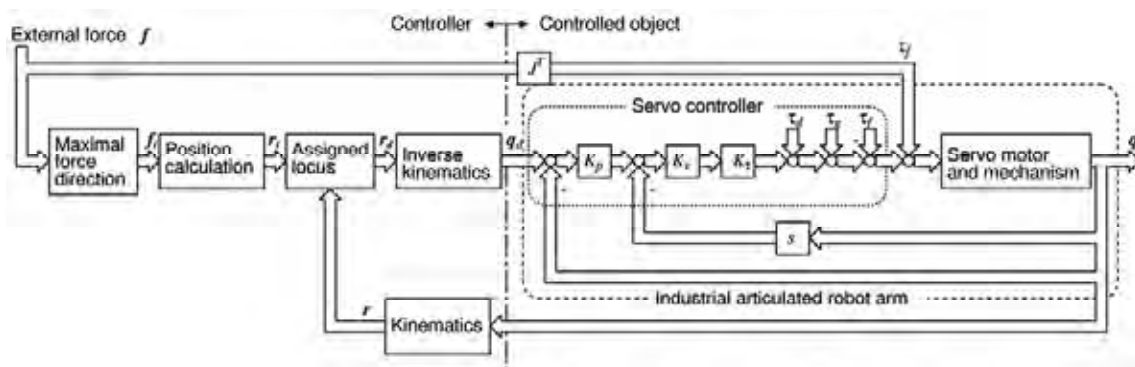

Figure 7. Block diagram of the forcefree control with assigned locus

The other components of the position of the mass point are determined by the assigned locus $h(r_d) = 0$. As a result, the position of the tip of the robot arm r_d under the forcefree control with assigned locus is determined by

$$
\begin{cases}
m\ddot{r}_d^i + d\dot{r}_d^i = f^i, \quad i = \text{argmax}_{j=x,y,z}\left|f^j\right|. \\
h(r_d) = 0
\end{cases}
\tag{20}
$$

The algorithm of forcefree control can be described as follows.

1. The external force f is measured by a force sensor.
2. The component having the maximum amplitude of the external force $i = \text{argmax}_{j=x,y,z}\left|f^j\right|$ is determined.
3. The ith component of the position of the tip of the robot arm r_d^i is determined by using the equation of motion.
4. The other components of the position of the tip of robot arm is determined by the assigned locus $h(r_d) = 0$.
5. The input of the servo controller q_d is calculated from the tip position r_d by using inverse kinematics.

3.2.3 Verification

Verification of the forcefree control with assigned locus was carried out by simulation and experiment. Besides, simulation and experiment were carried out under non-friction conditions which mean the coefficient of friction d was zero.

Simulation and experimental results are shown in Fig. 8. In Fig. 8, (a) and (b) illustrate the components of the external force f along the direction of X-axis and Y-axis, respectively, (c) and (d) show the joint trajectories of link1 and link2, respectively, (e) and (f) show the velocity v of X-axis and Y-axis, respectively, and (g) shows the locus of the tip of the robot arm. In Fig. 8, the dotted line denotes the simulation result and the bold line denotes the experimental results. In Fig. 8(g), dash line shows the assigned locus. It can be verified that simulation and experimental results are comparable, and both results have realized the exact assigned locus. This phenomenon illustrates the realization of the proposed forcefree control with assigned locus.
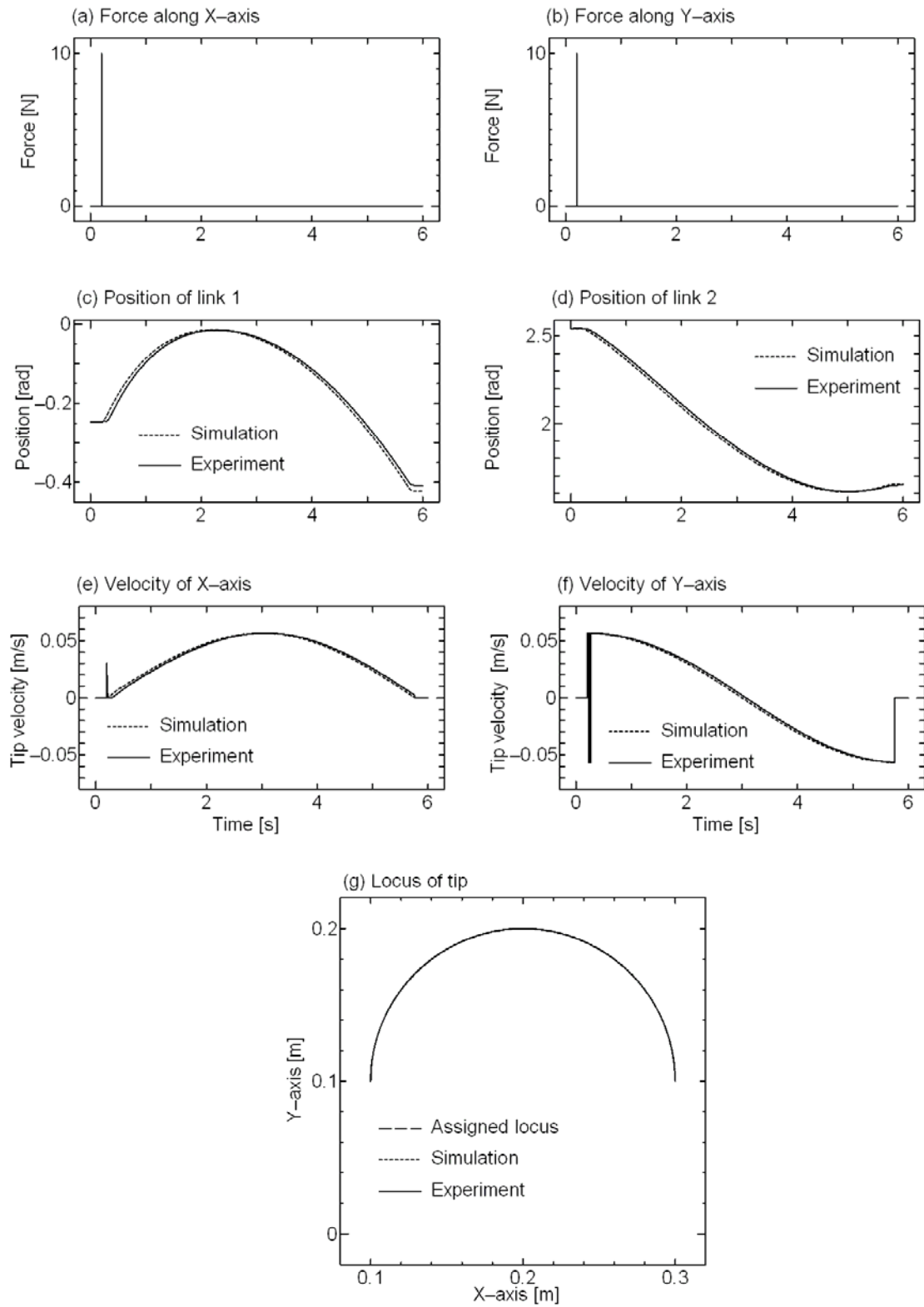
Figure 8. Simulation and experimental results of forcefree control with assigned locus

3.3 Position Information Based Forcefree Control

3.3.1 Necessity of Position Information Based Forcefree Control

The above explained forcefree control requires a force sensor or a torque monitor for the detection of the external force. In order to apply more general industrial robot arms, the forcefree control is extended, where the external force is estimated only by the position information.

In the servo controller of the industrial robot arms, a PI controller is used for the velocity loop and the torque disturbance is compensated by the integral action of the PI controller so that the robot arm is not moved by the external force. On the other hand, the torque information required for the forcefree control must be estimated by the change of position caused by the external force. Hence, the P controller must be used for the velocity loop. By changing the velocity controller from PI controller to P controller, the compensation of the torque disturbance vanishes and the torque information can be estimated from the change of position caused by the external force.

3.3.2 Derivation of Position Information Based Forcefree Control

Fig. 9 shows the block diagram of the position information based forcefree control. As in Fig. 9, the velocity loop in the servo controller is the P controller and the friction compensation τ_d, the gravity compensation τ_g and the external torque compensation τ_f are not included in the servo controller compared with the servo controller in Fig. 1.
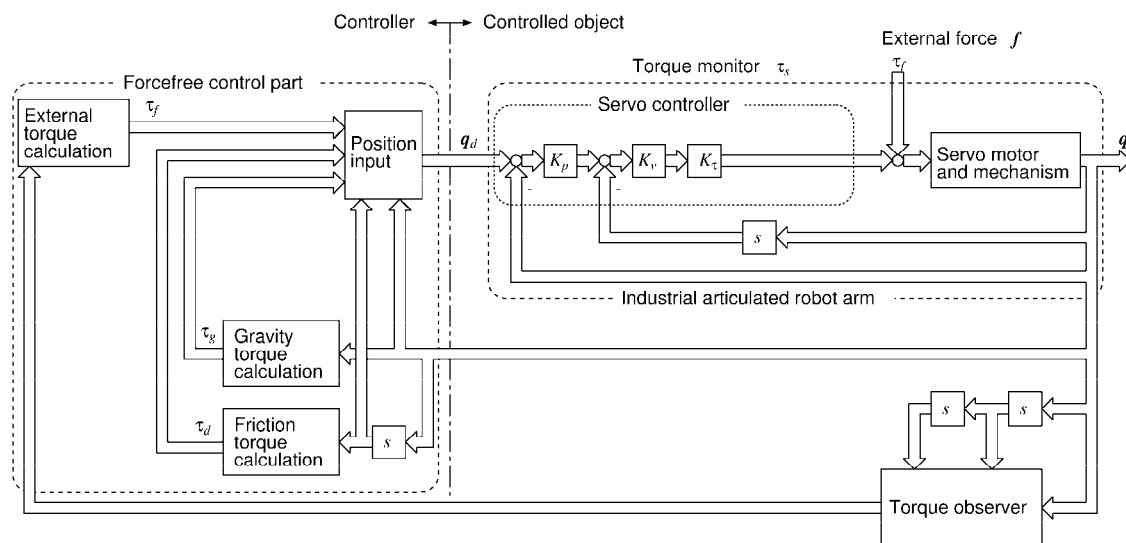

Figure 9. Block diagram of position information based forcefree control

The control input q_d is calculated by

$$q_d = K_p^{-1}\left(K_v^{-1}K_\tau^{-1}\left(\tau_f + \tau_d + \tau_g\right) + \dot{q}\right) + q \tag{21}$$

where the friction compensation torque τ_d and the gravity compensation torque τ_g are calculated by (8) and (12), respectively.

The external force compensation torque τ_f can be estimated by using the position information q and the velocity information $\dot{q}$ as

$$\tau_f = H(q)\ddot{q} + h(q,\dot{q}). \tag{22}$$

3.3.3 Verification

Verification of the position information based forcefree control was carried out by a simulation study and an experiment. Simulation and experimental results are shown in Fig. 10. In Fig. 10, (a) and (b) illustrate the components of the external force f in the directions of X-axis and Y-axis, respectively, (c) and (d) show the joint trajectories of link1 and link2, respectively, (e) and (f) show the velocity v of X-axis and Y-axis, respectively, (g) and (h) show the estimated torque by using the torque observer and the torque calculated by the force sensor, respectively, and (i) shows the locus of the tip of the robot arm. In Fig. 10, the dotted line denotes the simulation result and the bold line shows the experimental result. In (g) and (h), the estimated torque coincides with the actual torque. It can be verified that the simulation and experimental results are comparable, and the forcefree control can be achieved only by the usage of the position information.
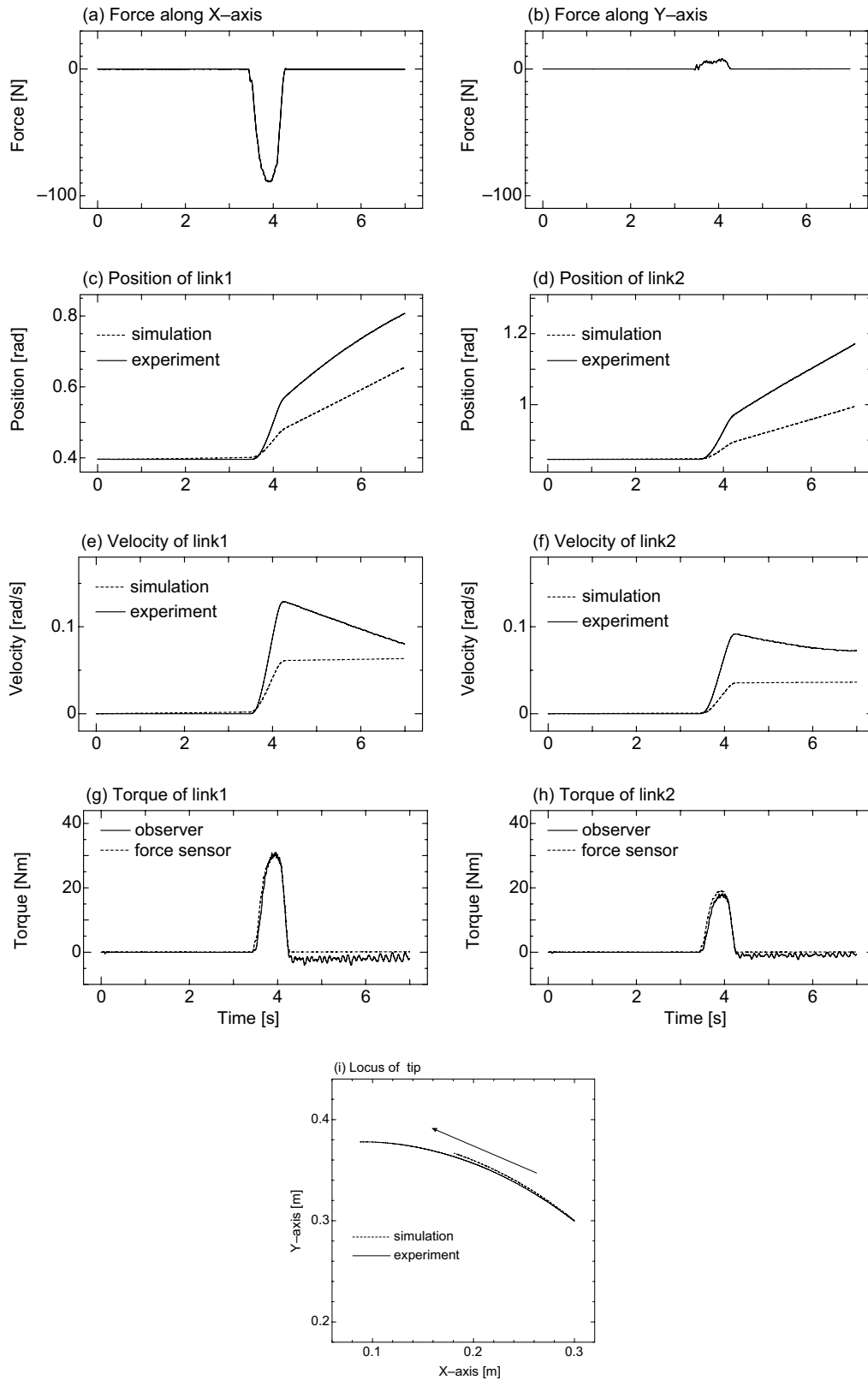
Figure 10. Simulation and experimental results of position information based forcefree control

4. Comparison between Forcefree Control and Force Control

4.1 Comparison between Forcefree Control with Independent Control and Inpedance Control

In order to illustrate the feature of the forcefree control, the forccefree control with independent compensation is compared with the impedance control (Hogan 1985; Scivicco & Siciliano, 2000). The impedance control is the typical force control, which enables the contact force between the tip of the robot arm and the object as assigned inertia, friction and stiffness. The impedance characteristics are expressed by

$$M_d \ddot{r} + D_d (\dot{r} - \dot{r}_d) + K_d (r - r_d) = F \tag{23}$$

where F is the assigned force between the tip of the robot arm and the object, M_d, D_d and K_d are the assigned inertia, friction and stiffness, respectively, and r_d, $\dot{r}_d$ are the objective position and the objective velocity in working coordinates, respectively. The dynamics of the robot arm in joint coordinates is expressed by

$$H(q)\ddot{q} + h(q,\dot{q}) + D\dot{q} + N_\mu \mathrm{sgn}(\dot{q}) + g(q) = \tau + J^T F \tag{24}$$

and the dynamics in working coordinates is expressed by

$$H_r(q)\ddot{r} + h_r(q,\dot{q}) + D_r \dot{r} + N_{\mu r}\mathrm{sgn}(\dot{r}) + g_r(q) = (J^T)^{-1}\tau + F. \tag{25}$$

By substituting (23) for (25), the torque input for the impedance control is obtained by

$$
\begin{aligned}
\tau = J^T \big[& H_r(q)M_d^{-1}\{-D_d(\dot{r}-\dot{r}_d) - K_d(r-r_d)\} \\
& + h_r(q,\dot{q}) + (H_r(q)M_d^{-1} - I)F + D_r \dot{r} + N_{\mu r}\mathrm{sgn}(\dot{r}) + g_r(q) \big]
\end{aligned}
\tag{26}
$$

The torque input of the forcefree control with independent compensation is derived as the same format of the impedance control. The dynamics of the forcefree control with independent compensation in joint coordinates (16) is transformed into working coordinates as

$$H_r(q)\ddot{r} + h_r(q,\dot{q}) = C^f F - C^d (D_r \dot{r} + N_{\mu r}\mathrm{sgn}(\dot{r})) - C^g g_r(q). \tag{27}$$

By substituting (27) for (24), the torque input for the forcefree control with independent compensation is obtained by

$$\tau = J^T\left[\left(C^f - I\right)F + \left(I - C^d\right)\left(D_r\dot{r} + N_{\mu r}\text{sgn}(\dot{r})\right) + \left(I - C^g\right)g_r(q)\right] \tag{28}$$

By comparing the torque input of the impedance control (26) and that of the forcefree control with independent compensation (28), the following relationship is fulfilled.

$$H_r(q)M_d^{-1}\left\{-D_d(\dot{r} - \dot{r}_d) - K_d(r - r_d)\right\} + h_r(q,\dot{q}) = 0 \tag{29}$$

$$M_d = \left(C^f\right)^{-1}H_r(q) \tag{30}$$

$$D_d = O \tag{31}$$

$$K_d = O \tag{32}$$

$$h_r(q,\dot{q}) = 0 \tag{33}$$

$$C^d = C^g = O \tag{34}$$

The difference between the forcefree control with independent compensation and the impedance control is as follows;

1. In impedance control, the objective trajectory r_d is defined whereas no objective trajectory exists in the forcefree control with independent compensation.
2. The forcefree control with independent compensation can tune the effects of the friction and the gravity whereas the impedance compensation do perfect compensation.

As a result, the forcefree control with independent compensation is completely different control strategy from the impedance control.

4.2 Comparison between Forcefree Control with Assigned Locus and Impedance Control

In the case of forcefree control with assigned locus and the impedance control, the tip of the robot arm is related to joint motion, but actually, joint coordinate is not necessary to consider because generalized coordinates are defined in working coordinate.

In case of impedance control, inertia is compensated by adjusting a compliance matrix. On the contrary to the forcefree control with assigned locus, inertia can be adjusted through independent setting of the value of the mass point.

Moreover, in case of impedance control, identification of coefficients of viscous friction and calculation of gravity term must be done a priori for the friction compensation and the gravity compensation. On the contrary to the forcefree control with assigned locus, these compensations are not required because a dynamic equation of the mass point is defined in non-friction and non-gravity space.

Although forcefree control with assigned locus is capable of following the assigned locus, impedance control is not thus capable. Therefore, forcefree control with assigned locus has the many advantages over impedance control counterparts. Other general force control methods have same problems as impedance control.

The impedance control is expressed by

$$M_d \ddot{r} + D_d \left(\dot{r} - \dot{r}_d \right) + K_d \left(r - r_d \right) = f \tag{35}$$

where f is the assigned force between the tip of the robot arm and the object, M_d, D_d and K_d are the assigned inertia, friction and stiffness, and r_d, $\dot{r}_d$ are the objective position and the objective velocity in working coordinates, respectively.

The mass point type forcefree control is expressed by

$$m\ddot{r} = f \tag{36}$$

where m is the assigned mass of the mass point. By comparing (35) and (36), the mass point type forcefree control is achieved by

$$M_d = m \tag{37}$$

$$D_d = 0 \tag{38}$$

$$K_d = 0 \tag{39}$$

After achieving the mass point type forcefree control by the impedance control, the forcefree control with assigned locus is accomplished in exactly the same way explained in section 3.2.2.

Table 1 summarized the comparison of the forcefree control with independent compensation, the forcefree control with assigned locus and the impedance control.

	Forcefree control with independent compensation	Forcefree control with assigned locus	Impedance control
Objective	Free motion by external force	Free motion by external force with assigned locus	Desirable mechanical impedance
Model	Dynamics of industrial articulated robot arm	Dynamics of industrial articulated robot arm	Mechanical impedance between tip arm and object
Motion	Passive motion against external force	Passive motion against external force	Active motion to realize assigned force
Rigidity	Zero	Zero	Setting by virtual spring
Inertia	Setting by coefficient of inertia	Setting by virtual mass	Setting by virtual mass
Friction	Setting by coefficient of friction	Setting by virtual friction	Setting by virtual damper
Gravity	Setting by the coefficient of gravity	Zero	Compensation
Target	Industrial articulated robot arm	Industrial articulated robot arm	Articulated robot arm
Coordinates	Joint coordinates	Cartesian coordinates	Cartesian coordinates
Locus following	Impossible	Possible	Impossible
Command	Position	Position	Torque, Position

Table 1. Comparison among forcefree control with independent compensation, forcefree control with assigned locus and impedance control

5. Applications of Forcefree Control

5.1 Pull-Out Work

Pull-out work means that the workpiece is pulled out by the push-rod, where the workpiece is held by the robot arm, and it is usually used in aluminum casting in industry. The operation follows the sequence, a) the hand of the robot arm grasps the workpiece, b) the workpiece is pushed out by the push-rod, and c) the workpiece is released by the force from the push-rod. The motion of the robot arm requires flexibility in order to follow the pushed workpiece.

Experimental results of pull-out work by the force-free control is shown in Fig. 11. Fig. 11(a) and (b) show the torque monitor outputs of link 1 and link 2 caused by the push-rod, respectively, (c) and (d) show the position of link 1 and link 2, respectively, and Fig. 11(e) shows the locus of the tip of the robot arm.

It guarantees the realization of pull-out work with industrial articulated robot arm based on the forcefree control.

5.2 Direct Teaching

In general, the industrial robot arms carry out operations based on teaching-playback method. The teaching-playback method is separated into two parts, i.e., teaching part and playback part. In the teaching part, the robot arm is taught the data of operational positions and velocities. In the playback part, the robot arm carries out the operation according to the taught data.

The teaching of industrial articulated robot arms is categorized into two methods, i.e., on-line teaching and off-line teaching. Off-line teaching requires another space for teaching. Therefore, on-line teaching is used for industrial articulated robot arms. On-line teaching is also categorized into remote teaching and direct teaching. Here, the remote teaching means that the teaching is carried out by use of a teach-pendant, i.e., a special equipment for teaching, and direct teaching means that the robot arm is moved by human direct force.

Usually, the teaching of industrial articulated robot arms is carried out by remote teaching. Remote teaching by use of teach-pendant, however, requires human skill because there exists a difference between operator coordinates and robot arm coordinates. Besides, the operation method of teach-pendant is not unique, thus depends on the robot arm manufacturer.

Direct teaching is useful for industrial articulated robot arms against remote teaching. The process of direct teaching is as follows; 1) the operator grasps the

tip of the robot arm, 2) the operator brings the tip of the robot arm to the teaching points by his hands, directly, and 3) teaching points are stored in memory. Operational velocities between teaching points are set after the position teaching process. In other words, anyone can easily carry out teaching. In direct teaching, operational positions of the industrial articulated robot arm are taught by human hands directly. The proposed forcefree control can be applied to realize the direct teaching of the industrial articulated robot arm. Forcefree control can realize non-gravity and non-friction motion of the industrial articulated robot arm under the given external force. In other words, an industrial articulated robot arm is actuated by human hands, directly. Here, position control of the tip of the robot arm is the important factor in direct teaching. Position control of the tip of the robot arm is carried out by the operator in direct teaching.

Direct-teaching for teaching-playback type robot arms is an application of the forcefree control with independent compensation, where the robot arm is manually moved by the human operator's hand. Usually, teaching of industrial articulated robot arms is carried out by using operational equipment and smooth teaching can be achieved if direct-teaching is realized.

Fig. 12 shows the experimental result of direct-teaching where the compensation coefficients are $C_f = 0.5E$, $C_d = E$, $C_g = 0$. As shown in Fig. 12, teaching was successfully done by the direct use of human hand. The forcefree control with independent compensation does not use the force sensors and any part of the robot arm can be used for motion of the robot arm.
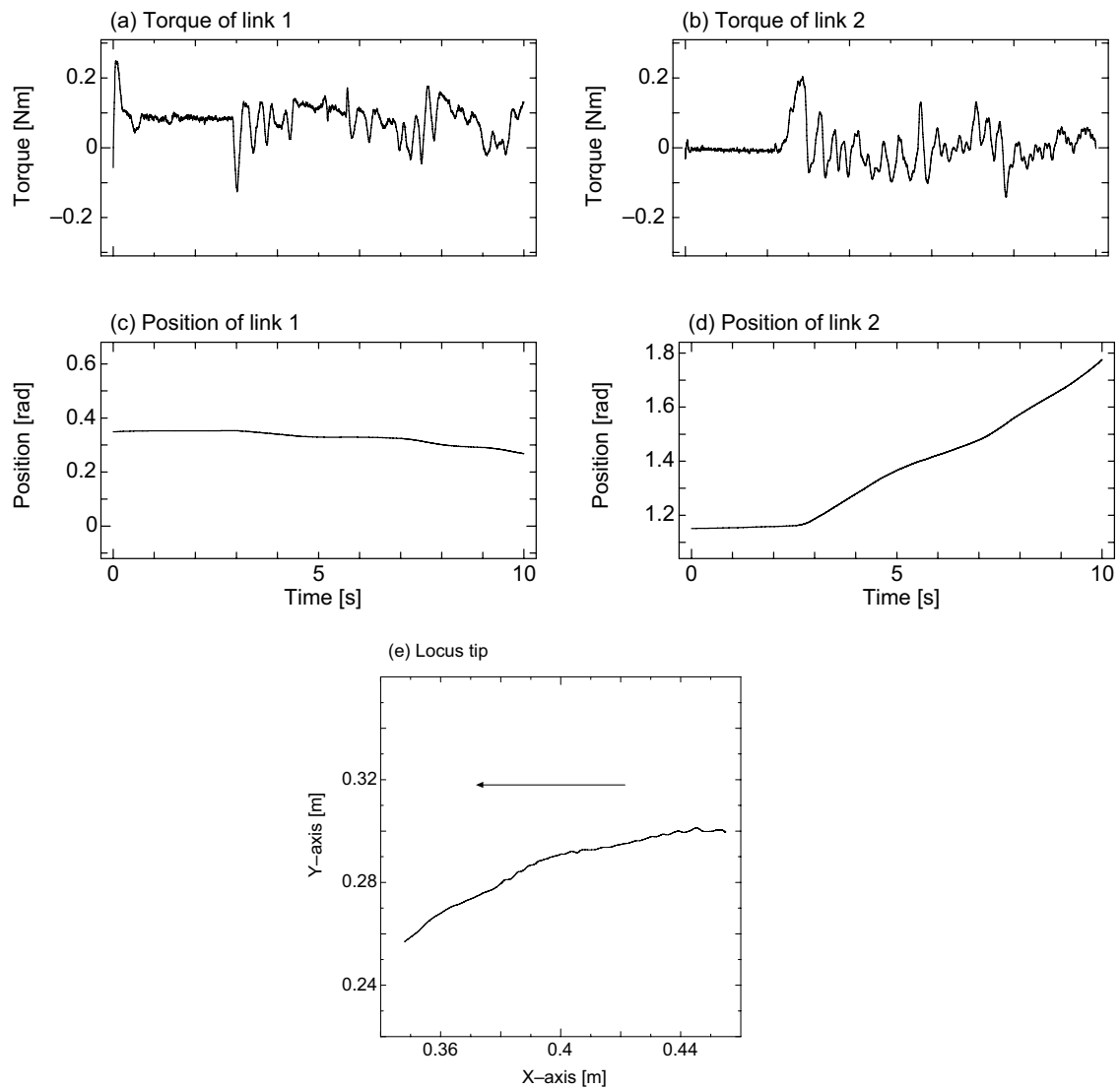
Figure 11. Experimental result of pull-out work by using the forcefree control with independent compensation ($C_f = 0.2E$, $C_d = C_g = 0$)
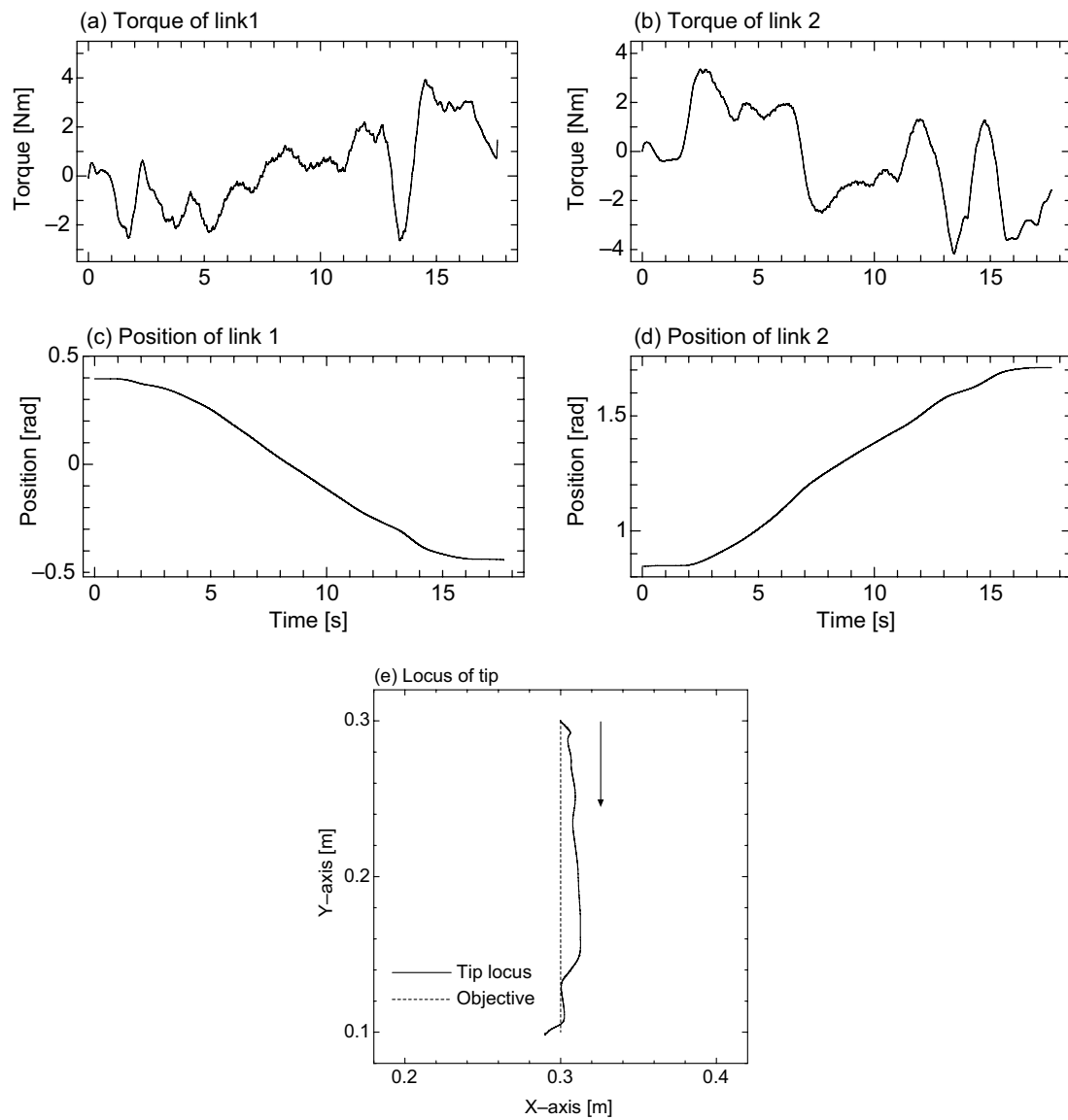
Figure 12. Experimental result of direct teaching by using the forcefree control with independent compensation ($C_f = 0.5E$, $C_d = E$, $C_g = 0$)

5.3 Rehabilitation Robot

The forcefree control with independent compensation uses the torque monitor in order to detect the external force. Hence, each joint can be monitored for unexpected torque deviation from the desired torque profile as a result of unplanned circumstances such as accidental contact with an object or human being. As a result, the forcefree control with independent compensation can also improve the safety of work with human operator. To utilize this feature, the forcefree control with independent compensation is applied to rehabilitation robots.

The forcefree control with independent compensation is applied to the control of a meal assistance orthosis for disabled persons both of direct-teaching of plate position and mouth position and safety operation against unexpected human motion.

If the forcefree control with independent compensation is installed in such systems, the safety will be improved because when the unexpected contact between the operator and the robot occurs, the escape motion of the robot arm can be invoked by the forcefree control method.

6. Conclusions

The proposed forcefree control realizes the passive motion of the robot arm according to the external force. Moreover, the forcefree control is extended to the forcefree control with independent compensation, the forcefree control with assigned locus and the position information based forcefree control. Experiments on an actual industrial robot arm were successfully carried out by the proposed methods. The comparison between the forcefree control and other force control is expressed and the features of the forcefree control are clarified. The proposed method requires no change in hardware of the robot arm and therefore is easily acceptable to many industrial applications.

7. References

Ciro, N., R. Koeppe, and G. Hirzinger, (2000). A Systematic Design Procedure of Force Controllers for Industrial Robots, IEEE/ASME Trans. Mechatronics, 5-21, 122-133.

Fu, K. S., R. C. Sonzalez and C. S. G. Lee, (1987). Robotics Control, Sensing, Vision, and Intelligence, pp. 82-144, McGraw-Hill, Inc., Singapore.

Hogan, N. (1985). Impedance Control; An Approach to Manipulation: Part I-III, Trans. of the ASME Journal of Dynamic System, Measurement, and Control, 107, 1-24.

Kyura, N., (1996). The Development of a Controller for Mechatronics Equipment, IEEE Trans. on Industrial Electronics, 43, 30-37.

Mason, M. T. (1981). Compliance and Force Control for Computer Controlled Manipulators, IEEE Trans. on Systems, Man, and Cybernetics, 11, 418-432.

Michael, B., M. H. John, L. J. Timothy, L. P. Tomas and T. M. Matthew, (1982). Robot Motion: Planning and Control, The MIT Press, Cambridge.

Nakamura, M., S. Goto, N. Kyura, (2004). Mechatronic Servo System Control, Springer-Verlag Berlin Heidelberg.

Sciavicco, L. and B. Siciliano, (2000). Modelling and Control of Robot Manipulators, pp. 271-280, Springer, London.

Predictive Force Control of Robot Manipulators in Nonrigid Environments

L.F. Baptista, J.M.C. Sousa and J.M.G. Sa da Costa

1. Introduction

The application of robot manipulators in industry is in general related to tasks such as manipulation or painting that requires only position control of the arm. Nonetheless, there are other robotic tasks like pushing, polishing and grinding that require interaction between the manipulator and a contact surface or environment. This fact leads to the desire of controlling the interaction between the robot and the environment. Although a lot of different control schemes has been proposed in the literature, as surveyed by (Zeng & Hemami, 1997 ; De Schutter et al., 1997), the major force control approaches can be classified as hybrid control (Raibert & Craig, 1981) or impedance control (Hogan, 1985). The hybrid control separates a robotic force task into two subspaces: a force controlled subspace and a position controlled subspace. Two independent controllers are then designed for each subspace. In contrast, impedance control does not attempt to control force explicitly but rather to control the relationship between force and position of the end-effector in contact with the environment. Furthermore, when the environment is rigid with known characteristics it is possible to plan a virtual trajectory, such that a desired force profile is obtained (Singh & Popa, 1995). However, the same does not hold in the presence of nonrigid environments, which disables a reliable application of the classical impedance controller. This problem has motivated the development and design of more sophisticated force control methodologies which usually take into consideration the dynamics of the environment. In (Love & Book, 1995) it is shown that the performance of an impedance controlled manipulator increases when the desired impedance includes some modeling of the environment. Another possible solution to tackle this problem is to use a model-based control scheme like predictive control, which incorporates the manipulator and environment models in a force optimization-based strategy (Wada et al., 1993). Recently, a force control strategy for robotic manipulators in the presence of nonrigid environments combining impedance control and a model predictive control (MPC) algorithm in a force control scheme has been proposed (Baptista et al., 2000b). In this force control methodology, the predictive

controller generate the position and velocity references in the constrained direction, to obtain a desired force profile acting on the environment. The main advantage of this control strategy is to provide an easy inclusion of the environment model in the controller design and thus to improve the global performance of the control system.

Usually, impedance and environmental models are linear, mainly because the solution of an unconstrained optimization procedure can be analytically obtained with moderate computational burden. However, a nonrigid environment has in general a nonlinear behavior, and a nonlinear model for the contact surface must be considered. Therefore, in this paper the linear spring/damper parallel combination, often used as a model of the environment, is replaced by a nonlinear one, where the damping effect depends on the penetration depth (Marhefka & Orin, 1996). Unfortunately, when a nonlinear model of the environment is used, the resulting optimization problem to be solved in the MPC scheme is nonconvex. This problem can be solved using discrete search techniques, such as the branch-and-bound algorithm (Sousa et al., 1997). This discretization, however, introduces a tradeoff between the number of discrete actions and the performance. Moreover, the discrete approximation can introduce oscillations around non-varying references, usually know as the chattering effect, and slow step responses depending on the selected set of discrete solutions. These effects are highly undesirable, especially in force control applications. A possible solution to this problem is a fuzzy scaling machine, which is proposed in this paper. Fuzzy logic has been used in several applications in robotics. In the specific field of robot force control, some relevant references, such as (Liu, 1995 ; Corbet et al., 1996 ; Lin & Huang, 1997), can be mentioned. However, these papers use fuzzy logic in the classic low level form, while in this paper fuzzy logic is applied in a higher level. Here, the fuzzy scaling machine alleviates the effects due to the discretization of the nonconvex optimization problem to be solved in the model predictive algorithm, which derives the virtual reference for the impedance controller considering a nonlinear environment. The fuzzy scaling machine proposed in this paper uses an adaptive set of discrete alternatives, based on the fulfillment of fuzzy criteria applied to force control. This approach has been used in predictive control (Sousa & Setnes, 1999), and is generalized here for model predictive algorithms. The adaptation is performed by a scaling factor multiplied by a set of alternatives. By using this approach, the number of alternatives is kept low, while performance is increased. Hence, the problems introduced by the discretization of the control actions are highly diminished.

For the purpose of performance analysis, the proposed predictive force control strategy with fuzzy scaling is compared with the impedance controller with force tracking by simulation with a two-degree-of-freedom (2-DOF) manipulator, considering a nonlinear model of the environment. The robustness of the predictive control scheme is tested considering unmodeled friction and Corio-

lis effects, as well as geometric and stiffness uncertainties on the contact surface.

The implementation and validation of advanced control algorithms, like the one presented above, require a flexible structure in terms of hardware and software. However, one of the major difficulties in testing advanced force/position control algorithms relies in the lack of available commercial open robot controllers. In fact, industrial robots are equipped with digital controllers having fixed control laws, generally of PID type with no possibility of modifying the control algorithms to improve their performance. Generally, robot controllers are programmed with specific languages with fixed programmed commands having internally defined path planners, trajectory interpolators and filters, among other functions. Moreover, in general those controllers only deal with position and velocity control, which is insufficient when it is necessary to obtain an accurate force/position tracking performance (Baptista et al., 2001b). Considering these difficulties, in the last years several open control architectures for robotic applications have been proposed. Generally, these solutions rely on digital signal processor techniques (Mandal & Payandeh, 1995 ; Jaritz & Spong, 1996) or in expensive VME hardware running under the VxWorks operating system (Kieffer & Yu, 1999). This fact has motivated the development of an open PC-based software kernel for management, supervision and control. The real-time software tool for the experimentation of the algorithms proposed in this paper was developed considering requirements such as low cost, high flexibility and possibility of incorporating new hardware devices and software tools (Baptista, 2000a).

This article is organized as follows. Section 2 summarizes the manipulator and the environment dynamic models. The impedance controller with force tracking is presented in section 3. Section 4 presents the model predictive algorithm with fuzzy scaling applied to force control. The simulation results for a 2-DOF robot manipulator are discussed in section 5. The experimental setup and the force control algorithms implemented in real-time are presented in section 6. The experimental results with a 2-DOF planar robot manipulator are presented in section 7. Finally, some conclusions are drawn in section 8.

2. Manipulator and environment modeling

Consider an n-link rigid-link manipulator constrained by contact with the environment, as shown in fig.1. The complete dynamic model is described by (Siciliano & Villani, 2000)

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + d(\dot{q}) = \tau - \tau_e \tag{1}$$

where $q, \dot{q}, \ddot{q} \in R^{n \times 1}$ correspond to the joint, position, velocity and acceleration vectors, respectively, $M(q) \in R^{n \times n}$ is the symmetric positive definite inertia matrix, $C(q, \dot{q}) \in R^{n \times n}$ is the centripetal and Coriolis matrix, $g(q) \in R^{n \times 1}$ contains the gravitational terms and $d(\dot{q})q \in R^{n \times 1}$ accounts for the frictional terms. The vector $\tau \in R^{n \times 1}$ is the joint input torque vector and $\tau_e \in R^{n \times 1}$ denote the generalized vector of joint torques exerted by the environment on the end-effector. From (1) it is possible to derive the robot dynamic model in the Cartesian space:

$$M_x(x)\ddot{x} + C_x(x, \dot{x})\dot{x} + g_x(x) + d_x(\dot{x}) = f - f_e \tag{2}$$

where x is the n-dimensional vector of the position and orientation of the manipulator's end-effector, $f = J^{-T}(q)\tau \in R^{n \times 1}$ is the robot's driving force, $f_e \in R^{n \times 1}$ is the contact force vector and J represents the Jacobian matrix.

The interaction force vector $f_e = \begin{bmatrix} f_n & f_t \end{bmatrix}^T$ is composed by the normal contact force f_n and the tangential contact forces f_t caused by friction contact between the end-effector and the surface. An accurate modeling of the contact between the manipulator and the environment is usually difficult to obtain due to the complexity of the robot's end-effector interaction with the environment. In this paper, the normal contact force f_n is modeled as a nonlinear spring-damper mechanical system according (Marhefka & Orin, 1996):

$$f_n = ke\delta x + \rho_e(\delta x)\dot{x} \tag{3}$$

where the terms k_e and ρ_e are the environment stiffness and damping coefficients, respectively, $\delta x = x - x_e$ is the penetration depth, where x_e stands for the distance between the surface and the base Cartesian frame. Notice that the damping effect depends non-linearly on the penetration depth δx. The tangential contact force vector f_t due to surface friction is assumed to be given as proposed by (Yao & Tomizuka, 1995):

$$f_t = \mu |f_n| \operatorname{sgn}(\dot{x}_p) \tag{4}$$

where $\dot{x}_p$ is the unconstrained or sliding velocity and μ is the dry friction coefficient between the end-effector and the contact surface.
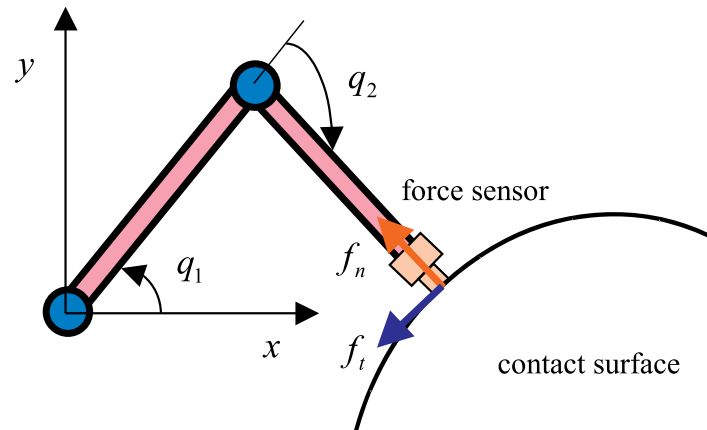
Figure 1. Robot manipulator applying a desired force on the environment.
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001a) with kind permission of Springer Science and Business Media).

3. Impedance control

The impedance controller proposed by (Hogan, 1985) aims at controlling the dynamic relation between the manipulator and the environment. The force exerted by the manipulator on the environment depends on the end-effector position and the correspondent impedance. The impedance of the robot is divided in the following terms: one that is physically intrinsic to the manipulator and the other that is given to the robot by the controller. The impedance control goal is to oblige the manipulator to follow the reference or target impedance. As shown by (Volpe & Khosla, 1995) a good impedance relation is achieved with a linear model of second order. The complete form of a second order type impedance control model, which is valid for free or constrained motion, is given by:

$$M_d \ddot{x} - B_d(\dot{x}_d - \dot{x}) - K_d(x_d - x) = -f_e \qquad (5)$$

where $\dot{x}_d, x_d$ are the desired velocity and position defined in the Cartesian space, respectively, and $\dot{x}, x$ are the end-effector velocity and position in Cartesian space, respectively. The matrices M_d, B_d, K_d are the desired inertia, damping and stiffness for the manipulator. The reference or target end-effector acceleration $u \equiv \ddot{x}$ is then given by:

$$u = M_d^{-1}(B_d \dot{e} + K_d e - f_e) \qquad (6)$$

where $\dot{e} = \dot{x}_d - \dot{x}$, $e = x_d - x$ are the velocity and position errors, respectively. Thus, u can be used as the command signal to an inner position control loop in order to drive the robot accordingly to the desired trajectory.

3.1 Virtual trajectory for force tracking

The major drawback of the impedance control scheme presented above is related to its poor force tracking capability, especially in the presence of nonrigid environments (Baptista et al., 2000b). However, from the conventional impedance control scheme it is possible to obtain a force control scheme in a steady-state contact condition with the surface. Considering the impedance control scheme (6) in the constrained direction, the following holds:

$$u_f = m_d^{-1}(b_d(\dot{x}_v - \dot{x}) + k_d(x_v - x) - f_n) \tag{7}$$

where x_v, $\dot{x}_v$ and u_f are the virtual position, velocity and target acceleration, respectively, while m_d, b_d, k_d are appropriate elements of M_d, B_d, K_d matrices defined in (5) in the constrained direction. The contact force f_n during steady-state contact with the surface is given by:

$$f_n = k_d(x_v - x) \tag{8}$$

Considering for simplicity the environment modeled by a linear spring with stiffness k_e the contact force is given by:

$$f_n = k_e \delta x \tag{9}$$

This leads to the following steady-state position and contact force (Singh & Popa, 1995):

$$x_{ss} = \frac{k_d x_v + k_e x_e}{k_d + k_e} \tag{10}$$

$$f_{n_{ss}} = \frac{k_d k_e}{k_d + k_e}(x_v - x_e) \tag{11}$$

It is possible to apply a desired force f_d on the system while simultaneously achieving the desired impedance by estimating the desired virtual position x_v as:

$$x_v = x_e + f_d \left(\frac{k_e + k_d}{k_e k_d} \right) \tag{12}$$

Moreover, when the environment stiffness is unknown, it is also possible to obtain the virtual position from f_d, f_n and δx (Jung & Hsia, 1995). In this case, by substitution of k_e in (12) the following virtual position x_v is obtained:

$$x_v = \begin{cases} x_e + \dfrac{f_d}{k_d} & \text{if } f_n = 0 \\[2ex] x_e + f_d \left(\dfrac{k_d \delta x + f_n}{k_d f_n} \right) & \text{if } f_n \neq 0 \end{cases} \tag{13}$$

which is valid for contact and non-contact condition. This approach enables the classical impedance controller, given by (6), with force tracking capability without explicit knowledge of the environment stiffness. Notice that $\dot{x}_v$ is usually assumed to be zero due to the noise always present in the force sensor measurements.

3.2 Impedance control with force tracking

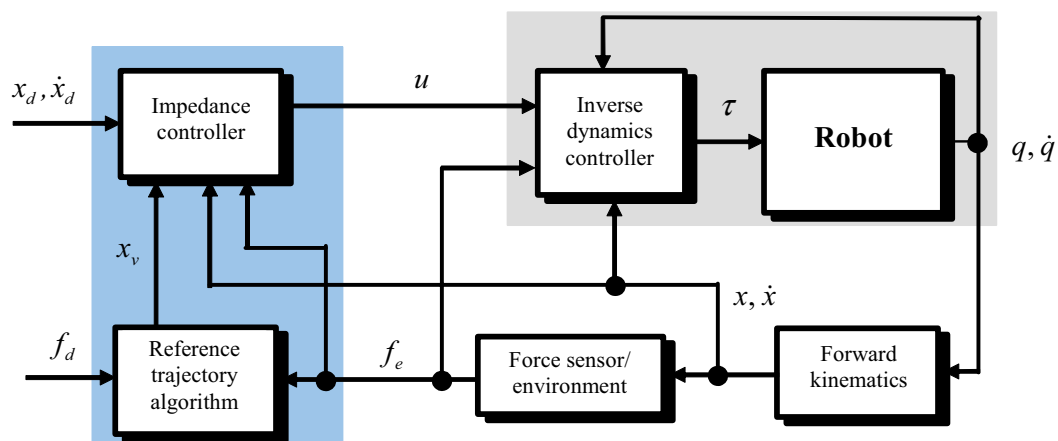The impedance control with force tracking (ICFT) block diagram is presented in fig.2.

Figure 2. Impedance control with force tracking (ICFT) block diagram.

In this scheme, the virtual position x_v given by (13), is computed in the *Reference trajectory algorithm* block, while the target acceleration vector

$u = \begin{bmatrix} u_f & u_p \end{bmatrix}^T$ with u_p is obtained from (6) and u_f from (7), is computed in the *Impedance controller* block. Moreover, the unconstrained target acceleration vector u_p is further compensated by a proportional-derivative (PD) controller, which is given by:

$$u_{pc} = u_p + K_P e + K_D \dot{e} \tag{14}$$

where K_P and K_D are proportional and derivative gain matrices, respectively. The target acceleration vector $u_c = \begin{bmatrix} u_f & u_{pc} \end{bmatrix}^T$ is then used as the driving signal to the inverse dynamics controller, in order to track the desired force profile. Since robot controllers are usually implemented in the joint space, it is useful to obtain the correspondent target joint acceleration u_q for the inverse dynamics controller.

Then, using the appropriate kinematics transformations, u_q is given by:

$$u_q = J^{-1}\left(u_c - \dot{J}(q)\dot{q}\right) \tag{15}$$

Then, applying an inverse dynamics controller in the inner control loop, the joint torques are given by:

$$\tau = \hat{M}(q)u_q + \hat{g}(q) + J(q)^T f_e \tag{16}$$

where $\hat{M}(q)$, $\hat{g}(q)$ are estimates of $M(q)$, $g(q)$ in the robot dynamic model (1). Notice that Coroilis and friction effects are neglected. The impedance controller with force tracking (ICFT) presented above is a good control approach for rigid environments since the end-effector velocity in the constrained direction is close to zero, which leads to a virtual position with an acceptable precision. However, for nonrigid environments the constrained velocity can hardly be zero, which limits the accuracy of the control system to track the desired force profile (Baptista et al, 2001a). To overcome the drawbacks of the scheme presented above, this paper proposes an alternative force control methodology based on a model predictive algorithm (MPA) which is presented in the next section.

4. Model predictive algorithms applied to force control

Predictive algorithms consist of a broad range of methods, which are used to predict a desired variable in an optimal way. The most common predictive algorithms are model predictive controllers (Maciejowski, 2002), which have one common feature; the controller is based on the prediction of the future system behavior by using a process model. In a more general way, predictive algorithms are based on the following basic concepts:

1. Use of a (nonlinear) model to predict the process outputs at future time periods over a prediction horizon;
2. Computation of a sequence of future inputs using the model of the system by minimizing a certain objective function;
3. Receding horizon principle; at each sampling period the optimization process is repeated with new measurements, and only the first input obtained is applied to the system.

In this paper, an MPA is used to predict the target position x_v to the impedance control law in (7), such that a desired force profile is obtained. In general, a predictive algorithm minimizes a cost function over a specified prediction horizon H_p. In order to reduce model-plant mismatch and disturbances in an effective way, the predictive algorithm is combined with an internal model control (IMC) structure (Economou et al., 1986) which increases the force tracking performance. A filter is included in the feedback loop of the predictive structure to reduce the noise present in the force sensor data. This filter stabilizes the loop by decreasing the gain, increasing the robustness of the force control loop. The sequence of future target positions $x_v(k).....x_v(k + H_p - 1)$ over a specified prediction horizon, produced by the MPA, results in a new target acceleration by the impedance control law (6), which determines the force to apply on the surface. Predictive algorithms need a prediction model to compute the optimal input. In this paper, the model must predict the contact force f_m based on the measured position x and velocity $\dot{x}$. This model must consider the dynamics of the environment given by (3). In order to minimize the number of calculations during the nonlinear optimization procedure, only the virtual trajectory is computed in an optimal way, and thus $\dot{x}_v$ is assumed to be zero. Therefore, the nonlinear prediction model in the constrained direction is given by:

$$m_d u_f + b_d \dot{x} - k_d (x_v - x) = -f_m \tag{17}$$

Note that a discrete version of this model is required, predicting the future values $f_m(k+i)$ based on the measured position $x(k)$ and the measured velocity

$\dot{x}(k)$ at time instant k. The predictive scheme is combined with an internal model control scheme, and the model-plant mismatch is given by

$$e_m(k) = f_n(k) - f_m(k) \qquad (18)$$

The desired force profile f_d is compensated by the filtered modeling error e_{mf}, as shown in fig.4, resulting in the modified force reference f_{dc} defined as:

$$f_{dc}(k) = f_d(k) - e_{mf}(k) \qquad (19)$$

The cost function considered for the force control scheme is then given by:

$$J(x_v) = \sum_{i=1}^{H_p} \left(f_{dc}(k+i) - f_m(k+i) \right)^2 \qquad (20)$$

The process inputs and outputs, as well as state variables, can be subjected to constraints, which must be incorporated in the optimization problem.
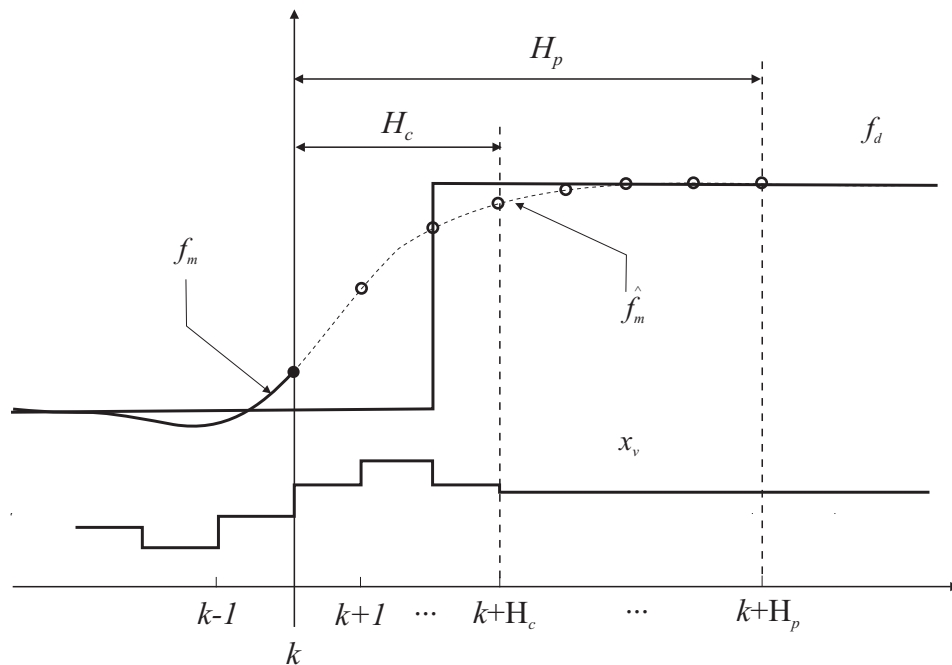

Figure 3. Basic principle of MPA applied to robot force control.

The performance of the MPA depends largely on the accuracy of the process model. The model must be able to accurately predict the future process out-

puts, and at the same time be computationally attractive to meet real-time demands. When both nonlinear models and constraints are present, the optimization problem is nonconvex. Efficient optimization methods for nonconvex optimization problems can be used when the solution space is discretized, and techniques such as Branch-and-Bound - B&B (Sousa et al., 1997) can be applied. The B&B method can be used in a recursive way, demanding less computation effort than other methods, and is used in this paper to solve the nonconvex optimization problem. Figure 3 presents the basic principle of a predictive strategy applied to robot force control.

4.1 Branch-and Bound Optimization

Branch-and-Bound algorithms solve optimization problems by partitioning the solution space. In this paper, B&B is used for the optimization problem that must be solved at each time instant k in the model predictive algorithm. A B&B algorithm can be characterized by two rules: *Branching rule* - defines how to divide a problem into sub-problems; and *Bounding rule* - establishes lower and upper bounds in the optimal solution of a sub-problem, allowing for the elimination of sub-problems that do not contain an optimal solution.
The model predicts the future outputs of the system, which are the forces $f_m(k+1),....,f_m(k+H_p)$ and can be given by (3) when the stiffness coefficient is considered to be constant. Let M be the possible discrete inputs of the system, which are denoted as w_j. Thus, at each step the desired positions $x_v(k+i-1) \in \Omega$, are given by $\Omega = \{\omega_j \mid j = 1, 2, ..., M\}$.
In the considered predictive scheme, the problem to be solved is represented by the objective function (20) minimizing the predicted force error. This optimization problem is successively decomposed by the branching rule into smaller sub-problems. At time instant $k+i$ the cumulative cost of a certain path followed so far, and leading to the output $f_m(k+i)$ is given by

$$J^{(i)} = \sum_{l=1}^{i} \left(f_{dc}(k+l) - f_m(k+l) \right)^2 \tag{21}$$

where $i = 1,...,H_p$, denotes the level corresponding to the time step $k+i$. A particular branch j at level i is created when the cumulative cost $J^{(i)}(u)$ plus a *lower bound* on the cost from the level i to the terminal level H_p for the branch j, denoted J_{L_j}, is lower than an *upper bound* of the total cost, denoted J_U:

$$J^{(i)} + J_{L_j} < J_U \tag{22}$$

Let the total number of branches verifying this rule at level i be given by N. In order to increase the efficiency of the B&B method, it is required that this number should be as low as possible, i.e. $N \square M$.

The major advantages of the B&B algorithm applied to MPA over other non-convex optimization methods are the following: the global discrete minimum containing the optimal solution is always found, guaranteeing good performance; and the B&B method implicitly deals with constraints. In fact, the presence of constraints improve the efficiency of bounding, restricting the search space by eliminating non-feasible sub-problems.

The most serious drawbacks of B&B are the exponential increase of the computational time with the prediction horizon and the number of alternatives, and the discretization of the possible inputs, which are the position references x_v in this paper. A solution to these problems is proposed in the next section.

4.2 Fuzzy scaling machine

Fuzzy predictive filters, as proposed in (Sousa & Setnes, 1999), select discrete control actions by using an adaptive set of control alternatives multiplied by a gain factor. This approach diminishes the problems introduced by the discretization of control actions in MPA. The predictive rules consider an error in order to infer a scaling factor, or gain, $\gamma(k) \in [0,1]$ for the discrete incremental inputs. For the robotic application considered in this paper this error is given by e_m, as defined in (18). The gain $\gamma(k)$ goes to the zero value when the system tends to a steady-state situation, i.e., the force error and the change in this error are both small. On the other hand, the gain increases when the force error or the change in this error is high. When the gain $\gamma(k)$ is small, the possible inputs are made close to each other, diminishing to a great extent, or even nullifying, oscillations of the output. When the system needs to change rapidly the gain is increased and the interval of variation of the inputs is much larger, allowing for a fast response of the system. The fuzzy scaling machine reduces thus the main problem introduced by the discretization of the inputs, i.e. a possible limit cycle due to the discrete inputs, maintaining also the number of necessary input alternatives low, which increases significantly the speed of the optimization algorithm. The design of the fuzzy scaling machine consists of three parts: the choice of the discrete inputs, the construction of the fuzzy rules for the gain filter, and the application of the B&B optimization. The first two parts are explained in the following.

Let the virtual position $x_v(k-1) \in X$, which was described in (17), represent the input reference at time instant $k-1$, where $X = [X^-, X^+]$ is the domain of this reference position. Upper and lower bounds must be defined for the possible changes in this reference signal at time k, which are respectively x_k^+ and

$$x_k^- : \ x_k^+ = X^+ - x_v(k-1), \ x_k^- = X^- - x_v(k-1).$$

These values are then defined as the maximum changes allowed for the virtual reference when it is increased or decreased, respectively. The adaptive set of incremental input alternatives can now be defined as

$$\Omega_k^* = \left\{ 0, \lambda_l x_k^+, \lambda_l x_k^- \,|\, ^\wedge l = 1, \ldots, N \right\} \tag{23}$$

The distribution λ_l must be chosen taking into account that $0 \leq \lambda_l \leq 1$. In this way, the choice of λ_l sets the maximum change allowed at each time instant by scaling the maximum variations x_k^+ and x_k^-. The parameter l is important to define the number of possible inputs. From (23) it follows that the cardinality of Ω_k, i.e., the number of discrete alternatives, is given by $M = 2l + 1$.

The fuzzy scaling machine applies a scaling factor, $\gamma(k) \in [0,1]$ to the adaptive set of inputs Ω_k^* in order to obtain the scaled inputs Ω_k of the optimization routine, the B&B in this case:

$$\Omega_k = \gamma(k) \cdot \Omega_k^* \tag{24}$$

The scaling factor $\gamma(k)$ must be chosen based on the predicted error between the reference and the system's output, which is defined as

$$e(k + H_p) = f_{dc}(k + H_p) - f_n(k + H_p), \tag{25}$$

where $f_{dc}(k + H_p)$ is the reference to be followed at time H_p, as in (19). Added to the error, the change in the error gives usually important indications on the evolution of the system behavior. This information can also be considered in the derivation of $\gamma(k)$. The change in error is given by

$$\Delta e(k) = e(k) - e(k-1). \tag{26}$$

The fuzzy rules to be constructed have as antecedents the predicted error and the change in the error, and as consequent a value for the scaling factor. Simple heuristic rules can be constructed noticing the following. The system is close to a steady-state situation when the error and the change in the error are both small. In this situation, the discrete virtual references must be scaled down, allowing smaller changes in the reference x_v, which yield smaller variations in the impedance controller, and $\gamma(k)$ should tend to zero. On the other hand, when the predicted error or the change in error are high, larger discrete refer-

ences must be considered, and $\gamma(k)$ should tend to its maximum value, i.e. 1. The trapezoidal and triangular membership functions $\mu_e(e(k+H_p))$ and $\mu_{\Delta e}(\Delta e(k))$ define the two following fuzzy criteria: "*small predicted error*" and "*small change in error*", respectively. The two criteria are aggregated using a fuzzy intersection; the minimum operator (Klir, 1995). In this way, the membership degree of these criteria using the min operator is given by:

$$\mu_\gamma(e(k+H_p),\Delta e(k)) = \min(\mu_e,\mu_{\Delta e}), \qquad (27)$$

The scaling factor $\gamma(k)$ must be the fuzzy complement of a certain membership degree μ_γ:

$$\gamma(k) = \bar{\mu}_\gamma = 1-\mu_\gamma. \qquad (28)$$

Summarizing, the set of inputs Ω_k^* at time instant k, which are virtual references in this paper, is defined in (23). These inputs are within the available input space at time k. Further, the inputs are scaled by the factor $\gamma(k)\in[0,1]$ to create a set of adaptive alternatives Ω_k, which are passed on to the optimization routine. At a certain time k, the value of $\gamma(k)$ is determined by simple fuzzy criteria, regarding the predicted error of the system. Note that the proposed fuzzy scaling machine has only the following design parameters: λ_l, and the membership functions μ_e and $\mu_{\Delta e}$. Moreover, the tuning of these parameters is not a hard task, allowing the use of some heuristics to derive them. Possible constraints on the input signal, which is the virtual trajectory in this paper, are implemented by selecting properly the parameters λ_l.
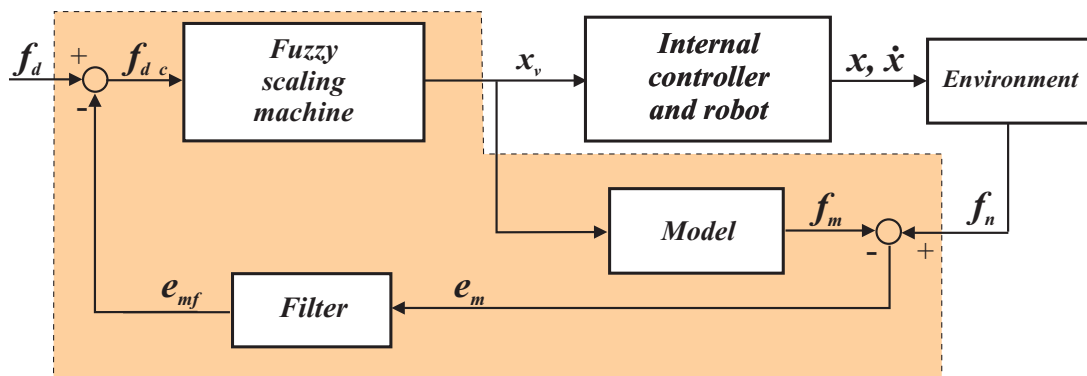

Figure 4. Block diagram of proposed predictive force control algorithm with fuzzy scaling machine.
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001a) with kind permission of Springer Science and Business Media).

Figure 4 depicts the proposed predictive force control algorithm with fuzzy scaling. The block *Fuzzy scaling machine* contains the model predictive algorithm, the B&B optimization and the fuzzy scaling strategy. The block *Internal controller and robot* implement the impedance and the inverse dynamics control algorithms. The robot dynamic model equations are also computed in this block. The block *Environment* contains the nonlinear model of the environment. In order to cope with disturbances and model-plant mismatches, an internal model controller is included in the control scheme. The block *Filter* belongs to the IMC structure (Baptista et al., 2001a).

5. Simulation results

The force control scheme introduced in this paper is applied to a robot through computer simulation for an end-effector force/position task in the presence of robot model uncertainties and inaccuracy in the environment location and the correspondent stiffness characteristics. The robot model represents the links 2 and 3 of the PUMA 560 robot. In all the simulations, a constant time step of 1 ms is used. The overall force control scheme including the dynamic model of the PUMA robot is simulated in the Matlab/Simulink environment. A nonrigid friction contact surface is placed in the vertical plane of the robot workspace where it is assumed that the end-effector always maintain contact with the surface during the complete task execution.

In order to analyze the force control scheme robustness to environment modeling uncertainties, a non rigid time-varying stiffness profile $k_e(t)$ is considered, given by:

$$k_e(t) = \begin{cases} 1000 + \sin(\pi t/2) & 0 < t < 2 \\ 1000e^{(-0.25(t-2))} & 2 \leq t < 3 \end{cases} \tag{29}$$

The damping coefficient and the coefficient of dry friction are settled to ρ_e=45 Ns/m^2 and μ=0.2, respectively. Notice that the stiffness coefficient is considered to be constant (k_e=1000 N/m) in the environment model used for predict the contact force f_m. The matrices in the impedance model (6) are defined as M_d = diag[2.5 2.5] and K_d = diag[250 2500] to obtain an accurate force tracking in the *x*-axis direction and an accurate position tracking performance in the *y*-axis direction.

The matrix B_d is computed to obtain a critically damped system behavior. The control scheme was tested considering a smooth step force profile of 10 N and a desired position trajectory from p_1 = [0.5 -0.2] m to p_2 = [0.5 0.6] m. Uncertainties in the location of the contact surface given by the final real position of p_{2r}=[0.512 0.6] m are considered in the simulations, as shown in fig.5.
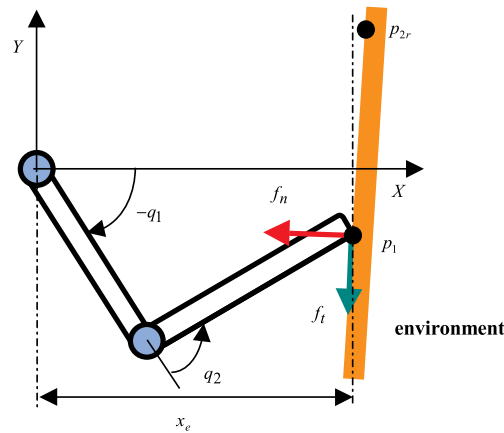
Figure 5. 2-DOF planar robot in contact with the environment.
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001a) with kind permission of Springer Science and Business Media).

The parameters of the predictive controller are $H_p = H_c = 2$ and the fuzzy scaling machine is applied only during the constant path of the reference force trajectory. This means that during the reference force transition periods, the fuzzy scaling inference is switched off. The discrete alternatives Δx_v for the fuzzy scaling machine are given by:

$$\Omega_k^* = \begin{bmatrix} -0.050 & 0 & 0.050 \end{bmatrix} \tag{30}$$

In the inner loop controller (16), only the elements of the inertia matrix and the gravitational terms with parameters 20% smaller than their exact values are considered. The Coriolis and friction terms were neglected in the implementation of the algorithm but considered in the simulation of the robot dynamic model. The proportional and derivative gains in (14) are settled to $K_P =$ diag[5000 5000] and $K_D =$ diag[500 500].
Simulations using the impedance controller with force tracking (ICFT) and the control algorithm proposed in this paper are compared. The conventional impedance controller uses the reference trajectory algorithm presented in (13) considering the environment modeled as a linear spring with k_e=1000 N/m. The simulation results obtained with the ICFT are presented in fig.6, which exhibit poor force tracking performance with relatively large force tracking errors. However, the ICFT follows the desired position trajectory with high accuracy; in fact, it is not possible to distinguish the reference from the actual y-axis position in fig.6.
The force control scheme uses the model predictive algorithm to compute the virtual trajectory x_v, the fuzzy scaling machine and the nonlinear environment model, which furnish the normal force described by (3). The force and position

results from the application of this controller are presented in fig.7. Comparing this figure with fig.6, it becomes clear that the proposed force controller presents a remarkable performance improvement in terms of force tracking capability. In fact, it is not possible to distinguish the reference force from the actual contact force. In terms of position control, similar performance is achieved. The results for both controllers can be compared in Table 1, where the error norm $\|\cdot\|$ for position and force errors, as well as the absolute maximum values for these errors are presented. The table shows that the force control performance is clearly superior for the MPA with fuzzy scaling machine.

Force control algorithms	$\|e_p\|$ [m]	Max(e_p) [mm]	$\|e_f\|$ [N]	Max(e_f) [N]
Impedance control with force tracking	0.041	0.836	60.426	4.420
MPA with fuzzy scaling machine	0.041	0.801	0.8021	0.064

Table 1. Euclidian norm of position, force errors and absolute maximum errors.
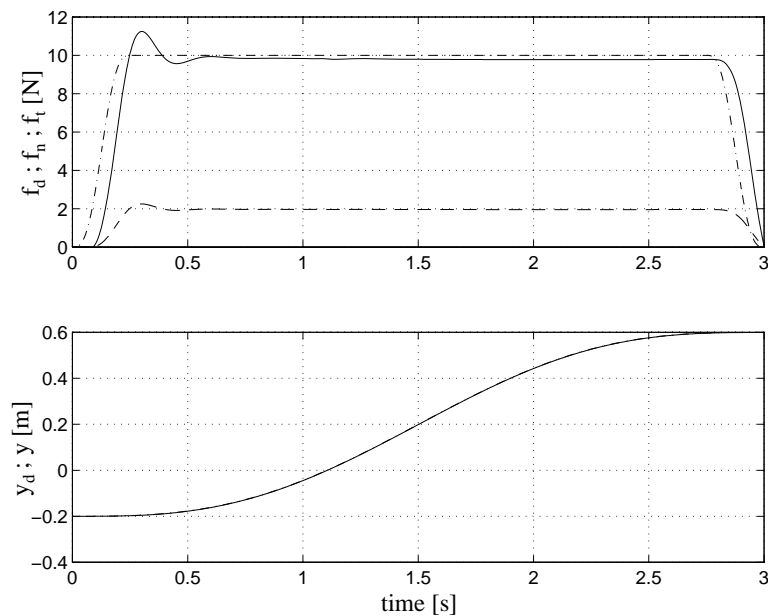

Figure 6. Impedance control with force tracking: desired force (dashdot), normal force (solid) and friction force (dashed) – top view; desired y-axis trajectory (dashdot) and actual position trajectory (solid) – bottom view.
 (Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001a) with kind permission of Springer Science and Business Media).

Figure 7. MPA with fuzzy scaling: desired force (dashdot), normal force (solid) and friction force (dashed) – top view; desired *y*-axis trajectory (dashdot) and actual position trajectory (solid) – bottom view.

(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001a) with kind permission of Springer Science and Business Media).
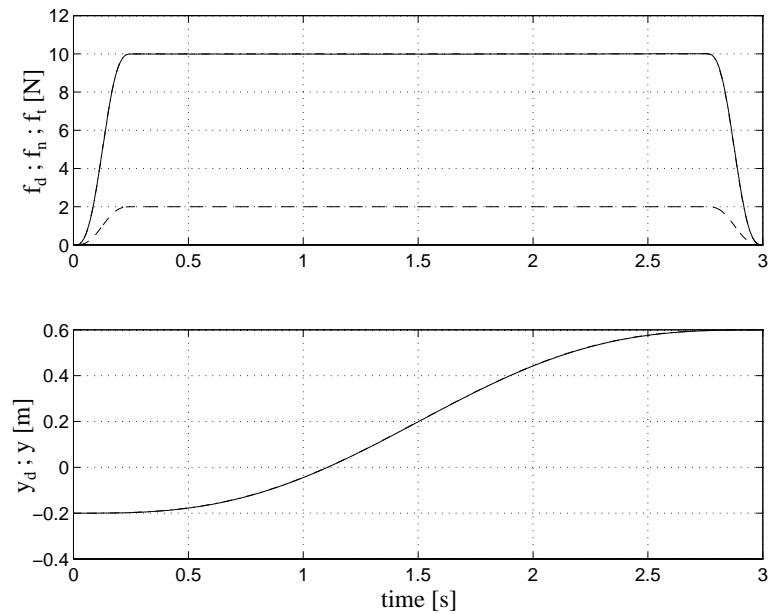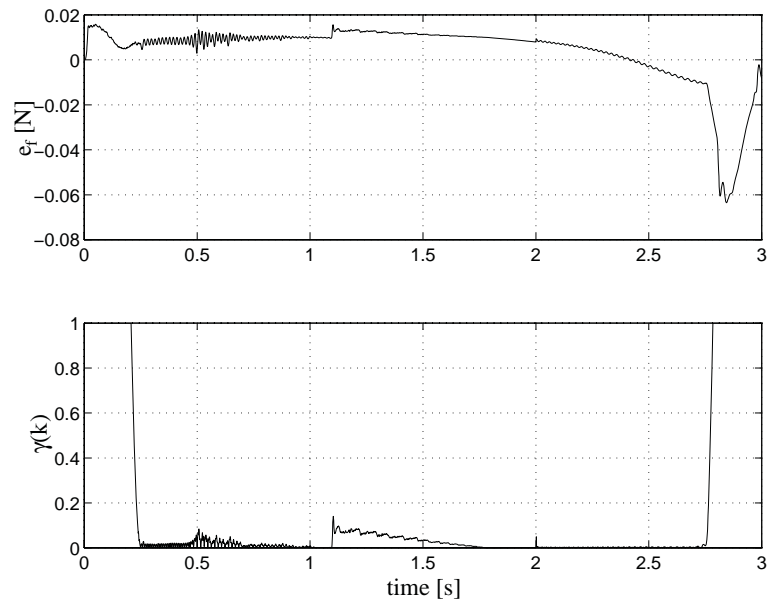

Figure 8. MPA with fuzzy scaling: contact force errors (top view) and fuzzy scaling factor γ(k) (bottom view).

(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001a) with kind permission of Springer Science and Business Media).
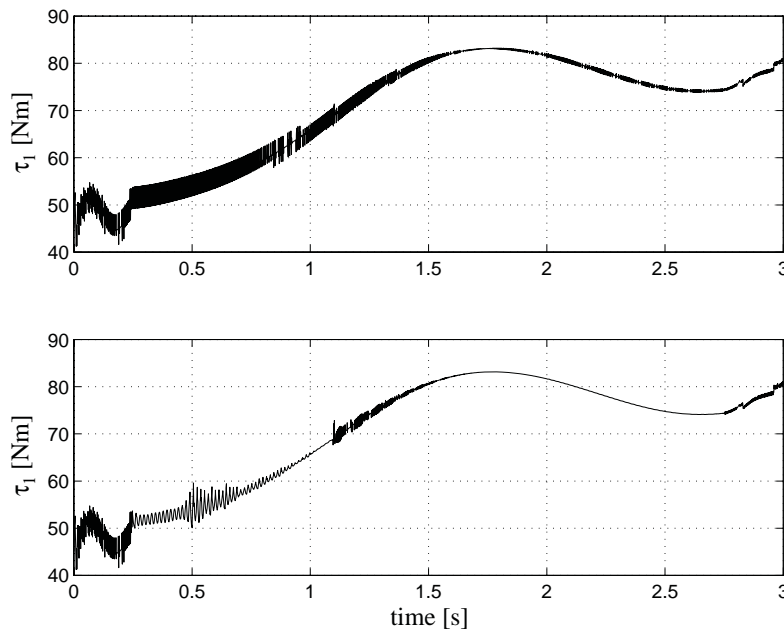
Figure 9. MPA with fuzzy scaling: joint torque τ_1 without (top view) and with fuzzy scaling machine (bottom view).

(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001a) with kind permission of Springer Science and Business Media).

In order to study in more detail the proposed force approach, fig. 8 presents the contact force error and the fuzzy scaling factor $\gamma(k)$ for the same trajectory. The factor $\gamma(k)$ exhibits a fast convergence to values around zero during the constant reference force path, which reduces the chattering present on the target trajectory and in the joint torques.

The joint torque τ_1 using the predictive approach with and without the fuzzy scaling machine is shown in fig. 9. The strategy without fuzzy scaling produce undesirable oscillations on the virtual trajectory x_v, which has the same effect on the joint torques τ. This effect is significantly reduced by the fuzzy scaling machine, as shown in fig.9 for the joint torque τ_1.

6. Experimental setup

In order to validate the proposed force control scheme presented in previous sections, real-time experiments were carried out with a 2-DOF planar manipulator and a nonrigid mechanical environment built at the Robotics Laboratory of Technical University of Lisbon/Instituto Superior Técnico (IST). A low cost open PC-based control architecture using a commercial servo-axis interface board was developed to control the robotic setup in an effective and reliable way.
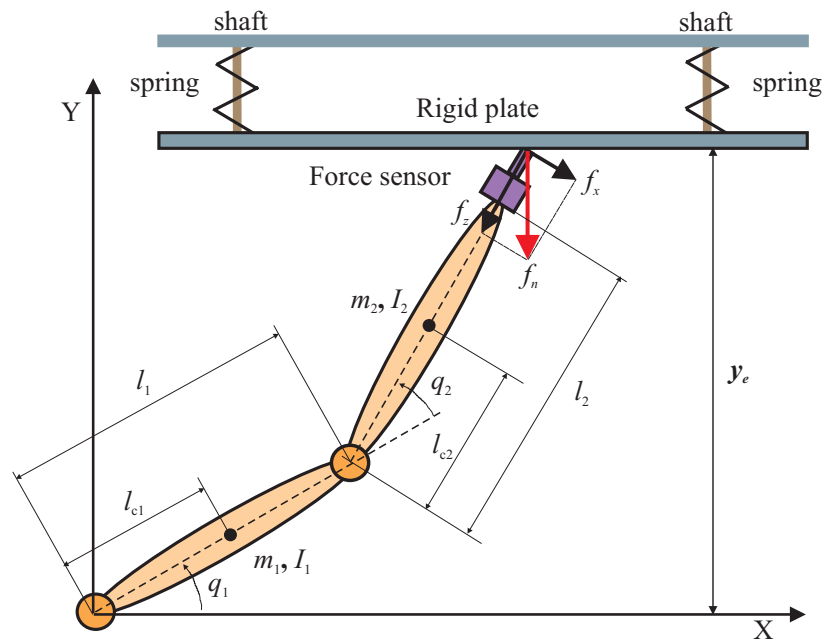
Figure 10. Top view schematic representation of the robotic setup. (Note: The arrows represent the fx, fz components and indicate the negative directions of the contact forces measured by the force sensor.).
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001b) with permission of Elsevier).

The planar robot has two revolute joints driven by HARMONIC DRIVE actuators (HDSH-14). Each actuator has an electric d.c. motor, a harmonic drive, an encoder and a tachogenerator. The robot links parameters are given in Table 2 where l_{ci} is the distance to the mass center of the link i, I_{zzi} is the inertia moment related to the z-axis and I_{mi} is the actuator's inertia moment related to the output shaft. The contact surface used for force control experiments is based on a steel plate with two springs in parallel guided by shafts with ball bearings (Baptista, 2000a). The top view of the planar robot and the nonrigid mechanical contact surface are shown in fig.10 and a picture of the robot and mechanical environment is depicted in fig.11.

Link I	l_i [m]	m_i [Kg]	l_{ci} [m]	I_{zzi} [Kg.m²]	I_{mi} [Kg.m²]
1	0.320	5.00	0.163	0.120	0.081
2	0.330	1.47	0.137	0.018	0.021

Table 2. Parameters of the planar 2-DOF robot. (Note: link length l_2 includes the force sensor length and the end-effector device length.)

Figure 11. Top view of the 2-DOF robot and the nonrigid mechanical environment.
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001b) with permission of Elsevier).

6.1 Control hardware and software tools

The control hardware is based on a PC Pentium 200 MHz, a servo-axis I/O card, linear power amplifiers and a 6-axis force/torque sensor. The power amplifiers are configured to operate as current amplifiers. In this functioning mode, the input control signal is a voltage in the range ± 10 V with current ratings in the interval [-2; 2] A. The signals are processed through a low cost ISA-bus servo I/O card from ServoToGo, Inc. The contact forces are measured by a JR[3] 6-axis force/torque sensor mounted at the end-effector of the arm. The force and torque signals are computed by a ISA-bus DSP card with a frequency rate of 8 KHz. Note that in order to reduce the surface friction during motion, a ball bearing is mounted at the end-effector contact device. The simplified control hardware architecture is shown in fig.12.

The real-time software tool used to implement the force control experiments was developed in the C++ language for the MS-DOS operating system. Real-time performance is guaranteed by reprogramming the Programmable Interval Timer (PIT) of the computer system (Baptista, 2000a). The periodic interrupt generated by the servo-axis I/O board PIT is configured to the Interrupt Request 5 (IRQ 5), which activates the Interrupt Service Routine 13 (ISR 13). This ISR is reserved for the control algorithm implementation at a sampling frequency F_s=1 KHz. The software uses also the ISR 8 which is activated by the computer's IRQ 0.
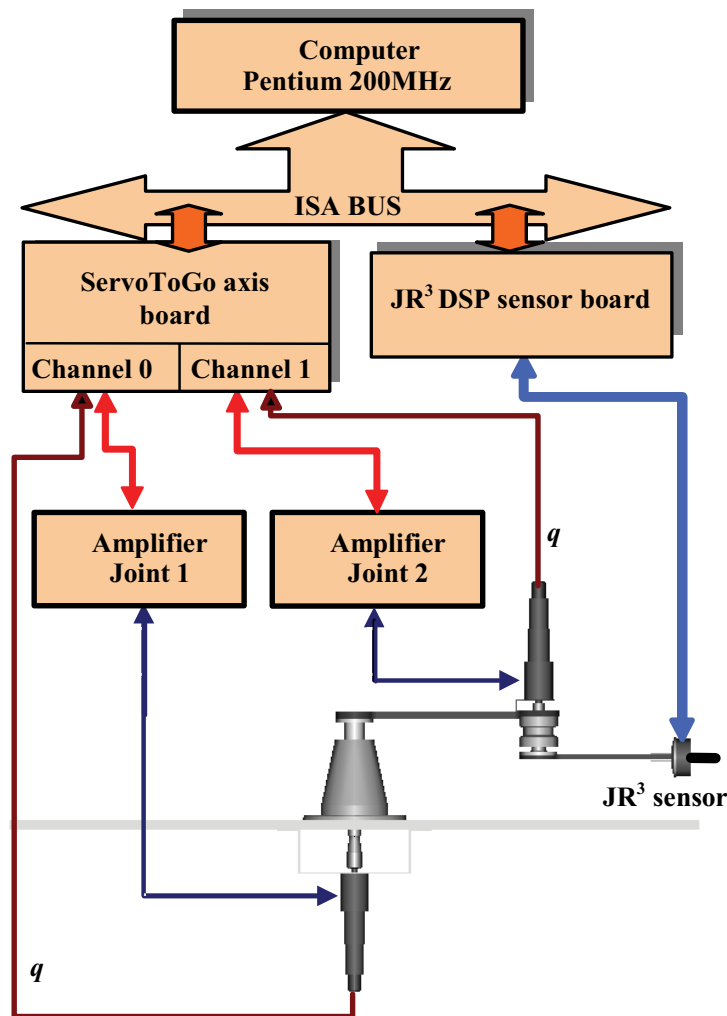
Figure 12. Simplified hardware control architecture of the 2-DOF robot.

This ISR is redefined to execute the safety routine at a frequency that is the double of the sampling frequency. In order to avoid the accumulation of interrupt requests, the ISR 13 increments a flag at the beginning of the execution of the selected control routine. This flag is decremented at the end of the execution of the procedure and is checked by the safety routine, which aborts the real-time procedure if its content is superior to the unit value. In this situation, the failure is reported to the operator by the interface program.

6.2 Real-time force control algorithms

For real-time implementation of the force control algorithms presented in previous sections, some simplifications were considered due to the higher number of calculations that must be performed in real-time. Let's first consider the

classical model-based impedance controller, derived from (2) and (5), repre-
sented by the following equation:

$$\tau = J^T \left(M_x(u + M_d^{-1}(B_d\dot{e} + K_d e - f_e)) + C_x(x,\dot{x})\dot{x} + g_x(x) + f_e \right) \tag{31}$$

It is possible to obtain an important simplification considering that $M_d = M_x$.
Moreover, it is also possible to consider ``quasi-static'' conditions, which means
that velocities $\dot{x} \approx 0$ and $\dot{q} \approx 0$. This is a reasonable approximation since force
control tasks are generally executed at low speeds. Thus, the following simpli-
fied impedance control law is obtained:

$$\tau = J(q)^T \left(B_d\dot{e} + K_d e \right) + g(q) \tag{32}$$

This control scheme is equivalent to a proportional-derivative cartesian posi-
tion control law with gravity compensation. It is possible to demonstrate that
for a constant reference x_d and without contact with the environment, asymp-
totic stability of x_d is obtained (de Witt et al, 1997). This control law can be used
to apply a desired contact force on the environment considering the virtual
references given by the target vector $x_t = \begin{bmatrix} x_v & x_d \end{bmatrix}^T$. The application of this vec-
tor in the control law (32), allows a force tracking capability without explicit
knowledge of the environment stiffness coefficient. In order to improve the
force tracking accuracy, an integral control action can be included in (32),
which leads to:

$$\tau = J(q)^T \left(B_d\dot{e} + K_d e + K_i \int e_f dt \right) + g(q) \tag{33}$$

where $e = x_t - x$ and $\dot{e} = \dot{x}_t - \dot{x}$ are redefined as virtual position and velocity er-
rors and $e_f = f_d - f_e$ are the force errors. Taking the previous assumptions in
consideration, the impedance control algorithm with force tracking imple-
mented in real-time is given by (33) where the gravitational terms $g(q)$ are null
since the robot is of planar type. The main steps of the controller implementa-
tion for each sampling period T_s are (Baptista et al., 2001b):

- Step 1: calculation of cartesian positions and velocities, $x(k)$ and $\dot{x}(k)$,
  respectively.
- Step 2: calculation of the normal contact force $f_n(k)$ from JR3 sensor data.
- Step 3: calculation of $x_v(k)$ in the constrained direction (y-axis):

$$
x_v(k) = \begin{cases} x_v(k-1) - \dfrac{\delta f_d(k)}{k_d} & \text{if } f_n(k) \ge 0 \\[3mm] x_v(k-1) - \dfrac{\delta f_d(k) + \kappa_f e_f(k)}{\hat{k}_e} & \text{if } f_n(k) < 0 \end{cases} \tag{34}
$$

where

$$
\delta f_d(k) = f_d(k) - f_d(k-1) \tag{35}
$$

$$
e_f(k) = f_d(k) - f_n(k) \tag{36}
$$

- Step 4: definition of the virtual vector

$$
x_v(k) = \begin{cases} x_d(k) & \square \; x\text{-axis direction} \\ x_v(k) & \perp \; y\text{-axis direction} \end{cases} \tag{37}
$$

- Step 5: calculation of the control actions in the Cartesian space:

$$
f(k) = -B_d \dot{x}(k) + K_d e(k) + K_i \left(T_s I_p(k) \right) \tag{38}
$$

where

$$
I_p(k) = \frac{e(k) + e(k-1)}{2} \tag{39}
$$

- Step 6: calculation of the joint control actions

$$
\tau(k) = J(q(k))^T f(k) \tag{40}
$$

Notice that in (35) the minus signals follow the convention of signals used by the force/torque sensor. In this algorithm, an estimate of the environment stiffness $\hat{k}_e$ is used to compute $x_v(k)$. The term $\kappa_f e_f(k)$ is used to compensate geometric and stiffness uncertainties and other non-modeled terms of the environment.

For purposes of performance analysis, the classical hybrid position/force control algorithm was also implemented in real-time. This algorithm has been proposed by (Raibert & Craig, 1981) and uses a selection matrix $S = \mathrm{diag}(s_j)$ with $j = 1, ..., m$, to separate the position and force subspaces, by the attribution of ``1'' to the force subspace and ``0'' to the position subspace. The matrix S is used to represent the force controlled direction and the matrix I-S to represent the remaining position controlled directions, where I is the iden-

tity matrix. In this algorithm, the control forces are given by:

$$f(k) = S f_f(k) + [1 - S] f_p(k)$$ (41)

The real-time implementation of the control laws for unconstrained space and constrained space, are respectively given by:

$$f(k) = \begin{cases} f_d(k) + K_{Pf} e(k) + T_s K_f I_{If}(k) - K_{Df} \dot{x} & \text{Force subspace} \\ K_P e(k) + T_s K_I I_p(k) - K_D \dot{x} & \text{Position subspace} \end{cases}$$ (42)

where e_f is defined in (37) and I_p follows the definition in (39) for the unconstrained direction. The coefficients K_P, K_I and K_D are the parameters of the PID position controller and K_{Pf}, K_{If} and K_{Df} are the parameters of the PID force controller. Notice that velocity damping is included in the control law for the constrained direction (Mandal & Payandeh, 1995).

Unfortunately, the control scheme presented in fig.4 is not possible to implement directly in real-time due to the high sampling rate used in the force control loop (F_s=1 KHz). Thus, one possible solution is to use more powerful hardware and software tools to implement the control scheme in a straightforward way. Due to computer hardware limitations, an alternative strategy was used to implement the proposed force control algorithm in real-time. The optimized virtual trajectory x_v for the impedance controller is computed off-line in Matlab/Simulink using a realist simulator of the experimental robotic setup. The optimized virtual trajectory calculated off-line with an additional term for compensation of uncertainties and modeling errors is then furnished as a reference signal (x_{vopt}) to the impedance controller described in Section 3. The virtual reference used in the simplified version of the MPA with fuzzy scaling implemented in real-time is then given by:

$$x_v(k) = x_{vopt}(k) + \kappa_f \frac{e_f(k)}{\hat{k}_e}$$ (43)

where κ_f and $\hat{k}_e$ are terms as defined in (34).

7. Experimental results

In this section, experimental results obtained with the predictive force algorithm with fuzzy scaling proposed in this paper are presented. These results are compared with the classical force control algorithms, namely the classical impedance controller and the hybrid position/force controller. Note that the predictive control algorithm uses a nonlinear model of the nonrigid environment where the stiffness coefficient is varying with the penetration depth δx.

It is necessary to obtain an estimate of the environment stiffness in the constrained direction $\hat{k}_e$, which is used in (34) and (43). This estimate is obtained experimentally by measuring the linear surface displacement when a force profile is applied by the end-effector. The collected pairs of $f_n - \delta x$ data are then used to estimate the global stiffness coefficient $\hat{k}_e$ =1636 N/m, using the least-squares method. The obtained results presented in fig.13, exhibit a large deformation of the surface when a 10 N smooth step force profile is applied on the surface. From the plot is possible to observe that the environment has a nonlinear behavior of hysteresis type (Baptista et al., 2001b).

The control algorithms are implemented in the cartesian space with a sampling period of T_s = 1 ms. The implemented force/position task is a linear trajectory with 100 mm length between x_i = 229.5 mm and x_f = 329.5 mm along x-axis, with simultaneous application of a smooth 10 N step force profile along the y-axis, during 8 seconds. It is assumed that the arm is initially in contact with the environment. Moreover, geometric uncertainty in the contact surface of 5 mm at the end of the trajectory is intentionally imposed to the environment. The filtered force data components (f_x, f_z) are obtained from the JR[3] DSP board considering a first order low-pass digital filter with 500 Hz cut-off frequency in order to attenuate the force sensor noise. The predictive force control algorithm, which derives the virtual position to the impedance controller, use the parameters H_c= H_p = 2. As refereed in Section 4, the fuzzy scaling machine is applied only during the constant path of the reference force profile.
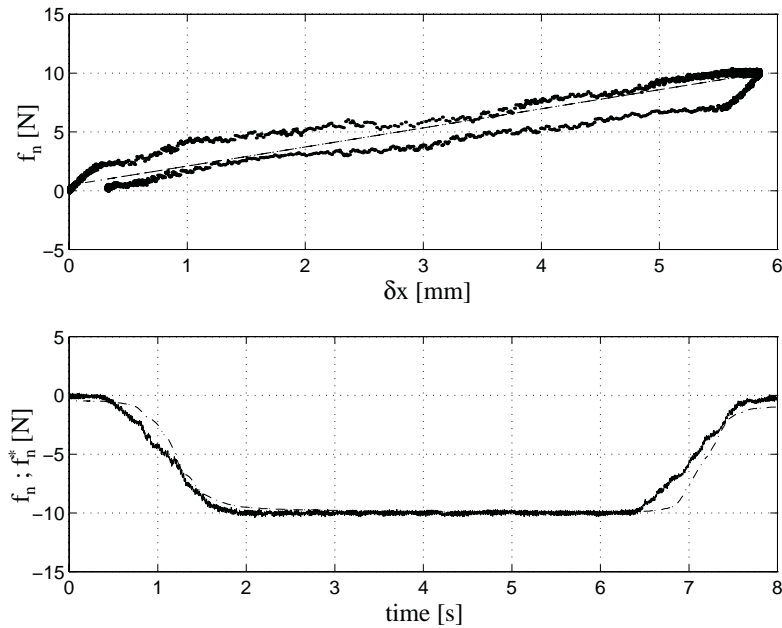
Figure 13. Estimated environment stiffness and contact forces. (Note: dashdot - estimated contact force $\hat{f}_n = \hat{k}_e \delta x$; solid - actual force f_n).

(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001b) with permission of Elsevier).

The maximum allowed increments Δx_v at each time step are the following (in mm):

$$\Omega^* = \begin{bmatrix} -0.75 \times 10^{-2} & 0 & 0.75 \times 10^{-2} \end{bmatrix} \tag{44}$$

The impedance control algorithm use the parameters presented in Table 3 for the real-time control experiments. The force compensation gain is set to $\kappa_f = 0.025$, and was selected from experimental trials.

Axis	K_d	K_i	B_d
x-axis	3000	4000	6
y-axis	2000	3000	2

Table 3. Impedance control parameters.

Figure 14 presents the end-effector's position in the x-y axis coordinates using the real-time implementation of the predictive force control algorithm with fuzzy scaling. For the same controller, the optimized virtual position given by (43) and the actual contact force are presented in fig.15. Figure 16 presents the obtained position and force errors. The experimental results obtained with the classical impedance controller are presented in fig. 17, where the position and

force errors are depicted. The parameters for the hybrid position/force controller used in the real-time tests are presented in Table 4 and the position and force errors are depicted in fig.18.

Axis	K_P	K_I	K_D
x-axis	3000	4000	6
y-axis	570	2500	2

Table 4. Parameters of the hybrid position/force controller.

The results presented for the predictive force control algorithm reveal an accurate force tracking accuracy when compared to the equivalent results obtained with the impedance and hybrid control algorithms. The proposed controller exhibit significantly better results, especially during the time-varying force reference trajectory. The impedance controller with force tracking reveal some difficulties in following the force reference path when this is not constant. The force control performance of the hybrid controller drops significantly during the final part of the trajectory due to the geometric uncertainty, which is not efficiently compensated by the algorithm.
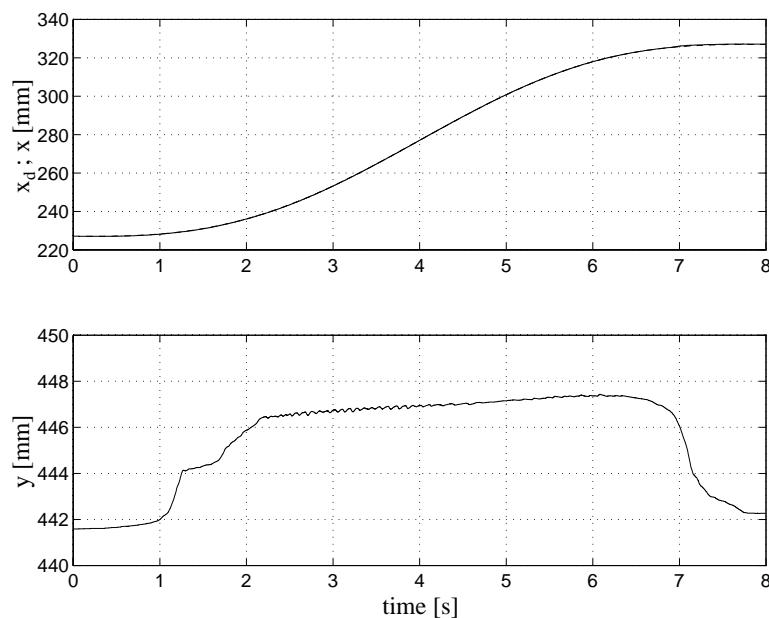

Figure 14. Predictive force control algorithm with fuzzy scaling: *X*-axis position coordinate, *Y*-axis constrained position coordinate.
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001b) with permission of Elsevier).
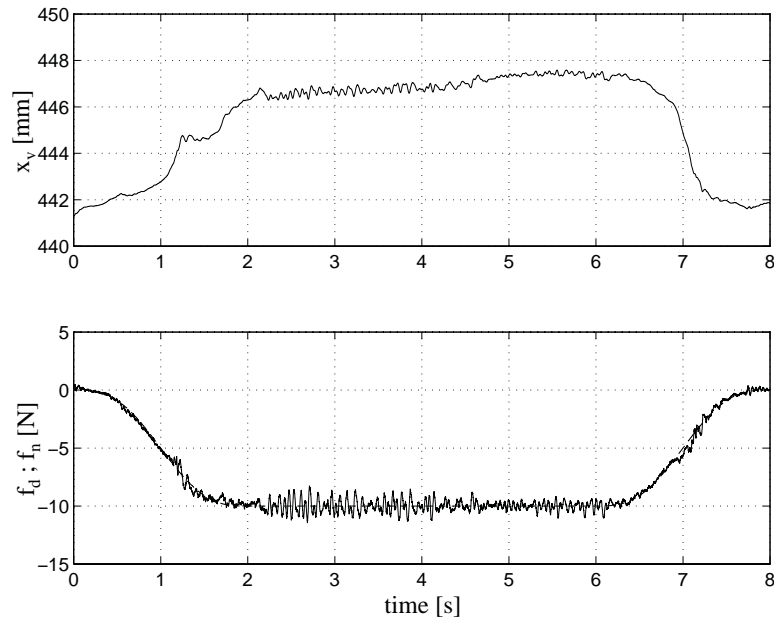
Figure 15. Predictive force control algorithm with fuzzy scaling: *Y*-axis virtual trajectory, desired and actual contact force (Note: solid-actual; dashdot-desired).
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001b) with permission of Elsevier).
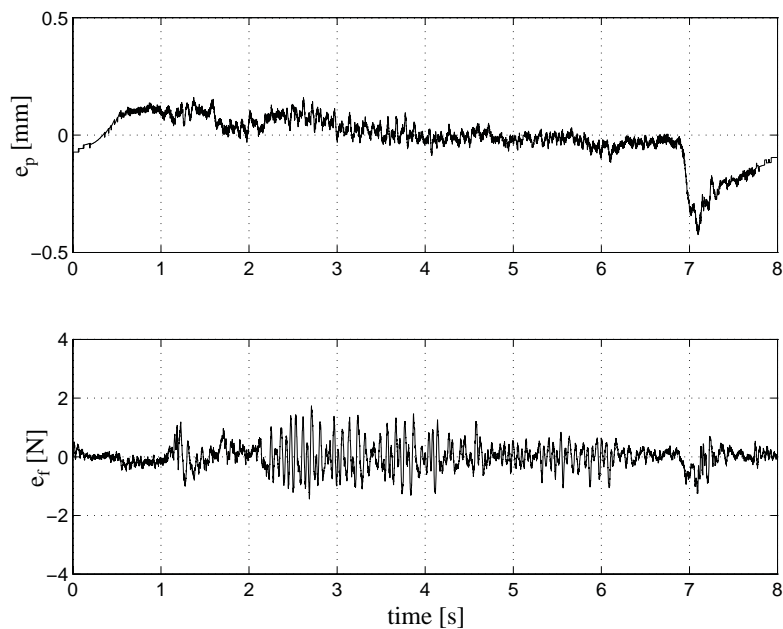

Figure 16. Predictive force control algorithm with fuzzy scaling: *x*-axis position errors and *y*-axis contact force errors.
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001b) with permission of Elsevier).

Figure 17. Impedance control with force tracking: *x*-axis position errors and *y*-axis contact force errors.
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001b) with permission of Elsevier).
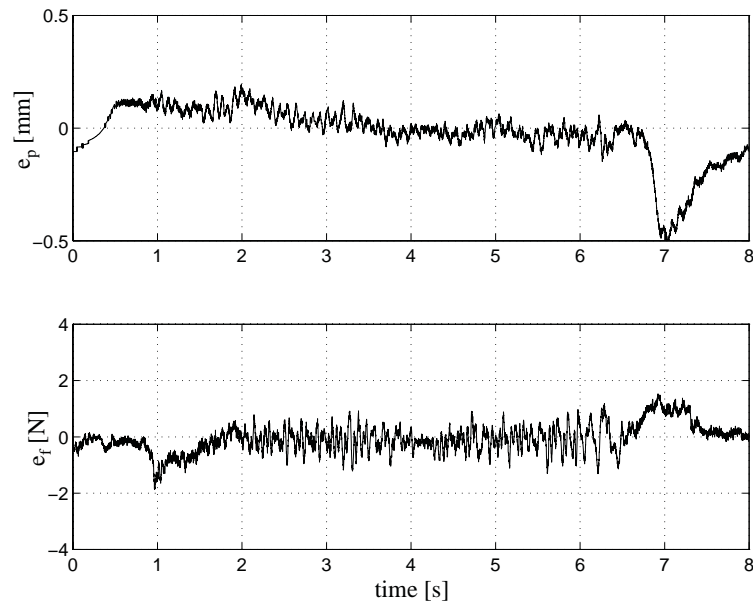
In order to compare the performance of the controllers, the Euclidian norm ($\|\cdot\|$) of position and force errors and the absolute maximum error values for the given task are presented in Table 5.

Force controllers	$\|e_p\|$ [m]	Max(e_p) [mm]	$\|e_f\|$ [N]	Max(e_f) [N]
Hybrid position/force control	8.1	0.43	63.89	2.76
Impedance control with force tracking	8.5	0.49	50.63	2.33
Predictive force control with fuzzy scaling	8.1	0.42	37.40	1.73

Table 5. Euclidian norm of position, force errors and absolute maximum errors.

From Table 4 results, is possible to conclude that predictive force control algorithm proposed in this paper reduces the norm of the force error in 26% when compared to the impedance controller with force tracking, which is a quite significant value. The reduction in the norm of the force error is even more significant when compared to the hybrid position/force controller results (41%). The experimental results reveal the successful implementation of the proposed predictive force control strategy in real-time, in the presence of non-rigid environments with uncertainties.
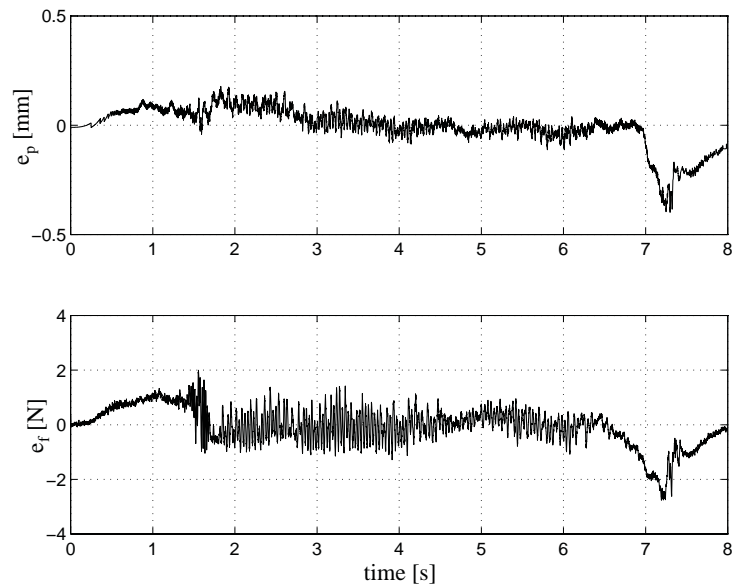
Figure 18. Hybrid position/force control: x-axis position errors and y-axis contact force errors.
(Reprinted from Baptista, L.; Sousa, J. & Sá da Costa, J. (2001b) with permission of Elsevier).

8. Conclusion

In this article, a force control scheme integrating a model predictive algorithm and an impedance controller in the presence of nonrigid environments is proposed. The force controller uses a MPA with a fuzzy scaling machine to alleviate the problems introduced by the discretization of inputs required by the B&B algorithm. The optimization procedure is nonconvex since a nonlinear model of the environment is considered. Simulation results were carried out considering a 2-DOF robot model interacting with a nonrigid environment with nonlinear characteristics. A real-time implementation of the fuzzy predictive algorithm is derived, which uses an optimal virtual trajectory computed off-line. The experimental results are compared to other classical force control techniques, which are the impedance control with force tracking and the hybrid position/force control. Simulation and experimental results reveal that the proposed MPA with fuzzy scaling force controller presents a significant better force tracking performance when compared to the conventional force controllers.

Future research will concentrate on the on-line implementation of the full version of the fuzzy predictive force control algorithm and the extension of the control scheme to a robot with more than two joints.

9. References

Baptista, L.; (2000a). *Adaptive force and position control of robot manipulators,* PhD Thesis, Technical University of Lisbon, Mechanical Engineering Department, Lisbon (in portuguese)

Baptista,L; Ayala Botto, M. & Sá da Costa, J. (2000b). A predictive force control approach of robot manipulators in nonrigid environments. *International Journal of Robotics and Automation,* 15(2) pp.94-102, ISSN 0826-8185

Baptista,L; Sousa, J. & Sá da Costa, J. (2001a). Force control of robot manipulators using a fuzzy predictive approach. *Journal of Intelligent and Robotic Systems,* 30(4), April 2001, pp.359-376, ISSN 0921-0296

Baptista,L; Sousa, J. & Sá da Costa, J. (2001b). Force predictive algorithms applied to real-time force control. *Control Engineering Practice,* 9(4), pp.411-423, ISSN 0967-0661

Corbet, T. ; Sepehri, N. & Lawrence, P. (1996). Fuzzy control of a class of hydraulically actuated industrial robots. *IEEE Transactions Control Systems Technology,* 4(4), pp.419-426

De Witt, C. ; Siciliano, B. & Bastin, G. (1997). Theory of robot control. Springer-Verlag, ISBN 3-540-76054-7, London, UK

De Schutter, J. ; Bruyninckx, H. & Spong, M. (1997). *Force control: a bird's eye. Proceedings of IEEE CSS/RAS International Workshop on Control Problems in Robotics and Automation: Future directions.* San Diego, December 1997, USA

Economou, C. ; Morari, M. & Palsson, B. (1986). *Internal model control. 5. Extension to nonlinear systems.* Industrial Engineering Chemical Process Design and Development, 25, American Chemical Society, pp.404-411

Hogan, N. (1985). Impedance control: An approach to manipulation: Part I-II-III. *Journal of Dynamic Systems, Measurement and Control,* 107(1), pp.1-24

Jaritz, A. & Spong, M. (1996). An experimental comparison of robust control algorithms on a direct drive manipulator. *IEEE Transactions on Control Systems Technology,* 4, pp.614-626

Jung, S. & Hsia, T. (1995). On neural network application to robust impedance control of robot manipulators. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 869-874

Lin, S. & Huang, A. (1997). Position-based fuzzy force control for dual industrial robots. *Journal of Intelligent and Robotic Systems*, 19, pp.393-409

Liu, M. (1995). Force-controlled-fuzzy-logic-based robotic deburring. *Control Engineering Practice*, 3(2), pp.189-201, ISSN 0967-0661

Love, L. & Book, W. (1995). Environment estimation for enhanced impedance control. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1854-1859

Kieffer, J. & Yu, K. (1999). Robotic force/velocity control for following unknown contours of granular materials. *Control Engineering Practice*, 7(10), October 1999, pp.1249-1256, ISSN 0967-0661

Klir, G. & Yuan, B. (1995). *Fuzzy set and fuzzy logic: theory and applications*, Englewood Cliffs, Prentice Hall, ISBN 0792377338, NJ

Maciejowski, J. (2002). *Predictive control with constraints*, Prentice Hall Inc., New York, USA, ISBN 0201398230

Marhefka, D. & Orin, D. (1996). Simulation of contact using a nonlinear damping model. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.1662-1668

Mandal, N. & Payandeh, S. (1995). Control strategies for robotic control tasks: an experimental study. *Journal of Robotic Systems*, 12(1), pp.67-92

Raibert, M & Craig, J. (1981). Hybrid position/force control of manipulators. *Journal of Dynamic Systems, Measurement and Control*, 102(2), pp.126-133

Siciliano, B. & Villani, L. (2000). *Robot force control*, Springer-Verlag, New York, ISBN 0792377338

Singh, S, & Popa, D. (1995). An analysis of some fundamental problems in adaptive control of force and impedance behavior: theory and experiments. *IEEE Transactions on Robotics and Automation*, 11, pp.912-921

Sousa, J. ; Babuska, R. & Verbruggen, H. (1997). Branch-and-bound optimization in fuzzy predictive control: an application to an air conditioning system. *Control Engineering Practice*, 5(10), October 1997, pp.1395-1406, ISSN 0967-0661

Sousa, J. & Setnes, M. (1999). Fuzzy predictive filters in model predictive control. *IEEE Transactions on Industrial Electronics*, 46(6), December 1999, pp.1225-1232

Volpe, R. & Khosla, P. (1995). On the equivalence of second order impedance control and proportional gain explicit force control. *International Journal of Robotics Research*, 14(6), pp.574-589

Yao, B. & Tomizuka, M. (1995). Adaptive control of robot manipulators in constrained motion – controller design. *Journal of Dynamic Systems, Measurement and Control*, 117, pp.320-328

Wada, H. ; Fukuda, T. ; Kosuge, K. ; Arai, F. & Watanabe, K. (1993). Damping control with consideration of dynamics of the environment. *Proceedings of the International Conference on Intelligent Robots and Systems,* 1, pp.1516-1520

Zeng, G. & Hemami, A. (1997). An overview of robot force control. *Robotica*, 15(5), pp.473-482

Friction Compensation in Hybrid Force/Velocity Control for Contour Tracking Tasks

Antonio Visioli, Giacomo Ziliani and Giovanni Legnani

1. Introduction

Nowadays robots in industrial settings are mainly used for repetitive tasks where they act as programmable devices reproducing previously recorded motions in a highly structured environment so that decision and initiative capability is rarely exploited. Contour tracking is, on the contrary, an example of a complex task that requires the manipulator to continuously and autonomously modify its path, coping with the uncertainties typical of unstructured environments (Siciliano & Villani, 1999). In many applications a robot is required to follow a contour while applying a normal force; these tasks include grinding (Thomessen & Lien, 2000), deburring (Ferretti et al., 2000; Ziliani et al., 2005), shape recovery (Ahmad & Lee, 1990), polishing and kinematic calibration (Legnani et al., 2001). The problem of tracking (known and) unknown contours has been studied by many researchers in the last two decades.
Hybrid force/velocity control (Raibert & Craig, 1981) appears to be suitable to be adopted in this context, because it explicitly controls the end-effector force in a selected direction and the end-effector velocity in the other complementary directions. Actually, two kinds of hybrid force/velocity control can be implemented (Roy & Whitcomb, 2002): 1) *explicit hybrid force/velocity control*, where the robot end-effector is controlled by directly imposing the joint torques based on the measured force and position/velocity errors, and 2) *implicit hybrid force/velocity control*, where the end-effector is controlled indirectly by suitably modifying the reference trajectories of the joint position/velocity inner control loops based on the measured force errors. A theoretical comparison (with experimental results) between these two approaches has been developed in (Volpe & Khosla, 1993), although the contour tracking task has not been considered.
Indeed, it is a matter of fact that these methodologies are not widely employed in industrial settings. This might be due to the fact that there is a lack of a characterisation of these techniques from an industrial point of view where the cost/benefit ratio has to be always taken into account. In order to (partially) address this fact, the implementation of an implicit and an explicit hybrid

force/velocity control law for contour tracking of objects of unknown shape performed by an industrial SCARA manipulator is discussed in this chapter. In particular, the problem of compensating joint friction effects, which have to be taken into account in the controller design in order to achieve reasonable performance in terms of normal force and tangential velocity errors, is investigated. Two model-based friction compensation methods are considered: a static method, based on a previously identified model and an adaptive method, where joint friction parameters are recursively updated.

2. Experimental Setup

The experimental set-up adopted for the experiments described in the following sections is available in the Applied Mechanics Laboratory of the University of Brescia and it consists of an industrial robot manipulator manufactured by ICOMATIC (Gussago, Italy) with a standard SCARA architecture where the vertical z axis has been blocked since a planar task is addressed. A detailed dynamic model is described in (Visioli & Legnani, 2002).

Both links have the same length of 0.33 m. The two joints are actuated by means of two DC motors that are driven by conventional PWM amplifiers and position measurements are available by means of two incremental encoders with 2000 pulses/rev. resolution. Harmonic Drive speed reducers are present and the reduction rate is 1/100 for both joints. Velocity is estimated through numerical differentiation whose output is then processed by a low-pass 2-order Butterworth filter with a 100 Hz cut-off frequency and a 1.0 damping ratio.

Figure 1. The SCARA robot during the contour tracking of a complex shape

An ATI 65/5 force/torque sensor capable of measuring forces in a range of ±65 N and with a resolution of 0.05 N is mounted at the manipulator's wrist. The corresponding signals are processed at 7.8 kHz frequency by an ISA DSP based board.

The contact is achieved by means of a proper plastic probe endowed with a ball bearing with an 8 mm diameter whose aim is reducing tangential friction forces that may arise from the contact with the piece (see Figure 1).

The overall control law is implemented (in C/C++ language) by means of a PC-based controller based on a QNX4 real time operating system. Acquisition and control are performed at a 1 kHz frequency.

3. Hybrid Force/Velocity Control

3.1 Problem formulation

A sketch of the SCARA robot is shown in Figure 2. Frame (0) refers to the robot base, while task frame (T) has its origin on the robot end-effector with its n and t axes that are directed respectively along the normal and tangential direction of the contour of the piece, whose geometry is assumed to be unknown; ϑ denotes the angle between n axis and x axis of frame (0).

Let $Q = [q_1, q_2]^T$ be the vector of the joint positions and $\dot{Q}$ its first time derivative. Since a suitable belt transmission keeps the end-effector with constant orientation with respect to the absolute frame, force measurements are directly available in frame (0). Let $F_{(0)} = [F_x, F_y]^T$, $F_{(T)} = [F_t, F_n]^T$ be the vector of the contact force in frame (0) and (T) respectively. They are related to each other by the equation $F_{(0)} = M_{0T}(\vartheta)F_{(T)}$ denoting with M_{ij} the rotation matrix from frame j to frame i. Note that

$$M_{0T}(\vartheta) = \begin{bmatrix} \sin\vartheta & \cos\vartheta \\ -\cos\vartheta & \sin\vartheta \end{bmatrix}. \tag{1}$$

Vector $V_{(T)} = [V_t, V_n]^T$ representing the Cartesian velocity in frame (T) can be obtained from the relation

$$V_{(T)} = M_{T0}(\vartheta)V_{(0)} = M_{T0}(\vartheta)J(Q)\dot{Q} \tag{2}$$

where $J(Q)$ is the robot Jacobian. The aim of the contour tracking task is to control the normal force and the tangential velocity of the robot probe along n and t directions of task frame (T) respectively.

These directions can be easily estimated, assuming that the contact friction force on the tangent direction is negligible with respect to the normal contact force (note that this is achieved by adopting a suitable probe endowed with a ball bearing, as described in Section 2), by on-line estimating the angle ϑ as:

$$\vartheta = \text{atan2}(F_y, F_x) = \arctan\left(\frac{F_y}{F_x}\right) \pm \pi. \tag{3}$$
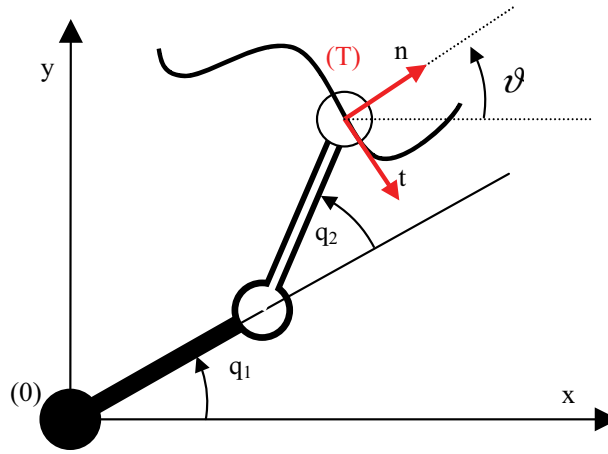

Figure 2. Sketch of a SCARA robot following a contour

3.2 Explicit hybrid force/velocity control

It is well known that the manipulator dynamics can be expressed as:

$$B(Q)\ddot{Q} + C(Q,\dot{Q})\dot{Q} + f(\dot{Q}) + G(Q) = \tau - J(Q)^T F_{(0)} \tag{4}$$

where $B(Q)$ is the inertia matrix, $C(Q,\dot{Q})$ is the matrix of centrifugal and Coriolis terms, $f = [f_1(\dot{q}_1), f_1(\dot{q}_2)]^T$ is the vector of joint friction forces, $G(Q)$ is the gravity forces term (null for the SCARA robot adopted because it works in a horizontal plane), $\tau = [\tau_1, \tau_2]^T$ is the joint torques vector, and $F_{(0)}$ was defined earlier as the vector of forces exerted from the robot on the environment. A thorough theoretical and experimental investigation of the robot identified its dynamics (Visioli & Legnani, 2002; Indri et al. 2002). It was found that, because of the low velocities and accelerations involved in conventional contour tracking tasks, the effects of the inertial and Coriolis forces can be neglected with respect to contact forces and friction terms. As a consequence of this, equation

(4) can be reduced to

$$\tau \cong f(\dot{Q}) + J^T(Q)F_{(0)}. \tag{5}$$

In an explicit hybrid force/velocity control law the robot end-effector is controlled by directly imposing the joint torques based on the measured force and position/velocity errors (*i.e.*, no joint position/velocity inner loops are present). Based on the robot dynamic equation (5), the control scheme shown in Figure 3 can be adopted. The joint torques τ_1 and τ_2 for the first and the second joint respectively are calculated as:

$$\tau = J^T(Q)M_{0T}(U_{(T)} + K_R R) + \hat{f} \tag{6}$$

where $R = [\bar{v}_t, \bar{f}_n]^T$ is the vector of the tangential velocity and normal force reference values, $K_R = diag[k_{V,ff}, k_{F,ff}]$ is the diagonal matrix of feedforward gains, $\hat{f} = [\hat{f}_1(\dot{q}_1), \hat{f}_1(\dot{q}_2)]^T$ is an available estimate of the joint friction torques (see Section 4) and

$$U_{(T)} = [u_{PID,V}, \; u_{PI,F} + k_{v,fb}(\bar{v}_n(t) - v_n(t))]^T \tag{7}$$

where $u_{PID,V}$ is the tangential velocity PID output, $u_{PI,F}$ is the normal force PI output, $\bar{v}_n(t) = 0$, $v_n(t)$ is the velocity of the end-effector in the normal direction and $k_{v,fb}$ is a proportional gain.
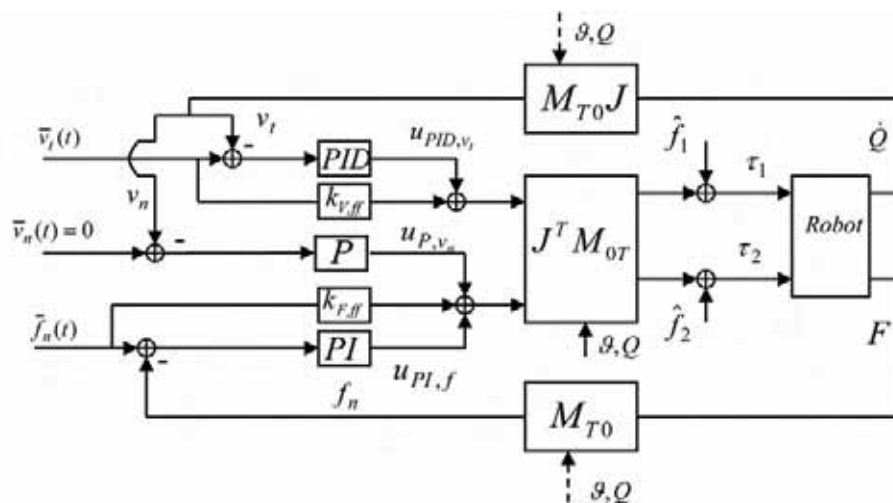

Figure 3. The explicit hybrid force/velocity control scheme

Note that the use of a normal force derivative term has been avoided in (6) (indeed, only the proportional and the integral actions have been employed) as the derivation of such a signal is ill-conditioned (Craig, 1989). Conversely, the adoption of a normal force velocity feedback loop has been proven to be effective to compensate for the large force oscillations due to the effects of link masses (and joint elasticities) in a large portion of the workspace (Jatta et al., 2006). Further, a gain scheduling approach has been adopted in order to take into account the configuration dependent dynamics of the manipulator during a constrained motion (Ziliani et al., 2006).

3.3 Implicit hybrid force/velocity control

In the considered implicit hybrid force/velocity control strategy, a hybrid force/velocity controller determines the reference input for an inner position control loop. The latter is shown in Figure 4, where $\bar{S} = [\bar{x}, \bar{y}]^{T}$ is the end-effector reference position in the Cartesian space and $\bar{q}_1$ and $\bar{q}_2$ are the corresponding joint reference position (which are determined by applying the inverse kinematics). In other words, a standard decentralized position control law (with friction compensation) is applied. The (outer) hybrid force/velocity controller is very similar to that of the explicit hybrid control law and it is depicted in Figure 5.

It is worth stressing at this point that it is claimed in the literature (De Schutter, 1986) that the use of an inner position control loop provides in general several advantages. In particular, the effects of the disturbances in the actuation system are reduced and the force control functionality can be simply added to the existing control architecture devoted to free motion control. Conversely, it is also claimed that the presence of an inner position loop causes a limitation of the bandwidth of the force loop.
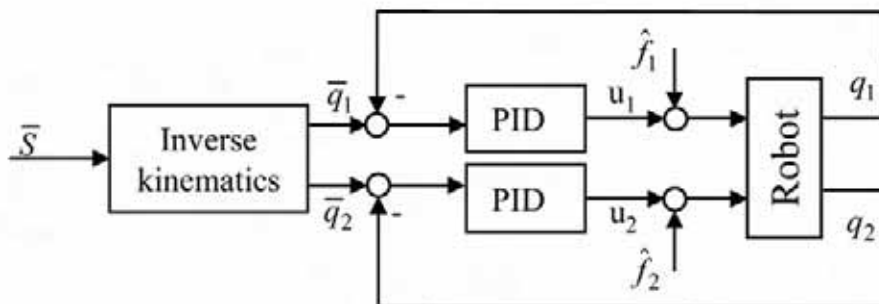

Figure 4. The inner position control loop in the implicit hybrid force/velocity control scheme
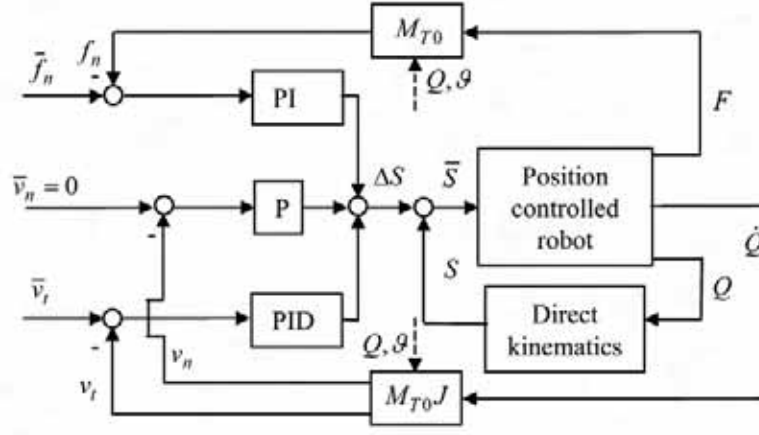
Figure 5. The implicit hybrid force/velocity control scheme

4. Joint Friction Compensation

4.1 Generalities

It is well known that friction compensation is in general very useful to improve the tracking performances of servosystems and this is indeed more significant when the task is dominated by low velocities, as in contour tracking. Actually, friction is a very complex phenomenon and different models to describe it have been proposed in the literature. Basically, they can be classified as static or dynamic models (Olsson et al., 1998). In general, when a static model is adopted, the friction force is described as a (nonlinear) function of the relative velocity of the two surfaces that are in contact. Based on these considerations, the friction terms $f_i(\dot{q}_i)$ mentioned in equations (4) and (5) can be approximated by polynomial functions of degree h (Bona et al., 2003). Positive and negative velocities might be considered separately to obtain better results in case the actual friction function is not symmetrical as might occur in industrial robots (Daemi & Heimann, 1996). Defining ($i = 1,2$):

$$P_i := [p_{i0}\ p_{i1}\ \cdots\ p_{ih}] = \begin{cases} \left[p_{i0}^{-}\ p_{i1}^{-}\ \cdots\ p_{ih}^{-}\right] & \text{if} \quad \dot{q}_i < 0 \\ \left[p_{i0}^{+}\ p_{i1}^{+}\ \cdots\ p_{ih}^{+}\right] & \text{if} \quad \dot{q}_i > 0 \end{cases} \tag{8}$$

and

$$\Omega_i := [1\ \dot{q}_i\ \dot{q}_i^2\ \cdots\ \dot{q}_i^h]^T \tag{9}$$

the friction term can be modelled for the ith axis as $f_i(\dot{q}_i) = P_i\Omega_i$. If $\hat{P}_i$ is an available estimate of vector P_i the joint friction torque can be estimated as

$$\hat{f}_i(\dot{q}_{ref,i}) = \hat{P}_i\Omega_{ref,i} \tag{10}$$

Where

$$\Omega_{ref,i} := [1\ \dot{q}_{ref,i}\ \dot{q}_{ref,i}^2\ \cdots\ \dot{q}_{ref,i}^h]^T \tag{11}$$

and $\dot{q}_{ref,i}$ is the ith joint velocity reference. Note that, $\Omega_{ref,i}$ was used instead of Ω_i because better experimental performance was achieved (Whitcomb et al., 1997). Since reference velocities are not directly available they have to be reconstructed from workspace references as

$$[\dot{q}_{ref,1}, \dot{q}_{ref,2}]^T = J^{-1}(Q)M_{0T}[\bar{v}_t, 0]^T. \tag{12}$$

Two methods for the determination of $\hat{P}_i$ are considered hereafter. They will be referred as Static model-based Friction Compensation (SFC) and Adaptive Friction Compensation (AFC).

4.2 Static Friction Compensation

The first method that has been considered consists of performing suitable experiments on the robot and then of applying an off-line estimation procedure based on a recursive least squares algorithm in order to estimate all the robot parameters including friction torques (h=3 was used) (Indri et al., 2002). Once the (static) friction model is estimated, it can be employed in the control law both for the explicit and the implicit hybrid force/velocity control law during the contour tracking tasks. However, this technique is time consuming (as *ad hoc* experiments have to be performed) and not always appropriate as it is known that friction torques may change over time and might not be always reliably predicted (Daemi & Heimann, 1996). Thus, control performance may decrease during the robot operation and therefore it is useful to compensate for the joint friction effect using an adaptive procedure.

4.3 Adaptive Friction Compensation

The second method considered for the evaluation of $\hat{P}$ is to employ a simple gradient descent based algorithm for each joint (Visioli et al., 1999). For this purpose, a friction error signal can be defined as

$$e = [e_1(\dot{q}_1), e_2(\dot{q}_2)]^T = f(\dot{Q}) - \hat{f} \tag{13}$$

The friction error signal can be determined suitably both for the explicit and the implicit hybrid control law. In the first case, if the model expressed by (5) represents a perfect prediction, then the output of the PID controllers in (6) would be equal to zero. Consequently the PID output can be regarded as a joint friction prediction error signal. In other words, adopting $k_{V,ff} = 0$ and $k_{F,ff} = 1$ equations (5) and (6) result in:

$$\hat{f}(Q) + u_{PID}^* + J^T(Q)F_{(0)} = f(\dot{Q}) + J^T(Q)F_{(0)} \tag{14}$$

where

$$u_{PID}^* = J^T(Q)M_{0T}U_{(T)} \tag{15}$$

is the workspace output of PID regulators transformed into the joint space. By following a similar reasoning, in case the implicit control law is employed, it is

$$\hat{f}(Q) + u_{PID}^* = f(\dot{Q}) + J^T(Q)F_{(0)} \tag{16}$$

where

$$u_{PID}^* = [u_1, u_2] \tag{17}$$

is the vector of the outputs of the joint position control loops.

The friction error signal (12) can therefore be set equal to u_{PID}^*, namely,

$$e = u_{PID}^* \tag{18}$$

where equation (15) or (17) has to be considered if the explicit or the implicit controller is adopted respectively.

Based on the value of the error signal (13), the polynomial coefficients of the friction function can be updated every control step k with the following AFC algorithm ($i=1,2$):

$$\begin{aligned} \Delta\hat{P}_i(k) &= \eta e_i(k)\Omega_i(k) + \alpha\Delta\hat{P}_i(k-1) \\ \hat{P}_i(k) &= \hat{P}_i(k-1) + \Delta\hat{P}_i(k) \end{aligned} \tag{19}$$

representing a standard least-mean-square algorithm (see (Haykin, 1999) for details and for an analysis of the convergence). The updated vector $\hat{P}_i(k)$ can then be used in model (10). Note that parameter η determines the velocity of the descent to the minimum and therefore the adaptation velocity. However, a high value of η might produce undesirable oscillations in the values of the polynomial coefficients. In any case, an appropriate value of η can be easily selected starting from a small value and increasing it until these oscillations begin to appear. Parameter α is the momentum coefficient, that helps to prevent large oscillations of the weight values and to accelerate the descent when the gradient is small. For the experiments presented in Section 5, the values $\eta = 0.005$, $\alpha = 0.9$ and $h = 1$ have been selected. Actually, choosing $h = 1$ means that a local linearization of the friction function is employed for each value of the joint velocity. The capability of modelling a complex nonlinear function, i.e. the friction phenomena is therefore due to the changing of the polynomial coefficients in order to always have an appropriate local approximation (see Figure 6). Indeed, the aim of the AFC technique is not to provide a correct global model of the friction effect, but to compensate for it. Thus, it is sufficient that the friction effect is accurately predicted just in the neighborhood of the joint velocity value in the considered time instant, since the adaptation procedure is always active to compensate for variations.
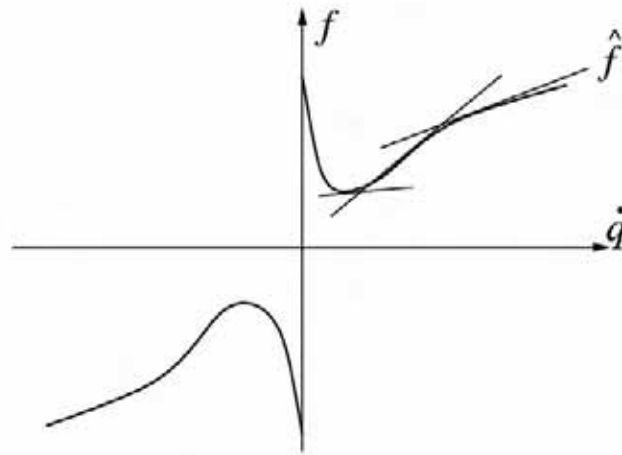

Figure 6. Local linearisation of the friction function

5. Experimental Results

Experimental results, aiming at verifying the effects of the joint friction compensation, have been obtained by considering an iron disk with a diameter of 180 mm placed in two different positions (called `A' and `B') of the manipulator workspace, as shown in Figure 7.

It is worth stressing again that the contour tracking task is performed by the control algorithm without any knowledge of the geometry of the workpiece.

In both cases, a large set of different normal force set-points [20, 25, 30, 35, 40] N, each at different tangential velocities [10, 20, 30, 40, 50] mm/s, has been selected and the RMS force and velocity error has been calculated for both the explicit and implicit hybrid force/velocity control law, where the use of the SFC method and of the AFC method has evaluated together with the absence of joint friction compensation. The control system parameters, namely the parameters of the PID controllers in the explicit and implicit control schemes, have been selected after a careful trial-and-error procedure.
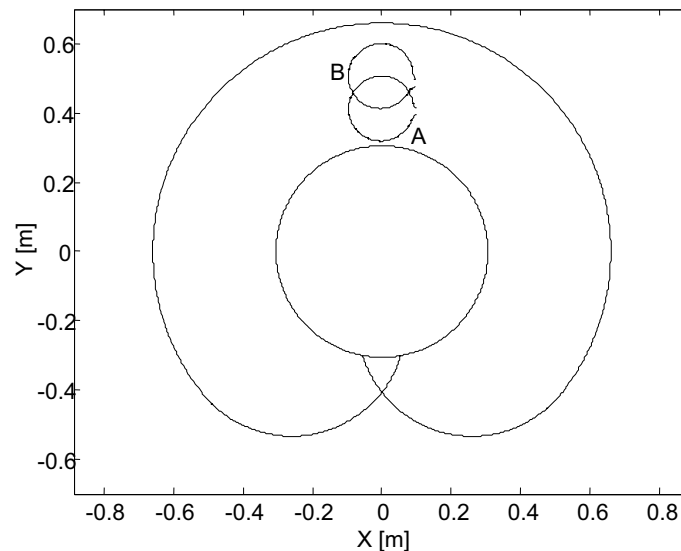
Figure 7. Positions of the workpiece in the manipulator workspace

It is worth noting that the implicit hybrid force/velocity control scheme requires a more significant tuning effort than the explicit one because of the presence of the additional position inner loop.

Results are shown in Figures 8-11. In order to understand the results better, the plots of the obtained normal force and tangential velocity signals for the considered control schemes applied to disk A when the normal force set-point is 30 N and the tangential velocity set-point is 10 mm/s are shown in Figures 12-13.
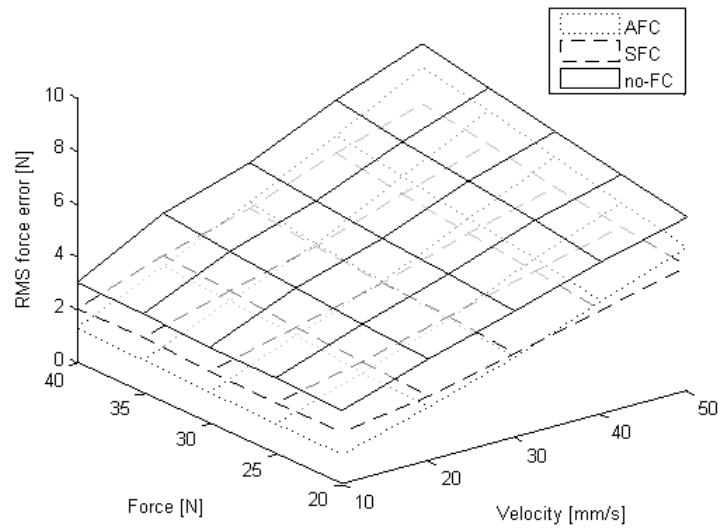
a)
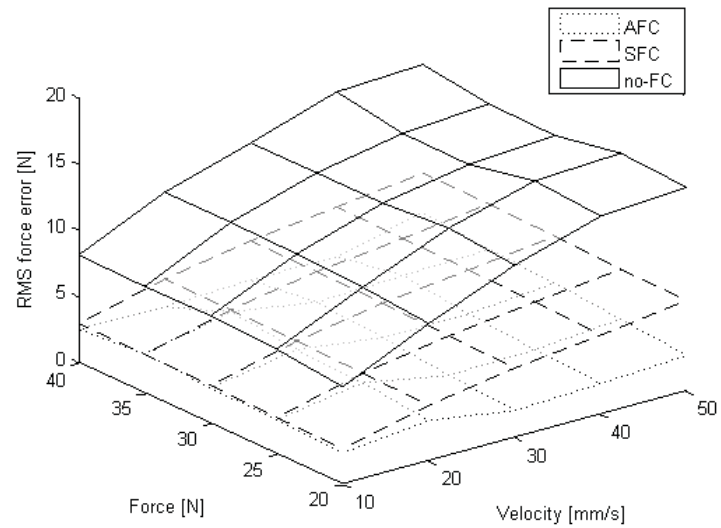
b)

Figure 8. Force RMS error with no friction compensation and with SFC and AFC for the disk in the A position. a) implicit control; b) explicit control.

a)

b)

Figure 9. Force RMS error with no friction compensation and with SFC and AFC for the disk in the B position. a) implicit control; b) explicit control.

a)

b)

Figure 10. Velocity RMS error with no friction compensation and with SFC and AFC for the disk in the A position. a) implicit control; b) explicit control.

(a)

b)

Figure 11. Velocity RMS error with no friction compensation and with SFC and AFC for the disk in the B position. A) implicit control; b) explicit control.
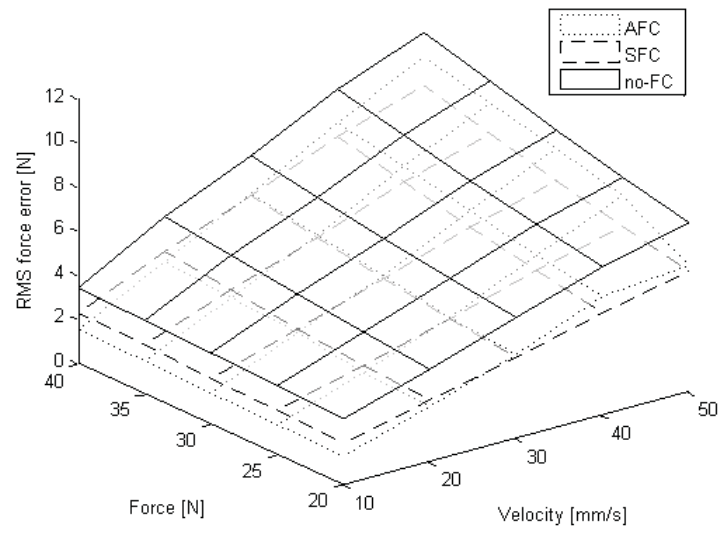
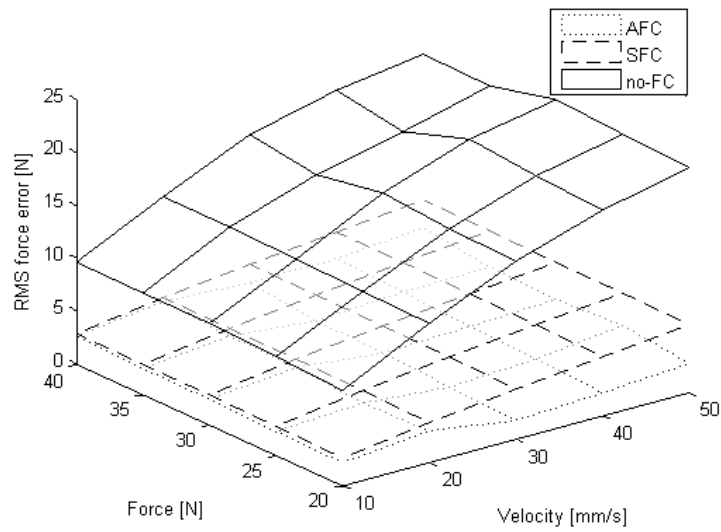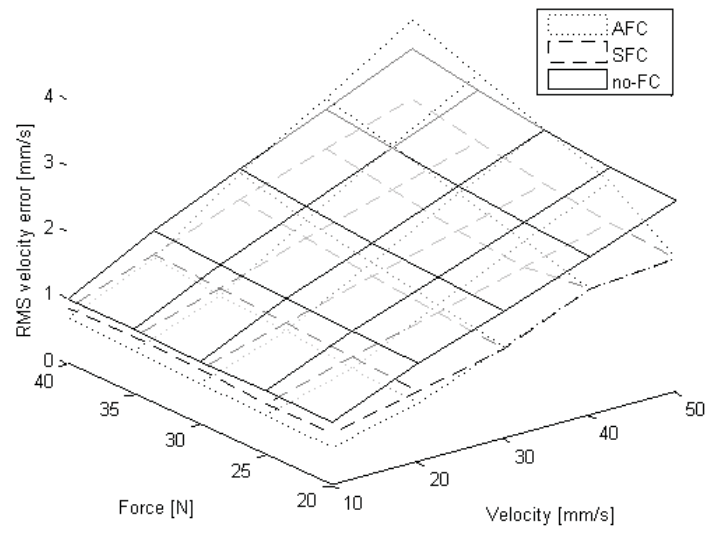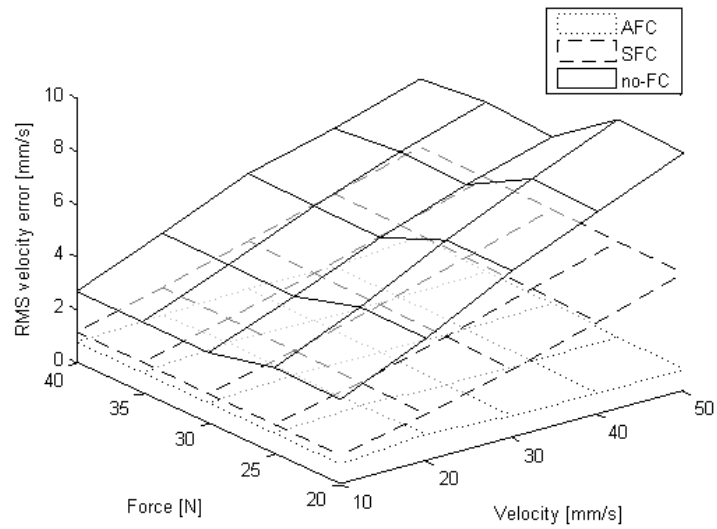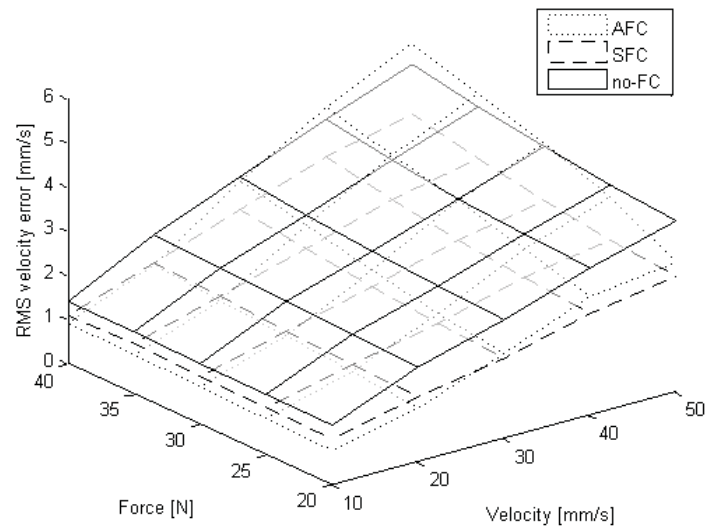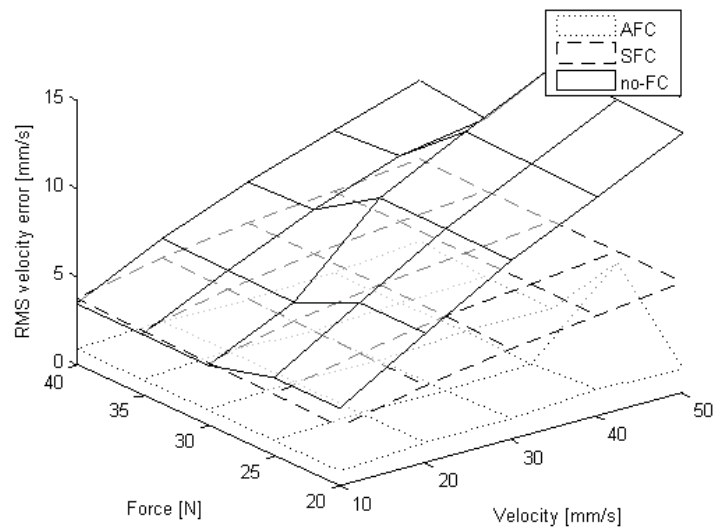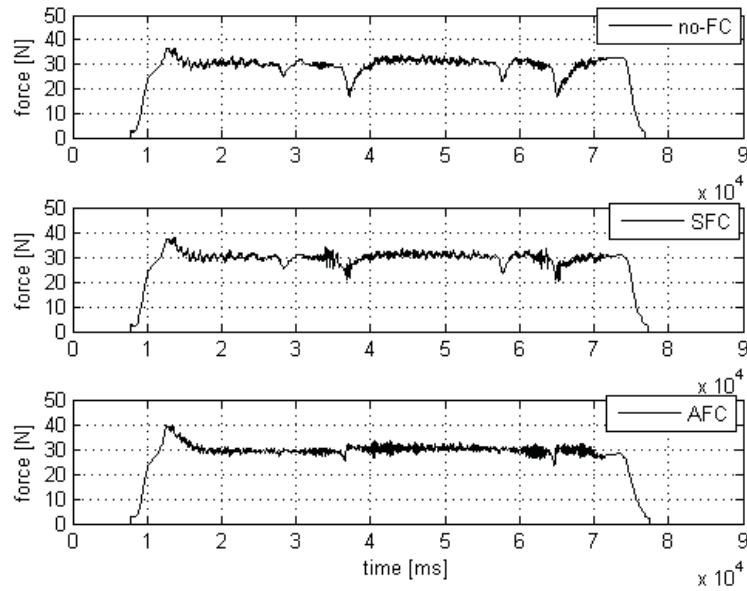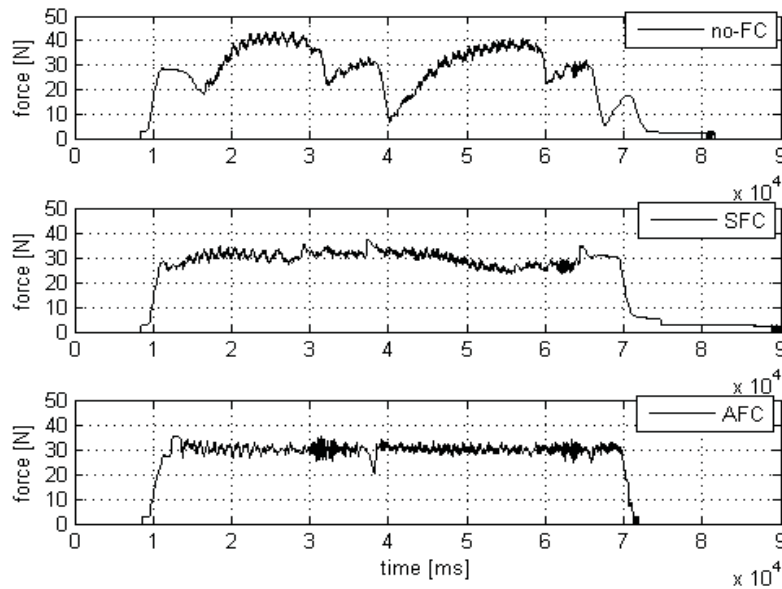Figure 12. Normal force signals with no friction compensation and with SFC and AFC for the disk in the A position. a) implicit control; b) explicit control.
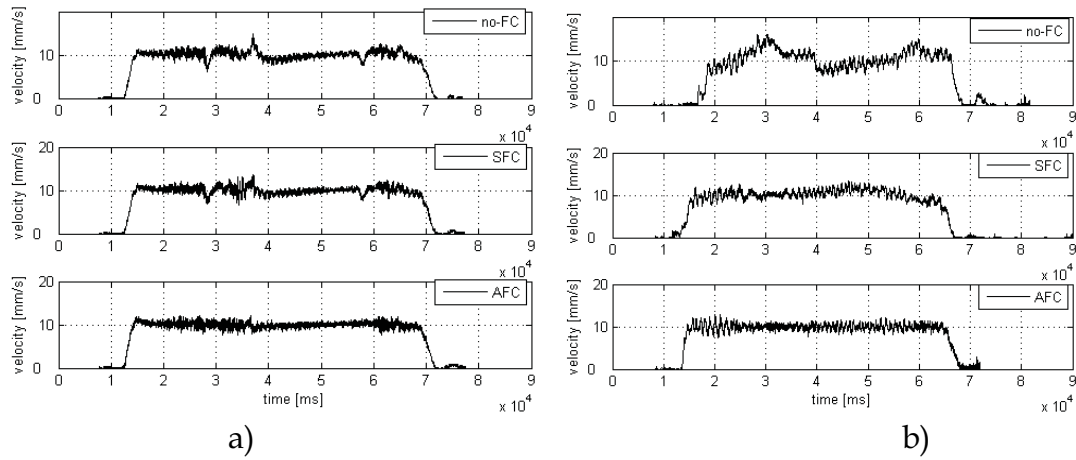
Figure 13. Tangential velocity signals with no friction compensation and with SFC and AFC for the disk in the A position. a) implicit control; b) explicit control.

The mean value of the normal force and tangent velocity RMS error for the different experiments is then reported in Table 1.

From the results presented it can be deduced that a friction compensation strategy is indeed necessary especially for the explicit control law. This is motivated by the fact that the inner joint position control loops in the implicit control law are somehow able to cope with the friction effects. However, it has to be noted again that the implicit control law requires a greater tuning effort than the explicit one (although, from another point of view, it has the advantage that it can be applied to a pre-existing motion control architecture).

The Adaptive Friction Compensation strategy provides definitely the best results for the explicit control scheme both in terms of normal force and tangential velocity, while for the implicit control law the performance obtained by the Adaptive Friction Compensation scheme and by the Static Friction Compensation scheme are similar. In any case, the great advantage for the AFC of being a model-free scheme (*i.e.*, no preliminary experiment is required to derive a friction model and robustness to variation of the friction parameters is assured) makes it more appealing to be applied in a practical context.

It is worth stressing that the AFC strategy is effective in reducing the normal force and tangential velocity errors especially when the joint velocity sign changes. This fact can be evaluated by considering the resulting two joint velocities that would be necessary in order to achieve the required tangential velocity of 10 mm/s (for disk A). They are reported in Figure 14 (compare with Figures 12 and 13, for example at time *t*=3.9 s when the velocity of the first joint changes its sign it appears that the normal force and tangential velocity errors increase significantly when no friction compensation is applied, especially for the explicit control).

Further, the explicit hybrid force/velocity controller (with AFC) provides basically the same performance (in terms of both normal force and tangential velocity errors) disregarding the different normal force and tangential velocity set-points and the different position of the workpiece in the manipulator workspace. This is indeed a remarkable issue that is due to the higher bandwidth provided by the explicit control than the implicit one.

	Normal force [N]				Tangential velocity [mm/s]			
	Position A		Position B		Position A		Position B	
	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit
AFC	3.74	2.97	4.70	2.83	1.8	0.89	2.5	1.3
SFC	3.65	5.16	4.50	4.80	1.5	2.7	2.1	5.1
no-FC	5.26	12.32	6.27	16.05	2.0	5.5	2.9	8.7

Table 1. Mean value of the normal force and tangent velocity RMS error for the different experiments
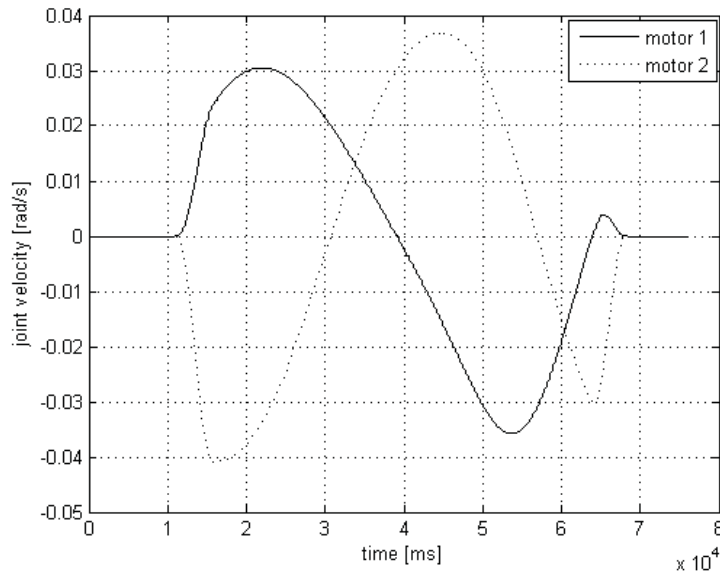

Figure 14. Required joint velocities for tracking disk A with the tangential velocity of 10 mm/s

6. Conclusions

Tasks based on force control are seldom performed by robot manipulators in industrial settings. This might be due to the lack of a thorough characterisation of the methodologies developed theoretically from an industrial point of view. Indeed, it has to be demonstrated that a proposed control strategy can be applied effectively in all the possible situations that might arise in an industrial context and, in general, the cost/benefit ratio should be clearly outlined.

In this chapter the use of hybrid force/velocity control for the contour tracking of an object of unknown shape performed by an industrial robot SCARA manipulator has been discussed. In particular, both the implicit and explicit control laws have been considered and the compensation of the joint friction effect has been addressed.

The pros and cons of the use of an inner joint position control loop have been outlined and it has been shown that the application of a friction compensation strategy is essential if the explicit control law is selected. In this context, the use of the devised Adaptive Friction Compensation strategy is advisable as it provides basically the same (high) performance in the different considered task and its application does not require any previous knowledge of the friction model, that is, no *ad hoc* experiments have to be performed.

8. References

Ahmad, S. & Lee C. N. (1990). Shape recovery from robot contour-tracking with force feedback, *Proceedings of IEEE International Conference on Robotics and Automation,* pp. 447-452, Cincinnati (OH), May 1990

Bona, B.; Indri, M. & Smaldone N. (2003). Nonlinear friction estimation for digital control of direct-drive manipulators. *Proceedings of European Control Conference,* Cambridge (UK), September 2003

Craig, J. J. (1989). *Introduction to Robotics: Mechanics and Control,* Prentice-Hall, 0131236296

Daemi M. & Heimann B. (1996). Identification and compensation of gear friction for modelling of robots. *Proceedings of CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators,* pp. 89-99, Udine (I)

De Schutter J. (1986). Compliant robot motion: task formulation and control. *PhD thesis*, Katholieke Universiteit Leuven

Ferretti, G.; Magnani, G. & Rocco, P. (2000). Triangular force/position control with application to robotic deburring. *Machine Intelligence and Robotic Control,* pp. 83-91

Haykin, S. (1999). *Neural Networks – A Comprehensive Foundation*, Prentice-Hall, 0132733501

Indri, M.; Calafiore, G.; Legnani, G.; Jatta, F. & Visioli, A. (2002). Optimized

dynamic calibration of a SCARA robot. *Preprints of 15th IFAC World Congress on Automatic Control,* Barcelona (E), July 2002

Jatta, F.; Legnani, G.; Visioli, A. & Ziliani, G. (2006). On the use of velocity feedback in hybrid force/velocity control of industrial manipulators. *Control Engineering Practice,* Vol. 14, pp. 1045-1055.

Legnani G.; Adamini R. & Jatta F. (2001). Calibration of a SCARA robot by contour tracking of an object of known geometry, *Proceedings of International Symposium on Robotics,* pp. 510-515, Seoul (ROK), April 2001

Olsson, H.; Åström, K. J.; Canudas de Wit, C.; Gafvert, M. & Lischinsky P. (1998). Friction models and friction compensation. *European Journal of Control,* Vol. 4, pp. 176-195

Raibert, M. H. & Craig, J. J. (1981). Hybrid position/force control of manipulators. *ASME Journal of Dynamic Systems, Measurements, and Control,* Vol. 102, pp. 126-133

Roy, J. & Whitcomb, L. L. (2002). Adaptive force control of position/velocity controlled robots: theory and experiments. *IEEE Transactions on Robotics and Automation,* Vol. 18, pp. 121-137

Siciliano, B. & Villani, L. (1999). *Robot Force Control,* Kluwer Academic Publisher, 0792377338

Thomessen, T. & Lien T. K. (2000). Robot control system for safe and rapid programming of grinding applications, *Proceedings of International Symposium on Robotics,* pp. 18-30, Montreal (C), May 2000

Visioli, A. & Legnani, G. (2002). On the trajectory tracking control of industrial SCARA robot manipulators. *IEEE Transactions on Industrial Electronics,* Vol. 49, pp. 224-232

Visioli, A.; Adamini, R. & Legnani, G. (1999). Adaptive friction compensation for industrial robot control. *Proceedings of ASME/IEEE International Conference on Advanced Intelligent Mechatronics,* pp. 577-582, Como (I), July 1999

Volpe, R. & Khosla, P. (1993). A theoretical and experimental investigation of explicit force control strategies for manipulators. *IEEE Transactions on Automatic Control,* Vol. 38, pp. 1634-1650

Whitcomb L. L.; Arimoto, S.; Naniwa, T. & Ozaki, F. (1997). Adaptive model-based hybrid control of geometrically constrained robot arms. *IEEE Transactions on Robotics and Automation,* Vol. 13, pp. 105-116

Ziliani, G.; Legnani, G. & Visioli, A. (2005). A mechatronic design for robotic deburring, *Proceedings of IEEE International Symposium on Industrial Electronics,* pp. 1575-1580, Dubrovnik (HR), June 2005

Ziliani, G.; Visioli, A. & Legnani, G. (2006). Gain scheduling for hybrid force/velocity control in contour tracking task. *International Journal of Advanced Robotic Systems* Vol. 3, pp. 367-374.

Industrial Robot Control System Parametric Design on the Base of Methods for Uncertain Systems Robustness

Alla A. Nesenchuk and Victor A. Nesenchuk

1. Introduction

Industrial robots often operate in conditions of their parameters substantial variation that causes variation of their control systems characteristic equations coefficients values, thus generating the equations families. Analysis of the dynamic systems characteristic polynomial families stability, the stable polynomials and polynomial families synthesis represent complicated and important task (Polyak, 2002, a). Within the parametric approach to the problem the series of the effective methods for analysis have been developed (Bhattaharyya et al., 1995; Polyak, 2002, a). In this way, V. L. Kharitonov (Kharitonov, 1978) proved that for the interval uncertain polynomials family asymptotic stability verification it is necessary and enough to verify only four polynomials of the family with the definite constant coefficients. In the works of Y. Z. Tsypkin and B. T. Polyak the frequency approach to the polynomially described systems robustness was offered (Polyak & Tsypkin, 1990; Polyak & Scherbakov, 2002; Tsypkin & Polyak, 1990; Tsypkin, 1995). This approach comprises the robust stability criteria for linear continuous systems, the methods for calculating the maximal disturbance swing for the nominal stable system on the base of the Tsypkin – Polyak hodograph. These results were generalized to the linear discrete systems (Tsypkin & Polyak, 1990). The robust stability criterion for the relay control systems with the interval linear part was obtained (Tsypkin, 1995). The super-stable linear systems were considered (Polyak & Scherbakov, 2002).

The problem for calculating the polynomial instability radius on the base of the frequency approach is investigated (Kraev & Fursov, 2004). The technique for composing the stability domain in the space of a single parameter or two parameters of the system with the D-decomposition approach application is developed (Gryazina & Polyak. 2006).

The method for definition of the nominal polynomial coefficients deviations limit values, ensuring the hurwitz stability, has been offered (Barmish, 1984). The task here is reduced to the single-parameter optimization problem. The similar tasks are solved by A. Bartlett (Bartlett et al., 1987) and C. Soh (Soh et

al., 1985). Conditions for the generalized stability of polynomials with the linearly dependent coefficients (polytopes) have been obtained (Bartlett et al., 1987; Rantzer, 1992).

One of the most important stages, while calculating dynamic systems with uncertain parameters, is ensuring robust quality. The control process qualitative characteristics are defined by the characteristic equations roots location in the complex plane (the plane of system fundamental frequencies). In this connection, three main groups of tasks being solved can be distinguished: determining the assured roots location domain (region) for the given system, finding conditions of whether roots get into the given region or not (determination of the Λ-stability conditions) and locating roots in the given domain (ensuring Λ-stability).

The frequency stability criteria for the linear systems families and also the method for finding the largest disturbance range of their characteristic equations coefficients, which guarantees the system asymptotic stability, are considered by B. T. Polyak and Y. Z. Tsypkin (Polyak & Tsypkin, 1990). The assured domain of the interval polynomial roots location is found in (Soh et al., 1985). The root locus theory is used in (Gaivoronsky, 2006) for this task solution. Conditions (Vicino, 1989; Shaw & Jayasuriya, 1993) for the interval polynomial roots getting into the given domain of some convex shape are defined.

The parametric approach to robustness, based on the root locus theory (Rimsky, 1972; Rimsky & Taborovetz, 1978; Nesenchuk, 2002; Nesenchuk, 2005), is considered in this chapter in application to the industrial anthropomorphous robot control system parametric design. The developed techniques allow to set up the values of the parameter variation intervals limits for the cases when the stability verification showed, that the given system was unstable, and to ensure the system robust quality by locating the characteristic equations family roots within the given quality domain.

2. Industrial robot and its control system description

Most industrial robots are used for transportation of various items (parts), e. g. for installing parts and machine tools in the cutting machines adjustments, for moving parts and units, etc. During the robot operation due to some internal or external reasons its parameters vary, causing variation of the system characteristic equation coefficients. This variation can be rather substantial. In such conditions the system is considered, as the uncertain system.

2.1 General description of the anthropomorphous industrial robot

The industrial robot considered here is used for operation as an integrated part of the flexible industrial modules including those for stamping, mechanical as-

sembly, welding, machine cutting, casting production, etc. The industrial robot is shown in fig. 1. It comprises manipulator 1 of anthropomorphous structure, control block 2 including periphery equipment and connecting cables 3. Manipulator has six units (1–8 in fig. 1) and correspondingly is of six degrees of freedom (see fig. 1): column 4 turn, shoulder 5 swing, arm 6 swing, hand 7 swing, turn and rotation. The arm is connected with the joining element 8. Controlling robots of such a type, belonging to the third generation, is based on the hierarchical principle and features the distributed data processing. It is based on application of special control processors for autonomous control by every degree of freedom (lower executive control level) and central processor coordinating their operation (higher tactical control level).

2.2 Industrial robot manipulator unit control system, its structure and mathematical model

Executive control of every manipulator unit is usually executed in coordinates of this unit (Nof, 1989) and is of the positional type. It is the closed-loop servo-control system not depending on the other control levels. Although real unit control is executed by a digital device (microprocessor, controller) in a discrete way, the effect of digitization is usually neglected, as the digitization frequency is high enough to consider the unit and the controller as the analog (continuous) systems. As for the structure, the unit control loops are almost similar and differ only in the parameter values. Therefore, any unit of the industrial robot can be considered for investigating the dynamic properties.
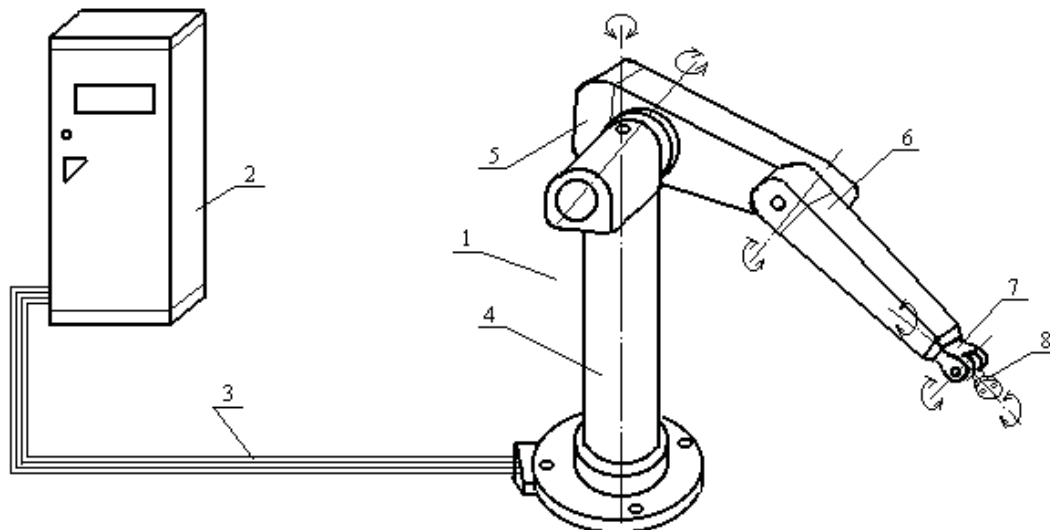
Figure 1. Anthropomorphous industrial robot

The structure of the manipulator unit subordinate control is shown in fig. 2. The simplified version of the structure is presented in fig 3.

In fig. 2 the plant is represented by elements 1–4 (a DC motor); 5 is the sensor transforming the analog speed signal into the speed code (photo-pulse sensor), 6 is the element combining the speed PI regulator, code-pulse width transformer and capacity amplifier, 7 is the transformer of analog position signal into the position code (photo-pulse sensor), 8 is the proportional regulator of the manipulator shoulder position, 9 is the transfer mechanism (reducer). In fig. 3 the transfer function

$$W_p'(s) = W_p(s)s$$

where $W_p(s)$ is the plant transfer function.

Substitute corresponding parameters and express the plant transfer function as follows:

$$W_p(s) = \frac{\varphi}{U_g} = \frac{1}{(j_m + j_l)\dfrac{L_A}{C_M}s^3 + (j_m + j_l)\dfrac{R_A}{C_M}s^2 + C_e s}, \tag{1}$$

where U_g is the input voltage, φ is the object shaft angle of rotation.
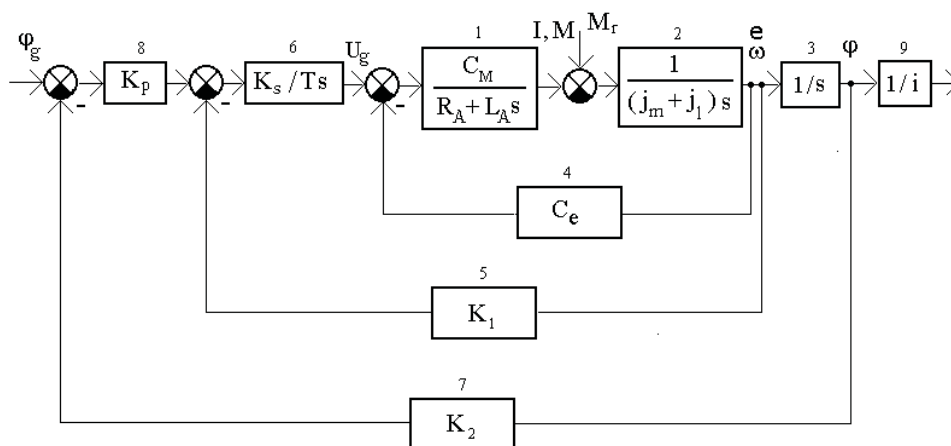
Figure 2. Control system for the industrial robot manipulator shoulder unit

On the basis of (1) write the manipulator unit control system characteristic equation

$$s^4 + \frac{R_A}{L_A}s^3 + \frac{C_e C_M}{j_m L_A}s^2 + \frac{C_M K_1 K_s}{j_m L_A T}s + \frac{C_M K_2 K_p K_s}{j_m L_A T} = 0$$

or as

$$a_0 s^4 + a_1 s^3 + a_2 s^2 + a_3 s + a_4 = 0 , \qquad (2)$$

where

$$a_0 = 1 ; \; a_1 = {R_A}\big/{L_A} ; \; a_2 = \frac{C_e C_M}{(j_m + j_{l'})L_A} ; \; a_3 = \frac{C_M K_1 K_s}{(j_m + j_l)L_A T} ; \; a_4 = \frac{C_M K_2 K_p K_s}{(j_m + j_l)L_A T} ;$$

- R_A is the motor anchor resistance;
- L_A is the anchor inductance;
- j_l is the load inertia moment;
- j_m is the anchor inertia moment;
- C_e is the electric-mechanical ratio of the motor;
- C_M is the constructive constant of the motor;
- T is the time constant of the PI regulator;
- K_1 and K_2 are photo-electric sensor coefficients;
- K_s and K_p are gains of regulators by speed and position correspondingly.

Suppose the robot unit has the following nominal parameters:

- $R_A = 0{,}63\ \Omega$;
- $L_A = 0{,}0014$ henry;
- $j_l = 2{,}04 \cdot 10^{-5} kg / m^2$
- $j_m = 40{,}8 \cdot 10^{-5} kg / m^2$;
- $C_e = 0{,}16\dfrac{V \cdot s}{rad}$;
- $C_M = C_e$;
- $T = 0{,}23\ s$;
- $K_1 = 66{,}7,\ K_2 = 250$;
- $K_s = 0{,}078,\ K_p = 2{,}5$.
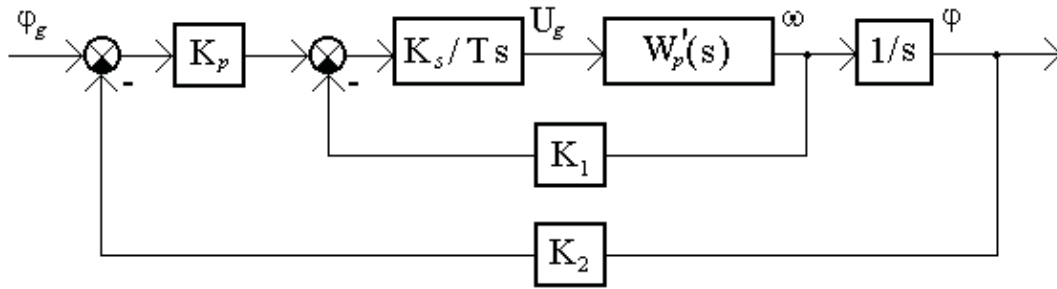
Figure 3. Structure of the position control system loop for the manipulator shoulder unit

After substitution of the nominal values into (2) rewrite the unit characteristic equation as

$$s^4 + 0,5 \cdot 10^3 s^3 + 0,427 \cdot 10^5 s^2 + 0,6 \cdot 10^7 s + 0,56 \cdot 10^8 = 0 \qquad (3)$$

The coefficients of (3) are the nominal ones and while robot operation they often vary within the enough wide intervals. For this reason when calculating the robot control system it is necessary to consider the parameters uncertainty and ensure the control system robustness.

3. The techniques for robust stability of systems with parametric uncertainty

The method is described for synthesis of the interval dynamic system (IDS) stable characteristic polynomials family from the given unstable one, based on the system model in the form of the free root locus portrait. This method allows to set up the given interval polynomial for ensuring its stability in cases, when it was found, that this polynomial was unstable. The distance, measured along the root locus portrait trajectories, is defined as the setting up criterion, in particular, the new polynomial can be selected as the nearest to the given one with consideration of the system quality requirements. The synthesis is carried on by calculating new boundaries of the polynomial constant term variation interval (stability interval), that allows to ensure stability without the system root locus portrait configuration modification

3.1 The task description

While investigating uncertain control systems for getting more complete representation of the processes, which occur in them, it seems substantial to discover correlation between algebraic, frequency and root locus methods of in-

vestigation. Such correlation exists and can be applied for finding dependence between the system characteristic equation coefficients values (parameters) and its dynamic properties to determine how and what coefficients should be changed for ensuring stability. One of the ways for establishing the above mentioned correlation can be investigation of the systems root locus portraits and Kharitonov's polynomials root loci (Kharitonov, 1978).

Consider the IDS, described by the family of characteristic polynomials

$$P(s) = \sum_{j=0}^{n} a_j s^{n-j} = 0,$$ (4)

where $a_j \in [\underline{a}_j, \overline{a}_j]$, $\underline{a}_0 > 0$, $j = 0, \ldots, n$; $\underline{a}_j$ and $\overline{a}_j$ are correspondingly the lower and upper boundaries of the closed interval of uncertainty, $[\underline{a}_j, \overline{a}_j]$; $s = \sigma + i\omega$.

The coefficients of polynomial (4) are in fact the uncertain parameters.

The task consists in synthesis of the stable interval family of polynomials (4) on the basis of the initial (given) unstable one, i. e., when the initial system stability verification by application of Kharitonov's polynomials gave the negative result. Calculation of new parameter variation intervals boundaries is made on the base of the initial boundaries in correspondence with the required dynamic characteristics of the system. The new boundaries values definition criteria can be different, in particular they can be selected the nearest to the given ones. In this case the distance, measured along the system roots trajectories, is accepted to be the criterion of such proximity.

3.2 The interval system model in the form of the root locus portrait

Introduce the series of definitions.

Definition 1.   Name the root locus of the dynamic system characteristic equation (polynomial), *as the dynamic system root locus.*

Definition 2.   Name the family (the set) of the interval dynamic system root loci, *as the root locus portrait of the interval dynamic system.*

Definition 3.   The algebraic equation coefficient or the parameter of the dynamic system, described by this equation, being varied in a definite way for generating the root locus, when it is assumed, that all the rest coefficients (parameters) are constant, name as the algebraic equation root locus free parameter or simply the root locus parameter.

Definition 4.   The root locus, which parameter is the coefficient a_k, name as the algebraic equation *root locus relative to the coefficient a_k.*

Definition 5.   The root locus relative to the dynamic system characteristic equation constant term name as the *free root locus of the dynamic system.*

Definition 6.   The points, where the root locus branches begin and the root locus parameter is equal to zero, name as the *root locus initial points.*

Remark 1.   One of the free root locus initial points is always located at the origin of the roots complex plane.

The above remark correctness follows from the form of equation (4).

Remark 2.   The free root locus positive real branch portion, adjacent to the initial point, located at the origin, is directed along the negative real half-branch σ of the complex plane to the left half-plane.

Remark 2 is correct due to the root loci properties (Uderman, 1972) and because real roots of equations with positive coefficients are always negative (see fig. 4).

The peculiarity of the free root loci, which distinguishes them from another types of root loci, consists in the fact, that all their branches strive to infinity, approaching to the corresponding asymptotes.

For carrying on investigation apply the Teodorchik – Evans free root loci (TEFRL) (Rimsky, 1972), i. e. the term "root locus" within this section will mean the TEFRL, which parameter is the system characteristic equation constant term.

To generate the IDS root locus portrait apply the family of the mapping functions

$$s^n + a_1 s^{n-1} + a_2 s^{n-2} + \ldots + a_{n-2} s^2 + a_{n-1} s = u(\sigma,\omega) + iv(\sigma,\omega) = - a_n, \tag{5}$$

where $u(\sigma,\omega)$ and $v(\sigma,\omega)$ are harmonic functions of two independent variables σ and ω; a_n is the root locus parameter; $s = \sigma + i\omega$. Analytical and graphical root loci are formed using mapping function (5). The root locus equation is as follows:

$$iv(\sigma,\omega) = 0 \tag{6}$$

and the parameter equation (Rimsky, 1972) as follows:

$$u(\sigma,\omega) = - a_n. \tag{7}$$

The fragmentary root locus portrait for the IDS of the forth order, which is made up of four Kharitonov's polynomials free root loci, is shown in fig. 4. The Kharitonov's polynomials h_1, h_2, h_3 and h_4 in this figure are represented by points (roots), marked with circles, triangles, squares and painted over squares correspondingly. There are the following designations: σ_{h_i}, $i = 1, 2, 3, 4$, – the cross centers of asymptotes for the root loci of every polynomial h_i, t_l, $l = 1, 2, 3$, – cross points of the root loci branches with the system asymptotic stability boundary, axis $i\omega$. The root loci initial points, which represent zeroes of mapping function (5), are depicted by X-s. Because in fig. 4 all roots of the Kharitonov's polynomials are completely located in the left half-plane, the given interval system is asymptotically stable (Kharitonov, 1978).
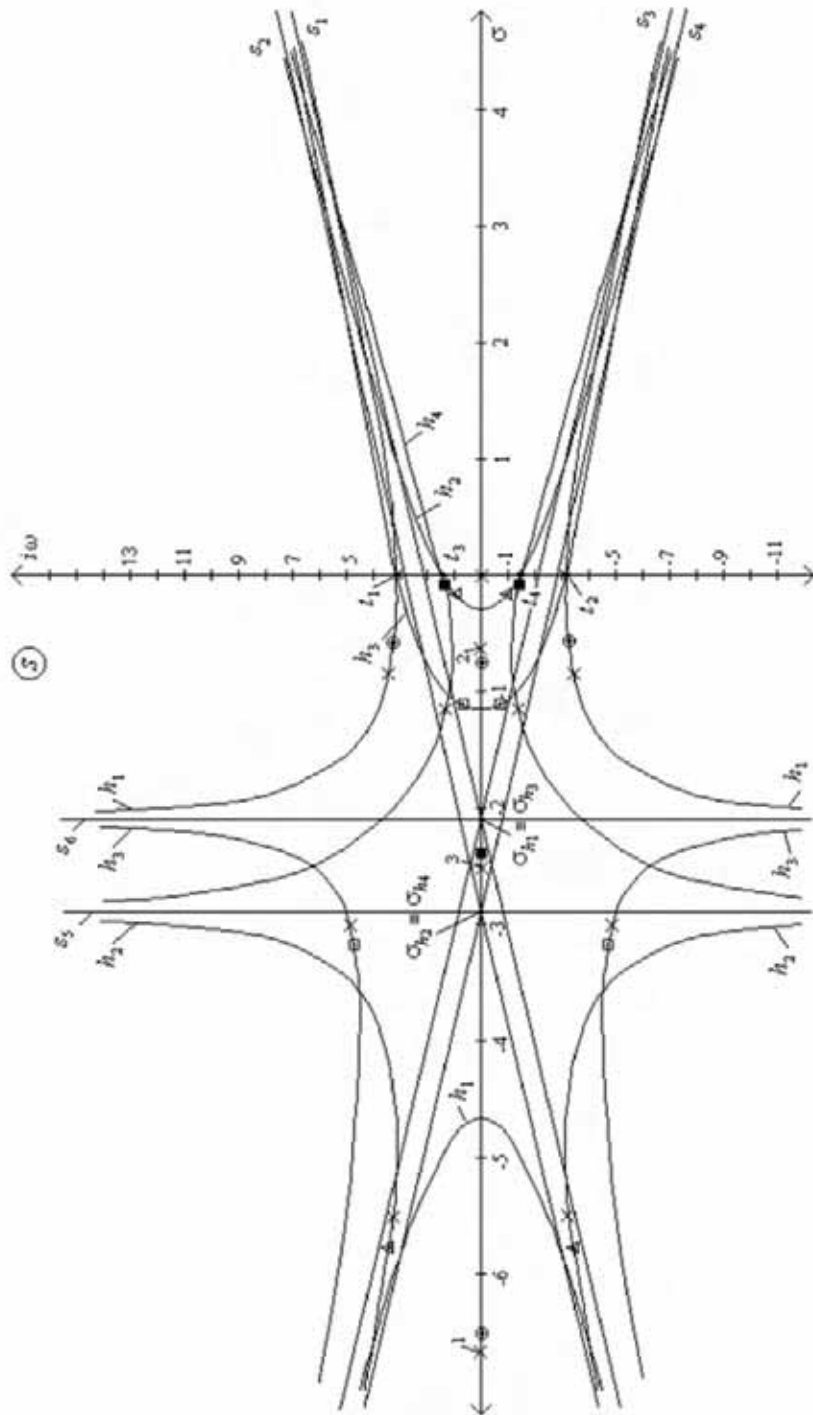
Figure 4. Root loci of the Kharitonov's polynomials for the system of class [4;0]

3.3 Investigation of the characteristic polynomial family root loci branches behavior at the asymptotic stability boundary of the system

The branches of the IDS root locus portrait, when crossing the stability boundary, generate on it the region (set) of cross points. Name this region, as *the cross region* and designate it as R_ω. According to the theory of the complex variable (Lavrentyev & Shabat, 1987) and due to the complex mapping function (5) continuity property, this region is the many-sheeted one and is composed of the separate sheets with every sheet (continuous subregion), formed by the separate branch while it moves in the complex plane following the parameters variation. The cross region portion, generated by only positive branches of the system root locus portrait, name as *the positive cross region* and designate it as R_ω^+.

$$R_\omega^+ \subset R_\omega. \tag{8}$$

Define also the subregion r_ω^+ (either continuous or discrete one) of the cross region R_ω^+ (8) generated by the root loci branches of any arbitrary *subfamily f* of the interval system polynomials family (4), and name it as *the (positive) cross subregion*, thus,

$$r_\omega^+ \subset R_\omega^+. \tag{9}$$

Introduce the following sets:

$$W_r^+ = \left\{\omega_{r\,i}^+\right\} \tag{10}$$

$$A_r^+ = \left\{a_{r\,i}^+\right\} \tag{11}$$

where W_r^+ is the set (family) of the cross subregion r_ω^+ (9) points coordinates $\omega_{r\,i}^+$; A_r^+ is the set (family) of values $a_{r\,i}^+$ of the root locus parameter a_n at the set W_r^+ points.

Define the minimal positive value $a_{r\,\min}^+$ of the root locus parameter within the cross subregion r_ω^+:

$$a_{r\,\min}^+ = \inf A_r^+. \tag{12}$$

Peculiarities of the IDS root loci initial points location make it possible to draw a conclusion about existence of its characteristic equation coefficients variation intervals, ensuring asymptotic stability of the given system.

Statement. If the initial points of the IDS characteristic polynomials arbitrary subfamily f free root loci, excluding points always situated at the origin, are located in the left complex half-plane $s,$ there exists the interval d of the root loci parameter a_n values, ensuring asymptotic stability of the subfamily f.

$$d = (0, a_{r\,\min}^+),$$
(13)

Proof. The subfamily f free root loci generate at the system stability boundary the cross subregion r_ω^+ (9) of cross points, which is formed by the set (10) of the cross points coordinates and corresponding set (11) of the parameters values. If the initial points are located, as it is defined by the statement, on every i-th branch of every polynomial root loci there exist an interval $r_i = (\sigma_{l_i}, 0)$ of roots values (starting from the branch initial point with coordinate σ_{l_i} until the point, where it crosses the stability boundary, axis $i\omega$ of the complex plane), which is completely located in the left half-plane. Therefore, there exists also the appropriate maximum possible common interval d_m (which is common for all the branches) of the root loci parameter a_n values (beginning from zero up to the definite maximum possible value $a_n = a_{r\,m}^+$), corresponding to the values of roots within some interval $r_k = (\sigma_{l_k}, 0)$, which ensures the system stability. Name this interval d_m the *dominating interval* and define it as $d_m = (0, a_{r\,m}^+)$. Designate the roots σ_i coordinates values interval, located on every positive i-th branch of the family and corresponding to the dominating interval, as $r_d = (\sigma_{l_i}, \sigma_{r_i})$. It is evident, that $a_{r\,m}^+$ will be maximum possible at the stability boundary, i. e. at $\sigma_{r_i} = 0$. Then, $\forall \sigma_{r_i} [a_{r\,m}^+ = a_{r\,\min}^+ \to \sigma_{r_i} \le 0]$, i. e. the dominating one is the interval $d_m = (0, a_{r\,\min}^+)$, which represents itself the interval d (13). Hence, the statement is correct.

Definition 7.   The interval of polynomial (4) root loci parameter values name *the polynomial stability interval by this parameter* or simply *the polynomial stability interval*, if the polynomial asymptotic stability property holds within this interval.

In case, if some initial points are located at the stability boundary (excluding the point, which is always located at the origin), and on the assumption, that all the rest points are located in the left half-plane, the additional analysis is required for finding the stability interval existence. For this purpose it is necessary to define the root loci branches direction at their outcome from the initial

points, located at the stability boundary, i. e. just to determine what half-plane they are directed to: left one or right one. Obviously, such stability interval exists in the following cases:

a) all the root loci branches with initial points, located at the stability boundary, are directed from these points to the left half-plane;
b) all positive root loci branches with initial points, located at the stability boundary, are directed from these points to the left half-plane.

To determine the above indicated branches direction at the initial points, it is enough to define the root locus sensitivity vector (Nesenchuk, 2005) direction at them.

As a result of the IDS root locus portraits analysis several general regularities have been discovered, being inherent in Kharitonov's polynomials free root loci: paired convergence of the root loci branches at the complex plane imaginary axis (points t_1, t_2, t_3, t_4 in fig. 4); paired convergence of the corresponding asymptotes at the real axis of the complex plane (points σ_{h1}, σ_{h2}, σ_{h3}, σ_{h4} in fig. 4); the tendency for the system robust properties variation while varying its characteristic polynomial coefficients values. It gives the possibility to fix the fact of existence of the system characteristic equation coefficients variation intervals, ensuring its robust stability and also to determine how the coefficients values should be changed for the system dynamic characteristics correction, if it is unstable.

The IDS root locus portraits investigation, which has been carried out, confirms that they can be successfully applied for the in-depth studying robust properties of these systems.

3.4 Parametric synthesis of stable uncertain systems

The conditions for existence of the polynomials (4) family coefficients stability intervals were formulated in the previous section. Here we define what these intervals values should be. For this purpose consider the polynomials (4) subfamily f, consisting of the system Kharitonov's polynomials, and develop the procedure for synthesis of the stable Kharitonov's polynomials on the base of the unstable ones, which depends on the root loci initial points location in relation to the asymptotic stability boundary. For the synthesis procedure development apply the Kharitonov's polynomials free root loci. Consider the case, when initial points are located in the left half-plane. In this case the algorithm of synthesis can be divided into the following stages.

Stage 1. Obtaining the Teodorchik – Evans free root loci equation (6) for each one of the IDS four Kharitonov's polynomials.

As the Kharitonov's polynomials represent the subfamily of the IDS polynomials family, they generate the above described cross subregion $r_\omega{}^+$ (9) on the stability boundary, which is formed by the set (10) of the cross points coordinates.

Stage 2. Calculating coordinates $\omega^+_{r\,i}$ of the set (10) by solution of the TEFRL equations, obtained in stage 1, relative to ω in condition, that $\sigma = 0$. In this way the set $W_r{}^+$ (10) is formed.

For every obtained value of $\omega^+_{r\,i}$ from $W_r{}^+$ the corresponding value of the variable coefficient a_n is calculated by formula (7), thus, forming the set $A_r{}^+$ (11).

Stage 3. Definition of the stability interval by the coefficient a_n.

For this purpose, using (12), define the minimal one, $a^+_{r\,\min}$, of the parameter values at points of the set $A_r{}^+$. Thus obtain the interval d (13) of the parameter a_n variation, which ensures stability of the Kharitonov's polynomials and, therefore, the system in whole.

Before describing the next stage of synthesis formulate the following theorem.

Theorem. For robust stability of the polynomial family (4) it is necessary and enough to ensure the upper limit of the constant term a_n variation interval to satisfy the inequality

$$\overline{a_n} < a^+_{r\,\min}, \tag{14}$$

if the family is stable at $a_n = 0$.

Proof. Let the coefficient a_n to be the polynomial (4) root locus parameter. Under the theorem condition family of (4) is stable at $a_n = 0$, i.e. the root loci initial points are located in the left half-plane. Therefore, in view of statement 1 the theorem is valid.

Stage 4. Comparing the obtained stability interval (13) with the given interval $a_n \in [\underline{a_n}, \overline{a_n}]$ of the parameter a_n variation in correspondence with inequality (14).

In case, if condition (14) is not satisfied, the upper limit $\overline{a_n}$ of the parameter variation interval is set up in correspondence with this inequality.

When the power n of the polynomial is less or equal than 3, $n \leq 3$, the above given theorem is applied without any conditions, i. e. it is not required to sat-

isfy condition of the Kharitonov's polynomials roots real parts negativity at $a_n = 0$, because in this case the coefficients positivity always guarantees negativity of the roots real parts.

The above described algorithm allows to carry on the parametric synthesis of the stable interval system without modification of its root locus portrait configuration, by simple procedure of setting up the characteristic polynomial constant term variation interval limits.

The *numerical example* , demonstrating the results obtained, is given below

Consider the interval system, described by the initial characteristic polynomial

$$s^4 + 10s^3 + 35s^2 + 50s + 24 = 0, \tag{15}$$

where the real coefficients are: $a_0 = 1$; $8{,}4 \le a_1 \le 11{,}6$; $24 \le a_2 \le 48$; $26{,}5 \le a_3 \le 83{,}1$; $8{,}99 \le a_4 \le 50{,}3$.

Let the coefficient a_4 to be the root locus parameter. Then, define the mapping function:

$$-a_4 = a_0\sigma^4 + 4a_0\sigma^3 i\omega - 6a_0\sigma^2\omega^2 - 4a_0\sigma i\omega^3 + a_0\omega^4 + a_1\sigma^3 + 3a_1\sigma^2 i\omega -$$
$$- 3a_1\delta\sigma^2 - a_1 i\omega^3 + a_2\sigma^2 + 2a_2\delta i\sigma - a_2\omega^2 + a_3\sigma + a_3 i\omega.$$

Write correspondingly the TEFRL and the parameter equations::

$$\omega(4a_0\sigma^3 - 4a_0\sigma\omega^2 + 3a_1\sigma^2 - a_1\omega^2 + 2a_2\sigma + a_3) = 0;$$
$$a_0\sigma^4 - 6a_0\sigma^2\omega^2 + a_0\omega^4 + a_1\sigma^3 - 3a_1\sigma\omega^2 + a_2\sigma^2 - a_2\sigma = -a_4.$$

Define the Kharitonov's polynomials for the interval system with the initial characteristic polynomial (15):

$$h_1(s) = s^4 + 8{,}4s^3 + 24s^2 + 83{,}1s + 50{,}3;$$
$$h_1(s) = s^4 + 11{,}6s^3 + 48s^2 + 26{,}5s + 8{,}99;$$
$$h_1(s) = s^4 + 8{,}4s^3 + 48s^2 + 83{,}1s + 8{,}99;$$
$$h_1(s) = s^4 + 11{,}6s^3 + 24s^2 + 26{,}5s + 50{,}3.$$

The root loci of these polynomials are represented in fig. 4, described above. Number of asymptotes n_a (in fig. 4 they are indicated as s_1, s_2, …, s_6) is constant for every one of Kharitonov's polynomials and is equal to

$$n_a = n - m = 4 - 0 = 4,$$

where m is the number of poles for function (5).

The centers of asymptotes are located on the axis σ and have coordinates: σ_{h1} = 2,10; σ_{h2} = 2,90; σ_{h3} = 2,10; σ_{h4} = 2,90 (see fig. 4). The asymptotes centers coordinates coincide in pairs: for the pair $h_1(s)$ and $h_3(s)$, and also for the pair $h_2(s)$ and $h_4(s)$.

The inclination angles of asymptotes for the given root loci are correspondingly the following:

$$\varphi_1 = 0^0; \quad \varphi_3 = 135^0;$$
$$\varphi_2 = 45^0; \quad \varphi_4 = 180^0.$$

According to fig. 4, every pair of the root loci strives to the same asymptotes, i.e. the pairs are formed by those root loci, which asymptotes centers coincide, as it was indicated above.

For definition of equation (15) coefficients intervals, ensuring the system stability, stability condition (14) is applied. Thus, the following values $a_{r\,i}^+$ of the set A_r^+ have been defined:

$a_{r\,1}^+$ = 139,67 for the polynomial h_1;

$a_{r\,2}^+$ = 116,33 for the polynomial h_2;

$a_{r\,3}^+$ = 377,75 for the polynomial h_3;

$a_{r\,4}^+$ = 54,89 for the polynomial h_4.

The minimal value is

$$a_{r\ \min}^+ = a_{r\ 4}^+ = 54,89.$$

Because $\bar{a}_4 < 54,89$, in correspondence with (14) the given interval system is asymptotically stable.

4. The method for ensuring uncertain systems quality

In this section the task is solved for locating the uncertain system roots within the trapezoidal domain. The method allows to locate roots of the uncertain system characteristic equations family within the given quality domain, thus ensuring the required system quality (generalized stability). The task is solved by inscribing the system circular root locus field into the given quality domain. The trapezoidal domain, bounded by the arbitrary algebraic curve, is considered. Peculiarity of the method consists in the root locus fields application.

The systems with parametric uncertainty are considered, described by the family of characteristic polynomials

$$p(s) = s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n \tag{16}$$

where $a_1, \ldots, a_n$ are coefficients, which depend linearly of some uncertain parameter k, and can be either real or complex ones.

For selection of the uncertain parameter k, transform equation (16) and rewrite it in the following form:

$$\phi(s) + k\psi(s) = 0 \tag{17}$$

where $\phi(s)$ and $\psi(s)$ are some polynomials of the complex variable s; k is the system uncertain parameter.

Based on (17), derive the expression for k in the form

$$k = f(s) = -\frac{\varphi(s)}{\psi(s)} = u(\sigma, \omega) + iv(\sigma, \omega) \tag{18}$$

where $u(\sigma, \omega)$, $v(\sigma, \omega)$ are harmonic functions of two independent real variables σ and ω.

Consider some provisions about the root locus fields.

Definition 8.   *The root locus field of the control system* is the field with the complex potential

$$\varphi(s) = u(\sigma, \omega) + iv(\sigma, \omega),$$

that is defined in every point of the extended free parameter complex plane by setting the *root locus image* existence over the whole plane (Rimsky & Taborovetz, 1978).

Then, set the root locus image by the real function $h = h(u, v, t)$, where t is the constant value for every image. Name t, as *the image parameter.* Suppose the image is defined over the whole free parameter plane by setting the corresponding boundaries of the parameter t. Thus, using mapping function (18), define in the general form the scalar root locus field function

$$f^* = f^*(\sigma, \omega) \tag{19}$$

and the root locus field level lines equation

$$f^*(\sigma,\omega) = L, \tag{20}$$

where $L = const = t_j$, t_j is the parameter of the j-th image, $-\infty \le t_j \le +\infty$, $j = 1, 2, 3, \ldots$

4.1 The task formulation

Define the quality domain Q (fig. 5) in the left complex half-plane of the system fundamental frequencies (roots plane), bounding the equation (16) roots location by the lines $L_\eta{}'$ and $L_\eta{}''$ of the equal degree of stability (stability margin) and the lines $L_{+\beta}$ and $L_{-\beta}$ of constant damping, that is equivalent to setting permissible limits for the following system quality indicators: degree of the system stability η and oscillation β. In fig. 5 the quality domain Q has the shape of a trapezoid.

The task consists in locating the characteristic equation (16) roots within the domain Q, i. e. in determination of such a domain D of the uncertain parameter k values, which ensure location of this equation roots (e. g., p_1, p_2, p_3, p_4 in fig. 5) within the given domain Q, when the system qualitative characteristics do not get beyond the preset limits for η and β, ensuring thus the system Q-stability and fulfillment of the condition. bounded by the lines of equal degree of stability and constant damping

$$k \in D \rightarrow s_i \in Q, \tag{21}$$

where $i = 1, 2, 3, \ldots, n$.

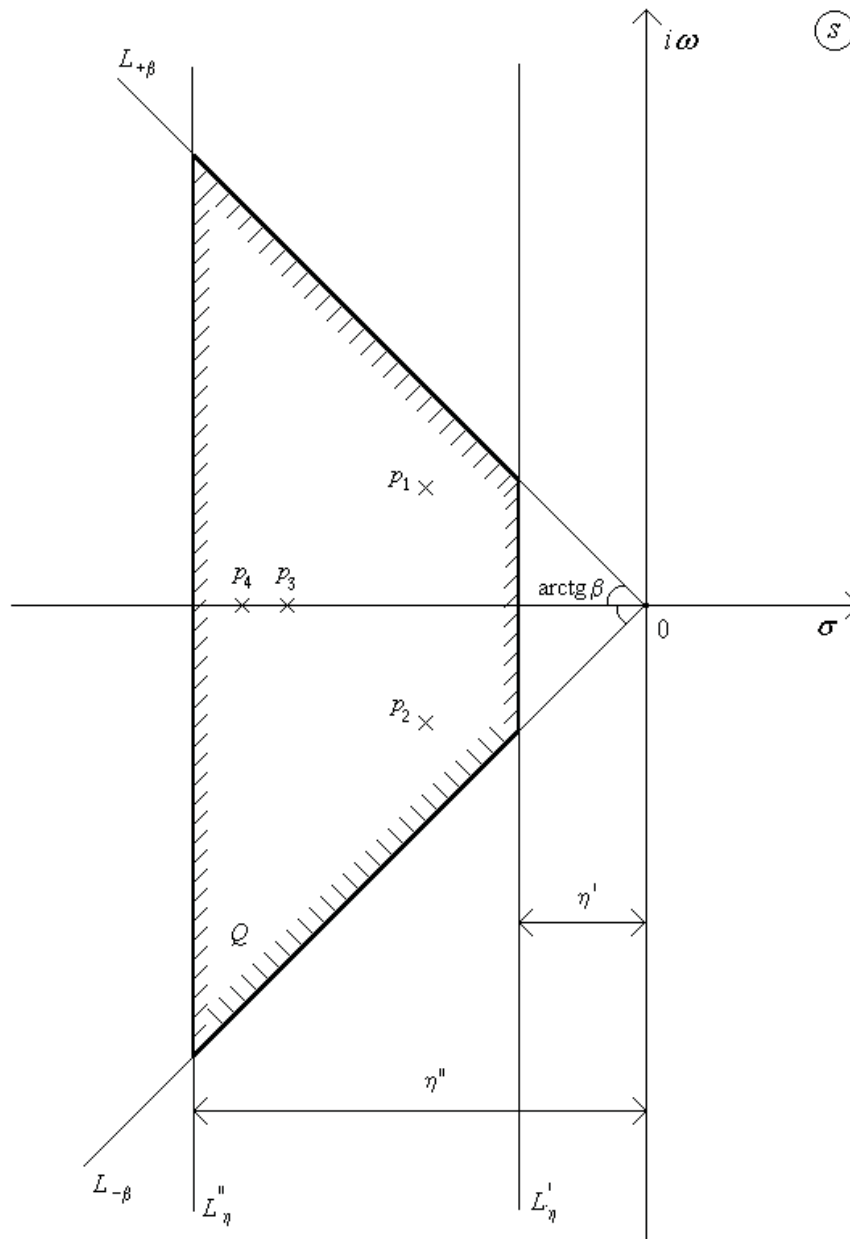Figure 5. The domain Q of the desired characteristic equation roots location, bounded by the lines of equal degree of stability and constant damping.

For solving the task, apply the root locus fields of the circular image (circular root locus fields – CRLF) (Rimsky, 1972; Nesenchuk, 2005).

The field function (19) and the level lines equation (20) for the CRLF in the general form:

$$f^* = f^*(\sigma, \omega, a, b) \tag{22}$$

$$f^*(\sigma, \omega, a, b) = \rho^2. \tag{23}$$

where a and b are the image center coordinates by axes u and v correspondingly, a = const and b = const; ρ is the circular image radius.

The circular root locus fields for the systems of class [3;0] are represented in fig. 6 and 7.

The CRLF location in the complex plane to the great extent is defined by the given circular image center location, which is mapped onto the complex plane by the field localization centers (see definition 2.4 in (Nesenchuk, 2005)).

Localization centers of the field, described by the level lines L_1 (L_1', L_1'', L_1'''), L_2, L_3, L_4, are located in the points C_1, C_2, C_3 (fig. 6, b). The level lines bound the corresponding domains (W_1, W_2, W_3, W_4 in fig. 6, b) in the plane s. Every such many-sheeted domain W represents the mapping of the root locus level line disk-image of the certain radius.

4.2 Locating roots in the given domain

The given task is solved by inscribing the level line of the CRLF, previously oriented in a special way in the complex plane, into the given quality domain of the system. This level line image in the free parameter plane k, that represents some circle of the radius r, will be the boundary of the required domain D (the required disk). Then, in case, if the circular image center is located in the origin, the following condition should be satisfied: $|k| \le r$.

The field orientation

For realization of the above indicated task solution algorithm, at first it is necessary to set orientation (location) of the scalar CRLF in relation to the system quality domain in such a way to ensure the possibility of the field level lines inscription into this domain. Assume the circular image center is located on the positive real axis u, including the origin. The desired location of the circular field is attained, when all its localization centers (i. e. the points, which represent mappings of the circular image center onto the complex plane s) are located inside the quality domain. The enough condition for ensuring such orientation of the field localization centers is location of function (18) zeroes within this domain.

As it was initially assumed, that the circular image center was located on the real axis, the localization centers can be set in two ways:

- in zeroes of function (18), i. e. in poles of the open-loop system transfer function;
- on the branches of the investigated control system Teodorchik – Evans root locus (TERL).

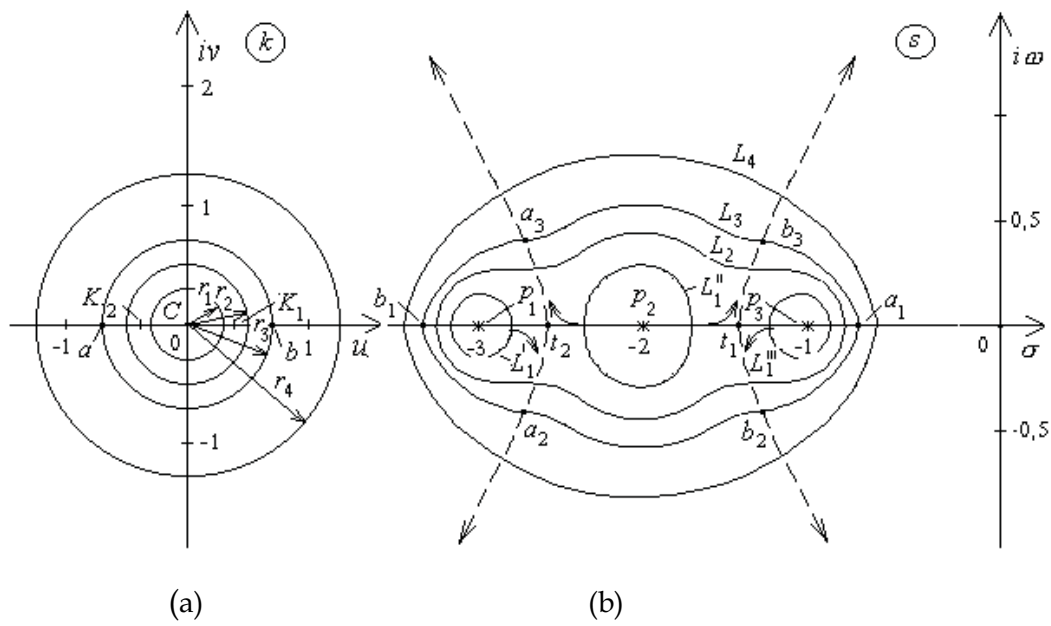(a)   (b)

Figure 6. Circular root locus field when setting the image center in the origin of the variable parameter plane k

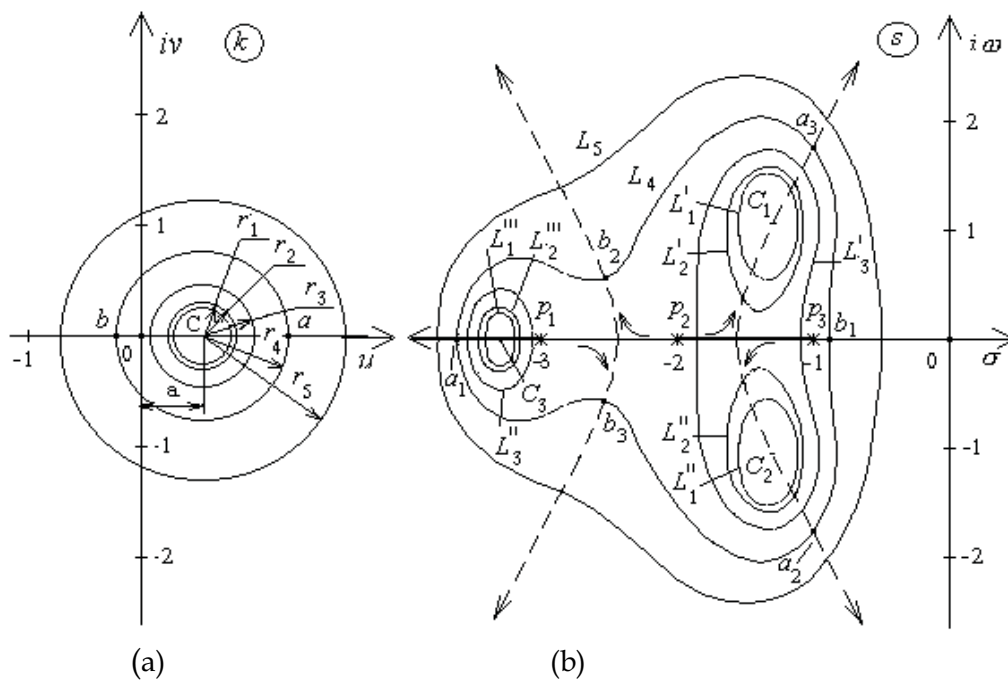(a)   (b)

Figure 7. Circular root locus field when shifting the image center in relation to the origin of the variable parameter plane k

In the first case the circular image center will be located in point C, where $k = 0$ (fig. 6, a). In the second case the field localization centers should be located on the TERL positive branches segments being completely located within the given quality domain. Coordinates $u = a$ and $v = b$ (fig. 7, a) of the corresponding image center are determined from formula (18).

The level lines inscription

After setting the field localization centers it is possible to start inscription of its level lines into the given quality domain. The inscription procedure consists in finding such a level line, which completely belongs to the given quality domain and which represents itself the mapping of the circular image with the maximal possible radius, that evidently will guarantee the required Q-stability of the family (16).

Conditionally divide the task into two subtasks of the level line inscription into the domain, bounded only by:

- the vertical lines of equal degree of stability;
- the inclined lines of constant damping.

Consider the first subtask. For its solution find the extreme points of contact of the CRLF level line and the lines L_η', L_η'' of equal degree of stability (fig. 5). Apply the formula for the gradient of the root locus field:

$$grad f^* = \frac{\partial f^*}{\partial \sigma}\vec{i} + \frac{\partial f^*}{\partial \omega}\vec{j}, \tag{24}$$

where $f^*(\sigma,\omega)$ is the field function; $\vec{i}, \vec{j}$ are projections of the identity vector, directed along the normal to the field level line, onto the axes σ and ω correspondingly.

Because in the desired points of contact the gradient (24) projections onto the axis $i\omega$ are equal to zero, determine these points coordinates by composing two systems of equations:

$$\left.\begin{array}{l} \dfrac{\partial f^*(\sigma,\omega)}{\partial \omega} = 0 \\[2mm] \qquad \sigma = \sigma_\eta' \end{array}\right\}; \tag{25}$$

$$\left.\begin{array}{l} \dfrac{\partial f^*(\sigma,\omega)}{\partial \omega} = 0 \\[2mm] \qquad \sigma = \sigma_\eta'' \end{array}\right\}, \tag{26}$$

where the first equation of every system represents projection of the gradient onto the axis ω; $\sigma_\eta{}'$ and $\sigma_\eta{}''$ are coordinates of cross points of the axis σ and the lines $L_\eta{}'$ and $L_\eta{}'$ correspondingly. From the first system of equations the coordinate ω of the extreme contact point of the line $L_\eta{}'$, bounding the quality domain from the right side, and the CRLF level line is determined. The second system allows to determine the coordinate ω of the extreme contact point (e. g., point t_3 in fig. 8) of the line $L_\eta{}''$, bounding the domain Q on the left side, and the CRLF level line.

Turn to the second subtask consideration. For its solution it is necessary to find the extreme contact point (points) of the CRLF level line and the line $L_{+\beta}$ or $L_{-\beta}$ (fig. 5) of constant damping. The only one line, $L_{+\beta}$ or $L_{-\beta}$, is chosen because when the image center is set on the axis u of the free parameter plane, the CRLF is symmetric in relation to the axis $i\omega$. The line $L_{+\beta}$ will be considered as a tangent to the CRLF level line.
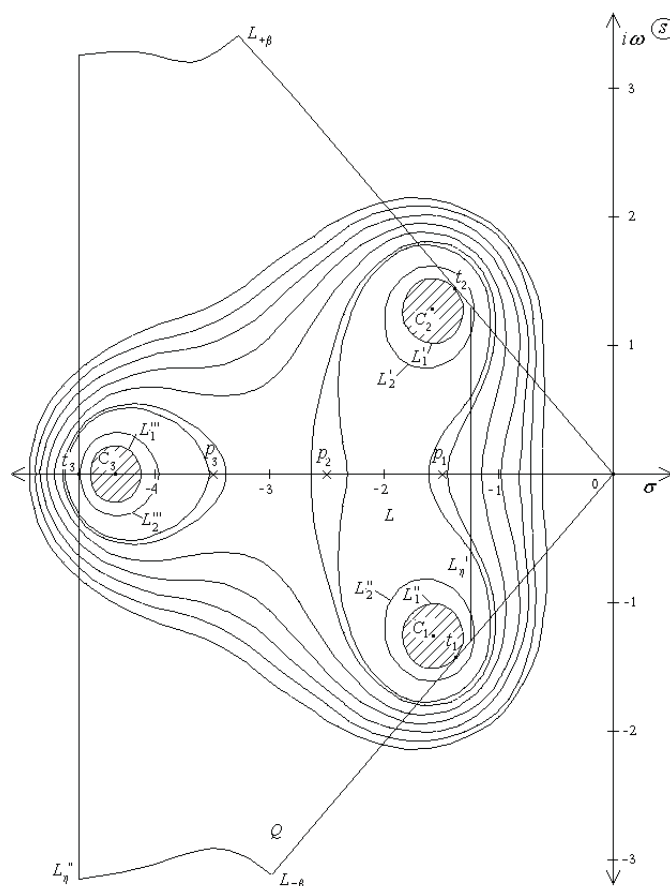

Figure 8. The domain of roots location, inscribed into the given quality domain

Write the equation of a tangent to the scalar CRLF level line (a tangent to the curve) in the general form:

$$\frac{\partial f *(\sigma, \omega)}{\partial \sigma}(\Delta - \sigma) + \frac{\partial f *(\sigma, \omega)}{\partial \omega}(\Omega - \omega) = 0, \tag{27}$$

where Δ, Ω are current coordinates of a point on the tangent; σ, ω are the point of contact coordinates.

As in this case the tangent to the level line passes through the origin, set coordinates Δ and Ω to zero and rewrite (27) in the following form:

$$\frac{\partial f *(\sigma, \omega)}{\partial \sigma}(-\sigma) + \frac{\partial f *(\sigma, \omega)}{\partial \omega}(-\omega) = 0. \tag{28}$$

On the other hand, the equation of the level line $L_{+\beta}$ is

$$\omega = \mu\sigma,$$

where μ is the real constant, $\mu = \text{tg } \beta$ (fig. 5), β is the angle between the constant damping line and the axis ω. By composing on the basis of the last two equations the system

$$\left.\begin{array}{c} \dfrac{\partial f *(\sigma, \omega)}{\partial \delta}(-\delta) + \dfrac{\partial f *(\sigma, \omega)}{\partial \omega}(-\omega) = 0 \\[2mm] \omega = \mu\sigma \end{array}\right\} \tag{29}$$

and solving (29), obtain coordinates σ and ω of the desired contact point.

It is necessary to note, that when solving both the first and the second subtasks, several points of contact to every quality domain boundary can be found. It is explained by the fact, that contact points are determined for both global and every local field level line. In this case the level line corresponding to the circular image of the minimal radius is always chosen. Thus, from three points t_1, t_2 and t_3 (fig. 8), found by the above described method, the point t_1 located on the level line L_1, corresponding to the circular image of the minimal radius, is chosen. This line represents itself the boundary of the desired domain D of the uncertain parameter k values, ensuring the required system operational quality indicators.

Consider *the numerical example*. The system quality domain Q (see fig. 5) is bounded by the lines of equal degree of stability, described by equations

$$\sigma = -1.2, \sigma = -4.7,$$

and the lines of constant damping with equations

$\sigma = \omega$, $\sigma = -\omega$.

Set the characteristic equation, describing the dynamic system of class [3;0] and select polynomials $\phi(s)$ and $\psi(s)$ (see (17)):

$$\phi(s) = s^3 + 7{,}5s^2 + 17{,}8s + 13{,}1; \tag{30}$$

$$\psi(s) = 1. \tag{31}$$

Suppose, that the polynomial constant term a_n is the uncertain parameter. It is required to determine the domain of the perturbed coefficient a_n values, belonging to the given quality domain Q.

Evidently, the poles $p_1 = -1.5$, $p_2 = -2.5$ and $p_3 = -3.5$ (in fig. 8 are marked by X-s) of the open loop transfer function are located inside the quality domain Q. Define the circular root locus field by setting the root locus image existence region over the whole plane of the free parameter a_n. For this purpose set the circular field location by defining circular image center in the point C with coordinates $a = 5$, $b = 0$ (fig. 7, a) in the free parameter plane a_n. Then, its localization centers are located in points C_1, C_2 and C_3 on the branches of the system Teodorchik – Evans root locus, as shown in fig. 7, b.

Calculations were carried on with application of the computer program for ensuring the required quality of control systems with parametric uncertainty, developed for the above described method realization. Polynomials (30), (31) and the domain Q boundaries equations were entered as the input data. The following results have been obtained.

The circular image root locus equation for the given system:

$$\omega^6 + 3\sigma^2\omega^4 + 15\sigma\omega^4 + 20{,}8\omega^4 + 3\omega^2\sigma^4 + 30\omega^2\sigma^3 + 113\omega^2\sigma^2 + 158\omega^2\sigma +$$
$$+ 4{,}32\omega^2 + \sigma^6 + 15\sigma^5 + 91{,}8\sigma + 303\sigma^3 + 587\sigma^2 + 643\sigma + 304 = 0.$$

The CRLF function, applied for the system investigation:

$$f*(\sigma,\omega) = \omega^6 + 3\omega^4\sigma^2 + 15\omega^4\sigma + 20{,}7\omega^4 + 3\omega^2 + 30\omega^2\sigma^3 + 113\omega^2\sigma^2 +$$
$$+ 158\omega^2\sigma + 45{,}3\omega^2 + \sigma^6 + 15{,}2\sigma^5 + 91{,}9\sigma^4 + 303\sigma^3 + 588\sigma^2 + 644\sigma + 328.$$

For determination of the CRLF level line, inscribed into the quality domain, the following systems of equations (25), (26) and (29) were solved:

$$6\omega^4 + 12\omega^2\sigma^2 + 60\omega^2\sigma + 82{,}8\omega^2 + 6\sigma^4 + 60\sigma^3 + 226\sigma^2 + 316\sigma + 90{,}6 = 0 \\ \sigma = -1{,}28 \Bigg\} ;$$

$$6\omega^4 + 12\omega^2\sigma^2 + 60\omega^2\sigma + 82{,}8\omega^2 + 6\sigma^4 + 60\sigma^3 + 226\sigma^2 + 316\sigma + 90{,}6 = 0 \\ \sigma = -4{,}68 \Bigg\} ;$$

$$(6\omega^4\sigma + 15\omega^4 + 12\omega^2\sigma^3 + 90\omega^2\sigma^2 + 226\omega^2\sigma + 158\omega^2 + 6\sigma^5 + 75\sigma^4 + \\ + 368\sigma^3 + 909\sigma^2 + 1180\sigma + 644)(-\sigma) + (6\omega^4 + 12\omega^2\sigma^2 + 60\omega^2\sigma + \\ + 82{,}8\omega^2 + 6\sigma^4 + 60\sigma^3 + 226\sigma^2 + 316\sigma + 90{,}6)(-\omega) = 0 \\ \omega = \sigma \Bigg\} .$$

The first equation of the first and the second system represents the CRLF gradient value in the contact points of the field level line and the lines, bounding the quality domain from the left and right (the lines of equal degree of stability), the second equation represents the equation of the lines of equal degree of stability. The first equation of the third system represents the equation of a tangent to the CRLF level line, which passes through the origin. As a result of these equations three points of contact of the CRLF level lines and the lines $L_\eta{}'$, $L_\eta{}''$ and $L_{+\beta}$, bounding the quality domain, are defined. In fig. 8 these points are t_1, t_2 for contact of level lines $L_1{}''$, $L_1{}'$ correspondingly and the constant damping line $L_{+\beta}$ and point t_3 for contact of the level line $L_2{}''$ and the line $L_\eta{}''$ of equal degree of stability. It has been found, that the point t_2 belongs to the level line, inscribed into the domain Q, and the lines $L_2{}'$, $L_2{}''$, which correspond to the contact point t_3, and the level line $L_2{}'''$ get beyond this domain (the lines $L_2{}'$, $L_2{}''$ and $L_2{}'''$ represent mappings of a single circular image). Thus, three simply connected closed regions (in fig. 8 they are cross-hatched) are formed, bounded correspondingly by three level lines $L_1{}'$, $L_1{}''$ and $L_1{}''$, representing three sheets of the three-sheeted domain, defined by mapping of the image disc onto the plane s using three branches of the three-valued mapping function. This three-sheeted domain represents the domain of the characteristic equation roots, satisfying the required quality. The image of this domain boundary onto the plane a_n is the circle of radius $r = 2$, bounding the desired closed domain D of the free parameter a_n values, which comply with the given conditions of the system stability.

The developed method for parametric synthesis of the dynamic systems, meeting the robust quality requirements, is based on the circular root locus fields application. It allows to select some regions of the system characteristic equation roots location, belonging to the given quality domain, which defines the required quality indicators values (degree of stability and oscillation), and also to define the corresponding regions of the variable parameters values, ensuring the status when the system quality characteristics do not get beyond the boundaries set. The main advantage of the method is, that it allows to determine the system parameters values, which ensure the required quality indica-

tors for cases when the given system does not comply with the requirements, i.e. to carry on the dynamic systems parametric synthesis. The method can be generalized to the tasks of roots location within the domains of other shapes.

5. Parametric design of the industrial robot robust control systemon the base of the method for interval control systems synthesis

Operation of the industrial robot (see section 2) in conditions of uncertainty is considered, when its parameters are subject to substantial variation. The above described technique is applied for solving the task of the anthropomorphous robot manipulator units control system parametric synthesis. It allows to find analytically the manipulator parameters values variation ranges, which will ensure maintaining the system stability property and the required operational quality within their limits, i. e. to ensure the system robustness.

5.1 Control system model for the case of operation in conditions of uncertainty

Robots loads change with variation of the weights of the items they carry, that causes variation of the load inertia moment j_l, which is linearly included into the characteristic equation coefficients (see (1) and (2)), generating their variation intervals. Currently during the design procedure robots parameters values in the cases of substantial parameters variation are obtained by the technique of tests and mistakes. Conduct parametric synthesis of the manipulator shoulder control system in conditions of its parameters uncertainty using the analytical method described in 3.
Let coefficients of the characteristic equation (3) for the manipulator shoulder unit vary within the following limits:

$$a_0 = 1; \, 0{,}4 \cdot 10^3 \le a_1 \le 0{,}5 \cdot 10^3; \, 0{,}373 \cdot 10^3 \le a_2 \le 0{,}427 \cdot 10^3; \, 0{,}52 \cdot 10^7 \le a_3 \le 0{,}6 \cdot 10^7;$$
$$0{,}488 \cdot 10^9 \le a_4 \le 0{,}56 \cdot 10^9.$$

Suppose any of the coefficients, e. g. a_4, varies continuously along the real axis in the plane of system fundamental frequencies. Taking into account expression (1), the complex mapping function (5), that determines root loci of the interval system relative to a_4, is defined as

$$f(s) = -\frac{\phi(s)}{\psi(s)} = -(s^4 + a_1 s^3 + a_2 s^2 + a_3 s). \tag{32}$$

The control system characteristic equation is

$$\phi(s) + a_4 \psi(s) = s^4 + a_1 s^3 + a_2 s^2 + a_3 s + a_4 = 0. \tag{33}$$

The limit values of equation (33) coefficients variation intervals are entered to the input of the package ANALRL for computer-aided investigation of control systems with variable parameters. During the package functioning the Kharitonov's polynomials of the system characteristic equation are formed:

$$h_1(s) = s^4 + 0,4 \cdot 10^3 s^3 + 0,373 \cdot 10^5 s^2 + 0,6 \cdot 10^7 s + 0,56 \cdot 10^9 = 0,$$
$$h_2(s) = s^4 + 0,5 \cdot 10^3 s^3 + 0,427 \cdot 10^5 s^2 + 0,52 \cdot 10^7 s + 0,488 \cdot 10^9 = 0,$$
$$h_3(s) = s^4 + 0,4 \cdot 10^3 s^3 + 0,427 \cdot 10^5 s^2 + 0,6 \cdot 10^7 s + 0,488 \cdot 10^9 = 0,$$
$$h_4(s) = s^4 + 0,5 \cdot 10^3 s^3 + 0,373 \cdot 10^5 s^2 + 0,52 \cdot 10^7 s + 0,56 \cdot 10^9 = 0.$$

These four equations form the basis for generation of the mathematical model for the robot interval control system in the form of the Kharotiniv's polynomials root loci.

Considering presence of the load inertia moment j_l substantial variations, it is required to find the coefficients variation intervals, ensuring stability of the characteristic equations family.

5.2 Procedure of the control system parametric synthesis

For the task solution apply the method, described in section 3, which allows to calculate the characteristic equations family coefficients intervals, ensuring the system robust stability.

First, zeroes of functions (32) (the poles of the open loop transfer function) are calculated for the above Kharitonov's polynomials and, if they are located in the left-half plane of the plane s (see statement given in subsection 3.3) or on the stability bound $i\omega$, the root loci of Kharitonov's polynomials are generated on the basis of these functions.

As for our example one of zeroes of function (32) is located on the stability bound (in the point $s = 0$), the direction of the corresponding root locus is verified. It is stated that the positive branch is directed from this zero to the left half plane that, according to the above given statement (see 3.3), means the existence of positive stability intervals of the system investigated.

After constructing the parameter functions according to the technique suggested above (see section 3.4), the variable coefficient values from the set A_r^+ (11)) in the cross points of the Kharitonov's polynomials root loci branches with the system stability bound $i\omega$ are determined. For the given case the following values have been obtained:

$a_{r\,1}^{+}$=0,334·10^9 (polynomial h_1);

$a_{r\,2}^{+}$=0,336·10^9 (polynomial h_2);

$a_{r\,3}^{+}$=0,414·10^9 (polynomial h_3);

$a_{r\,4}^{+}$= 0,280·10^9 (polynomial h_4).

According to the corresponding task algorithm and the obtained values, the minimal positive value $a_{r\,min}^{+}$ = 0.280·10⁹ is determined. The interval (13) of a_4 values variation is calculated that ensures system asymptotic stability: d = (0; 0.280·10⁹)). On the basis of the theorem, formulated in 3.4, the following stability condition of the interval system is formed:

0 < a_4 < 0.280·10⁹.

As the root locus parameter varies within the limits $\overline{a}_4$= 0.56·10⁹, and $\underline{a}_4$=0.488·10⁹, the upper one doesn't comply with the stability condition. For ensuring stability of the considered interval control system the upper limit should be set to $\overline{a}_4$= 0.280·10⁹. The limits of the acceptable interval of the coefficient a_4 variation are the following:

$\overline{a}_4$= 0.280·10⁹, $\underline{a}_4$= 0.

From the above described calculations it is evident that the developed method can be applied not only for investigating the interval system stability, but also for calculating intervals of its variable parameters in case the initial system is not stable. It is worth to pinpoint that the method allows to ensure the system asymptotic stability by setting up only one coefficient of its characteristic equation.

6. Conclusion

Industrial robots represent devices, which usually operate in conditions of substantial uncertainty. Therefore, in this chapter the problem of uncertain control systems stability and quality is considered in application to the industrial robot analysis and synthesis tasks solution. The task for synthesis of the interval control systems stable polynomials has been solved. For its solution the investigation of the system root locus portrait behavior at the asymptotic stability boundary has been carried out. On this basis the system robust stability condition was formulated. The method has been developed for setting up the interval polynomial for ensuring its stability in cases, when the stability verification showed, that the initial polynomial was unstable. If the system order $n > 3$, this method is applicable when the Kharitonov's polynomials free root loci initial points are located in the left complex half-plane, because in this case the root locus portrait configuration ensures existence of the stability interval on every branch of the root loci. When $n \leq 3$, the method is applied without any conditions (limitations). The algorithm considered allows to carry on parametric synthesis of the stable interval system without its root locus portrait modification by setting up the limit values of the characteristic polynomial coefficients variation intervals. Thus, the stability interval for the initially unstable polynomial is defined. The obtained stable polynomial can be selected to be the nearest to the initial (given) one in the sense of the distance, measured along its root trajectories with consideration of the appropriate system quality requirements.

The root locus method has also been developed for ensuring the required quality (Q-stability) of the uncertain control system. It is based on inscription of the circular root locus field level line into the given quality domain.

Currently during the industrial robots design procedure in the cases of substantial parameters variation the robots control systems parametric synthesis is often conducted by the method of tests and mistakes. The techniques, considered here, allow to carry on the analysis and parametric synthesis of the robot control system, operating in conditions of parametric uncertainty, using analytical procedures.

7. References

Barmish, B. (1984). Invariance of the strict hurwitz property for polynomials with perturbed coefficients. *IEEE Trans. Automat. Control*, Vol. 29, No. 10, (October 1984) 935–936, ISSN 0018-9286.

Bartlett, A.; Hollot, C. & Lin, H. (1987). Root Location of an entire polytope of polynomials in suffices to check the edges. *Math. Contr.,* Vol. 1, No. 1, (1987) 61–71, ISSN 0932-4194.

Bhattacharyya, S.; Chapellat, H. & Keel, L. (1995). *Robust Control: the Parametric Approach*, Prentice Hall, ISBN 013781576X, New York.

Gaivoronsky, S. (2006). Vertex analysis for the interval polynomial roots localization in the given sector [in Russian], *Theses of the Third International Conference on Control Problems*, p. 95, ISBN 5-201-14987-1, Moscow, June 20–22, 2006, Institute of Control Sciences of the Russian Academy of Sciences, Moscow.

Gryazina, E. & Polyak, B. (2006). Stability regions in the parameter space: D-decomposition revisited. *Automatica*, Vol. 42, No. 1, (January 2006) 13–26, ISSN 0005-1098.

Handbook of Industrial Robotics (1989). Edited by Nof, S. John Wiley and Sons, ISBN 04711778, New York.

Kharitonov, V. (1978). About asymptotic stability of the linear differential equations family equilibrium [in Russian]. *Differentsyal'nye Uravneniya*, Vol. XIV, No. 11, (November 1978) 2086–2088, ISSN 0374-0641.

Kraev, A. & Fursov, A. (2004). Estimating the instability radii for polynomials of arbitrary power [in Russian]. *Nonlinear Dynamics and Control*. Issue 4: Collection of Papers / Eds. Emelyanov, S. & Korovin, S. FIZMATLIT, ISBN 5-9221-0557-4, Moscow, pp. 127–134.

Lavrentyev, M. & Shabat, B. (1987). *Methods of the Complex Variable Functions Theory*, Nauka, Moscow.

Nesenchuk, A. (2002). Parametric synthesis of qualitative robust control systems using root locus fields. *Proceedings of the 15th Triennial World Congress of IFAC*, vol. E, pp. 331–335, ISBN 0-08-044220-X, Barcelona, Spain, July 21–26, 2002, Elsevier Science, Oxford.

Nesenchuk, A. (2005). *Analysis and Synthesis of Robust Dynamic Systems on the Basis of the Root Locus Approach* [in Russian], United Institute of Informatics Problems of the Belarusian National Academy of Sciences, ISBN 985-6744-18-0, Minsk.

Polyak, B. & Tsypkin, Y. (1990). Frequency criteria for robust stability and aperiodicity [in Russian]. *Avtomatika i Telemekhanika*, No. 9, (September 1990) 45–54, ISSN 005-2310.

Polyak, B. & Scherbakov, P. (2002), a. *Robust Stability and Control* [in Russian], Nauka, ISBN 5-02-002561-5, Moscow.

Polyak, B. & Scherbakov, P. (2002), b. Superstable linear control systems I, II ([n Russian]. *Avtomatika i Telemekhanika*, No. 8, (2002) 37–53, ISSN 005-2310.

Rantzer, A. (1992). Stability conditions for polytopes of polynomials. *IEEE Trans. Automat. Control*, Vol. 37, No. 1, (January 1992) 79–89, ISSN 0018-9286.

Rimsky, G. (1972). *Basics of the General Root Locus Theory for Control Systems* [in Russian], Nauka i Tekhnika, Minsk.

Rimsky, G & Taborovetz, V (1978). *Computer-Aided Investigation of the Automatic Systems*, Nauka I Tekhnika, Minsk.

Shaw, J. & Jayasuriya, S. (1993). Robust stability of an interval plant with respect to a convex region in the complex plane. *IEEE Trans. Automat. Control*, Vol. 38, No. 2, (February 1993) 284-287, ISSN 0018-9286.

Soh, C.; Berger, C. & Dabke, K. (1985). On the stability properties of polynomials with perturbed coefficients. *IEEE Trans. Automat. Control*, Vol. 30, No. 10, (October 1985) 1033–1036, ISSN 0018-9286.

Soh, Y. (1989). Strict hurwitz property of polynomials under coefficient perturbations. *IEEE Trans. Automat. Control*, Vol. 34, No. 6, (June 1989) 629–632, ISSN 0018-9286.

Tsypkin, Y. & Polyak, B. (1990). Frequency criteria of robust modality for the linear discrete systems [in Russian]. *Avtomatika i Teeamekhanika*, No. 4, (April 1990) 3–9, ISSN 005-2310.

Tsypkin, Y. (1995). Robust stability of the relay control systems [in Russian]. *Doklady RAN*, Vol. 340, No. 6, (February 1995) 751–753, ISSN 0869-5652.

Uderman, E. (1972). *The Root Locus Method in the Control Systems Theory* [in Russian], Nauka, Moscow.

Vicino, A. (1989). Robustness of pole location in perturbed systems. *Automatica*, Vol. 25, No. 1, (January 1989) 109-113, ISSN 0005-1098.

Stochastic Analysis of a System Containing One Robot and (n-1) Standby Safety Units with an Imperfect Switch

B.S.Dhillon and S.Cheng

1. Introduction

Robots are increasingly being used in industry to perform various types of tasks. These tasks include material handling, spot welding, arc welding and routing. The word 'Robot' is derived from the Czechoslovakian language, in which it means 'worker'. In 1959, the first commercial robot was manufactured by the Planet Corporation and today there around on million robots in use worldwide [1-4].

Although robots are used to replace humans performing various types of complex and hazardous tasks, unfortunately over the years a number of accidents involving robots have occurred. In fact, many people have been killed or injured [5-7]. In using robots, particularly in the industrial sector, often safety units are included with robots. A robot has to be safe and reliable. An unreliable robot may become the cause of unsafe conditions, high maintenance costs, inconvenient, etc.

As robots contain parts such as electrical, electronic, mechanical, pneumatic and hydraulic their reliability problem become a challenging task because of many different sources of failures. Thus, this paper presents a mathematical model for performing reliability and availability analyses of a system containing one robot and (n-1) standby safety units with a switch in mechanism that can fail. More specifically, the robot system is composed of one robot, n identical safety units and a switch to replace a failed safety unit.

The block diagram of the robot system is shown in Figure1 and its corresponding state space diagram is presented in Figure2. The numerals and letter n in the boxes of Figure2 denote system state.

At time t =0, robot, one safety unit and the switch to replace a failed safety unit start operating and n-1 safety units are on standby. The overall robot-safety system can fail the following two ways:
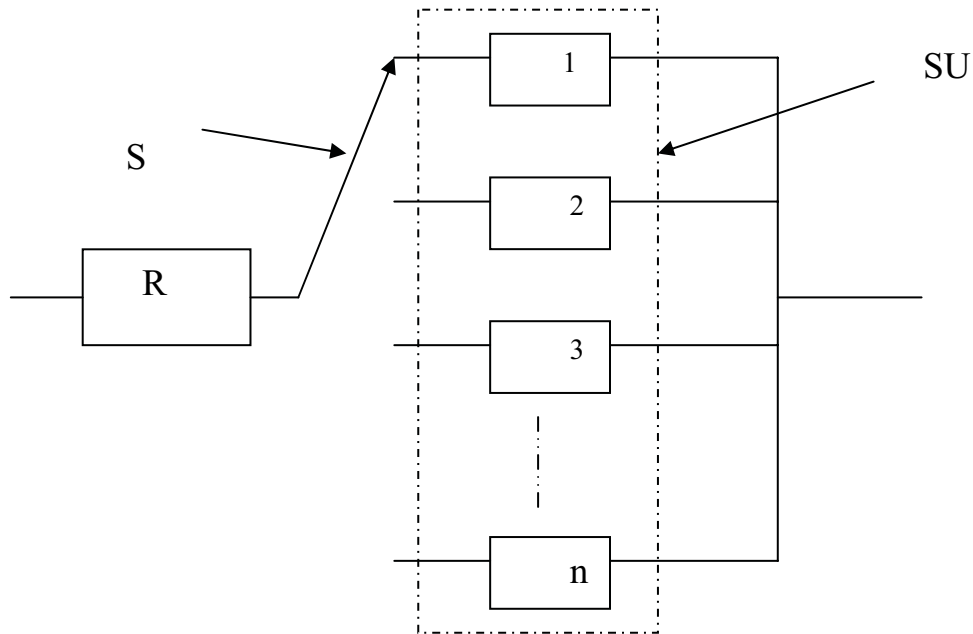
R : Robot
SU : n identical safety units (one operating and n-1 on standby)
S : Switch for replacing a failed safety unit and it can also fail.

Figure 1. The block diagram of the robot-safety system

- The robot fails with a normally working safety unit and the switch. In addition zero or more safety units are on standby.
- The robot fails with one or more safety units failed or considered failed and the switch is either working or failed.
- The following assumptions are associated with this model:
- The robot-safety system is composed of one robot, n identical safety units (only one operates and the rest remain on standby) and a switch.
- Robot, switch and one safety unit start operating simultaneously.
- The completely failed robot-safety system and its individually failed units (i.e. robot, switch and safety unit) can be repaired. Failure and repair rates of robot, switch and safety units are constant.
- The failure robot-safety system repair rates can be constant or non-constant.
- All failures are statistically independent.
- A repaired safety unit, robot, switch or the total robot-safety system is as good as new.
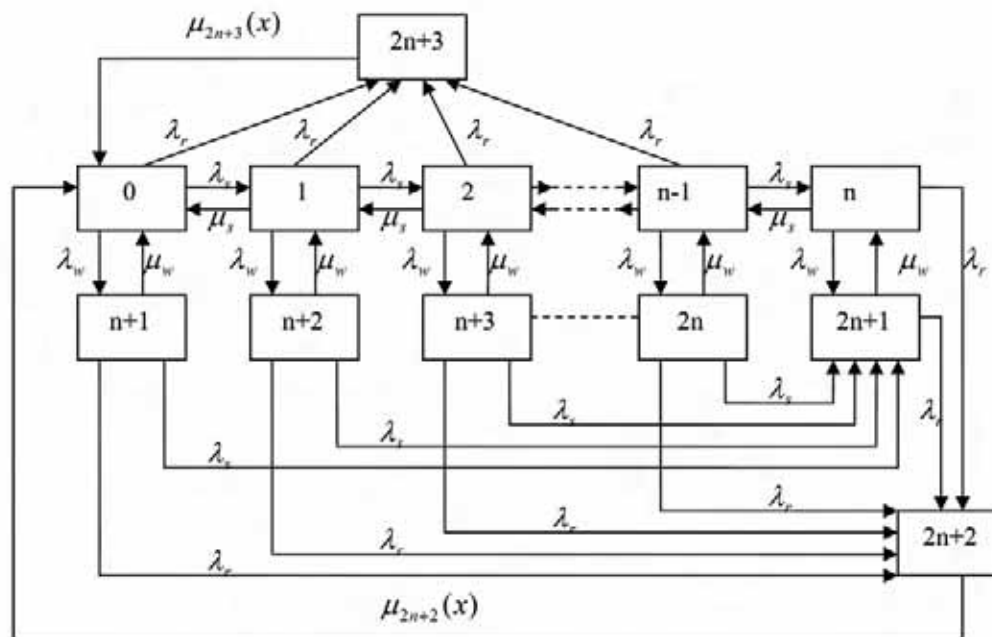
Figure 2. The state space diagram of the robot-safety system

1.1 Notation

The following symbols are associated with the model:

i i^{th} state of the robot-safety system.
for $i = 0$, means the robot, the switch and one safety unit are working normally;
for $i = 1$, means the robot, the switch, one safety unit are working normally and one
safety unit has failed;
for i = k, means the robot, the switch, one safety unit are working normally and k
safety units have failed; (i.e., k = 2,3……n-1);
for i = n, means the robot work, the switch are working normally and all safety units have failed;
for i = h, means the robot, one safety unit still work normally and h-n safety units and the switch have failed; (i.e., h = n+1, n+2,…… 2n)
for $i = 2n+1$, means the robot work normally and all the safety units and the switch have failed;

j	j^{th}	state of the robot–safety system:
		for j = 2n+2, means the total robot-safety system has failed (i.e. the robot , one or
		more safety units have failed or considered failed and the switch is either working or failed);
		for j = 2n+ 3, means the robot-safety system has failed (i.e. the robot has failed while a safety unit and the switch are working normally. In addition, zero or more safety units are on standby);
t		time.
λ_s		Constant failure rate of a safety unit.
λ_γ		Constant failure rate of the robot.
λ_w		Constant failure rate of the switch.
μ_s		Constant repair rate of a safety unit.
μ_w		Constant repair rate of the switch.
$\Delta_x :$		Finite repair time interval.
$\mu_j(x)$		Time dependent repair rate when the failed robot-safety system is in state j:and has an elapsed repair time of x; for j = 2n+2, 2n+3.
$P_j(x.t)\Delta_x$		The probability that at time t, the failed robot-safety system is in state j and the elapsed repair time lies in the interval [x, x+Δx]; for j = 2n+2, 2n+3 .
pdf		Probability density function.
$w^j(x)$		Pdf of repair time when the failed robot-safety system is in state j and has an elapsed time of x; for j = 2n+2, 2n+3 .
$P^j(t)$		Probability that the robot safety system is in state j at time t; for j = 2n+2, 2n+3.
$P^i(t)$		Probability that the robot-safety system is in state i at time t; for i = 0,1,2…2n+1.
P^i		Steady state probability that the robot-safety system is in state i; for i=0,1,..2n+1.
P^j		Steady state probability that robot-safety system is in state j; for j = 2n+2, 2n+3.
s		Laplace transform variable.
$P^i(s)$		Laplace transform of the probability that the robot-safety system is in state i; for i = 0,1,2…2n+1.
$P^j(s)$		Laplace transform of the probability that the robot-safety system is in state j; for j = 2n+2, 2n+3.

AVrs(s)   Laplace transform of the robot-safety system availability with one normally working safety unit, the switch and the robot.

AVr(s)   Laplace transform of the robot-safety system availability with or without a normally safety unit.

AVrs (t)   Robot-safety system time dependent availability with one normally working safety unit, the switch and the robot.

AVr(t)   Robot-safety system time dependent availability with or without a normally working safety unit.

SSAVrs   Robot-safety system steady state availability with one normally working safety unit, the switch and the robot.

SSAVr   Robot-safety system steady state availability with or without a normally working safety unit.

Rrs(s)   Laplace transform of the robot-safety system reliability with one normally working safety unit, the switch and the robot.

Rr(s)   Laplace transform of the robot safety system reliability with or without a normally working safety unit.

MTTFrs   Robot-safety system mean time to failure when the robot working normally with one normally working safety unit.

MTTFr   Robot-safety system mean time to failure with or without a normally working safety unit.

2. Generalized robot-safety system analysis

Using the supplementary method [8,9],the equations of the system associated with Fig.2 can be expressed as follows:

$$\frac{dP_0(t)}{dt} + a_0 P_0(t) = \mu_s P_1(t) + \mu_w P_{n+1}(t) + \sum_{j=2n+2}^{2n+3} P_j(x,t)\mu_j(x)dx \tag{1}$$

$$\frac{dP_i(t)}{dt} + a_i P_i(t) = \lambda_s P_{i-1}(t) + \mu_s P_{i+1}(t) + \mu_w P_{i+n+1}(t) \tag{2}$$

(for $i = 1,2,\ldots\ldots,n\text{-}1$)

$$\frac{dP_n(t)}{dt} + a_n P_n(t) = \lambda_s P_{n-1}(t) + \mu_w P_{2n+1}(t) \tag{3}$$

$$\frac{dP_i(t)}{dt} + a_i P_i(t) = \lambda_w P_{i-n-1}(t) \text{ (for } i = n+1,n+2,\ldots\ldots,2n) \tag{4}$$

$$\frac{dP_{2n+1}(t)}{dt} + a_{2n+1} P_{2n+1}(t) = \lambda_s \sum_{i=n+1}^{2n} P_i(t) + \lambda_w P_n(t) \tag{5}$$

where

$a_0 = \lambda_s + \lambda_w + \lambda_r$

$a_i = \lambda_s + \lambda_w + \lambda_r + \mu_s$ (for i = 1,2,……..,n-1)

$a_n = \lambda_w + \lambda_r + \mu_s$

$a_i = \lambda_s + \lambda_r + \mu_w$ (for i = n+1,n+2,……..,2n)

$a_{2n+1} = \lambda_r + \mu_w$

$$\frac{\partial P_j(x,t)}{\partial t} + \frac{\partial P_j(x,t)}{\partial x} + \mu_j(x)\, P_j(x,t) = 0 \text{ (for } j = 2n+2, 2n+3) \tag{6}$$

The associated boundary conditions are as follows:

$$P_{2n+2}(0,t) = \lambda_r \sum_{i=n}^{2n+1} P_i(t) \tag{7}$$

$$P_{2n+3}(0,t) = \lambda_r \sum_{i=0}^{n-1} P_i(t) \tag{8}$$

At time t = 0, $P_0(0)$ = 1, and all other initial state probabilities are equal to zero.

3. Generalized Robot-Safety System Laplace Transforms of State Probabilities

By solving Equations (1)-(8) with the Laplace transform method, we get the following

Laplace transforms of state probabilities:

$$P_0(s) = \left[s\left(1 + \sum_{i=1}^{n} Y_i(s) + \frac{\lambda_w}{s + a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i(s) + \sum_{j=2n+2}^{2n+3} a_j(s)\frac{1 - W_j(s)}{s}\right)\right]^{-1} = \frac{1}{G(s)} \tag{9}$$

$$P_i(s) = Y_i(s)\, P_0(s) \qquad\qquad \text{(for } i = 1,2,……..,n) \tag{10}$$

$$P_i(s) = V_i(s)\, P_0(s) \qquad\qquad \text{(for } i = n+2, n+2,……..,2n+1) \tag{11}$$

$$P_{n+1}(s) = \frac{\lambda_w}{s + a_{n+1}}\, P_0(s) \tag{12}$$

$$P_j(s) = a_j(s) \frac{1 - W_j(s)}{s} P_0(s) \qquad (\text{for } j = 2n+2, 2n+3) \qquad (13)$$

where

$$L_i(s) = (s + a_i) - \frac{\lambda_w \mu_w}{s + a_{i+n+1}} \qquad (\text{for } i = 1, 2, \ldots, n)$$

$$D_1(s) = L_1(s)$$

$$D_i(s) = L_i(s) - \frac{\lambda_s \mu_s}{D_{i-1}(s)} \qquad (\text{for } i = 2, \ldots, n)$$

$$A_i(s) = \frac{\lambda_s^i}{\prod_{h=1}^{i} D_h(s)} \qquad (\text{for } i = 1, 2, \ldots, n-1)$$

$$B_i(s) = \frac{\mu_s}{D_i(s)} \qquad (\text{for } i = 1, 2, \ldots, n-1)$$

$$Y_i(s) = \sum_{h=i}^{n-1} A_h(s) \prod_{k=i}^{h-1} B_k(s) + \prod_{h=i}^{n-1} B_h(s) Y_n(s)$$

$(\text{for } i = 1, 2, \ldots, n-1)$

$$V_i(s) = \frac{\lambda_w}{s + a_i} Y_{i-n-1}(s) \qquad (\text{for } i = n+2, \ldots, 2n)$$

$$V_{2n+1}(s) = \frac{\lambda_s \lambda_w}{(s + a_{n+1})(s + a_{2n+1})} + \frac{\lambda_s}{s + a_{2n+1}} \sum_{i=1}^{n-1} \frac{\lambda_w}{s + a_{i+n+1}} Y_i(s) + \frac{\lambda_w}{s + a_{2n+1}} Y_n(s)$$

$$Y_n(s) =$$

$$\frac{\lambda_s A_{n-1}(s) + \dfrac{\lambda_s \lambda_w \mu_w}{(s + a_{n+1})(s + a_{2n+1})} + \dfrac{\lambda_s \mu_w}{s + a_{2n+1}} \displaystyle\sum_{i=1}^{n-1} \dfrac{\lambda_w}{s + a_{i+n+1}} \sum_{h=i}^{n-1} [A_h(s) \prod_{k=i}^{h-1} B_k(s)]}{L_n(s) - \lambda_s B_{n-1}(s) - \dfrac{\lambda_s \mu_w}{s + a_{2n+1}} \displaystyle\sum_{i=1}^{n-1} \dfrac{\lambda_w}{s + a_{i+n+1}} \prod_{h=i}^{n-1} B_h(s)}$$

$$a_{2n+2}(s) = \lambda_r \left[Y_n(s) + \frac{\lambda_w}{s + a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i(s) \right]$$

$$a_{2n+3}(s) = \lambda_r [1 + \sum_{i=1}^{n-1} Y_i(s)]$$

$$G(s) = s(1 + \sum_{i=1}^{n} Y_i(s) + \frac{\lambda_w}{s + a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i(s) + \sum_{j=2n+2}^{2n+3} a_j(s) \frac{1 - W_j(s)}{s}) \tag{14}$$

$$W_j(s) = \int_0^\infty e^{-sx} w_j(x) dx \quad \text{for } j = 2n+2, 2n+3) \tag{15}$$

$$w_j(x) = \exp[-\int_0^x \mu_j(\delta) d\delta] \, \mu_j(x)$$

where

$w_j(x)$ is the failed robot safety system repair time probability density function
The Laplace transform of the robot-safety system availability with one normally working safety unit, the switch and the robot is given by:

$$AV_{rs}(s) = \sum_{i=0}^{n-1} P_i(s) + \sum_{i=n+1}^{2n} P_i(s) = \frac{1 + \sum_{i=1}^{n-1} Y_i(s) + \frac{\lambda_w}{s + a_{n+1}} + \sum_{i=n+2}^{2n} V_i(s)}{G(s)} \tag{16}$$

The Laplace transform of the robot-safety system availability with or without a normally working safety unit:

$$AV_r(s) = \sum_{i=0}^{2n+1} P_i(s) = \frac{1 + \frac{\lambda_w}{s + a_{n+1}} + \sum_{i=1}^{n} Y_i(s) + \sum_{i=n+2}^{2n+1} V_i(s)}{G(s)} \tag{17}$$

Taking the inverse Laplace transforms of the above equations, we can obtain the time dependent state probabilities, $P_i(t)$ and $P_j(t)$, and robot-safety system availabilities,

AVrs(t) and AVr(t).

3.1 Robot Safety System Time Dependent Analysis For A Special Case

For two safety units (i.e., one working, other one on standby)
Substituting n=2 into Equations (9)-(16), we get

$$P_0(s) = \frac{1}{s[1 + \sum_{i=1}^{2} Y_i(s) + \frac{\lambda_w}{s + a_3} + \sum_{i=4}^{5} V_i(s) + \sum_{j=6}^{7} a_j(s)\frac{1 - W_j(s)}{s}]} = \frac{1}{G(s)} \tag{18}$$

$$P_i(s) = Y_i(s) P_0(s) \qquad \text{(for } i = 1,2) \tag{19}$$

$$P_3(s) = \frac{\lambda_w}{s + a_3} P_0(s) \tag{20}$$

$$P_i(s) = V_i(s) P_0(s) \qquad \text{(for } i = 4,5) \tag{21}$$

$$P_j(s) = a_j(s)\frac{1 - W_j(s)}{s} P_0(s) \tag{22}$$

where

$$Y_2(s) = \frac{\lambda_s \frac{\lambda_s}{L_1(s)} + \frac{\lambda_s \lambda_w \mu_w}{(s + a_3)(s + a_5)} + \frac{\lambda_s \lambda_w \mu_w}{(s + a_4)(s + a_5)} \frac{\mu_s}{L_1(s)}}{L_2(s) - \lambda_s \frac{\mu_s}{L_1(s)} - \frac{\lambda_s \lambda_w \mu_w}{(s + a_4)(s + a_5)} \frac{\mu_s}{L_1(s)}}$$

$$Y_1(s) = \frac{\lambda_s}{L_1(s)} + \frac{\mu_s}{L_1(s)} Y_2(s)$$

$$V_5(s) = \frac{\lambda_s \lambda_w}{(s + a_3)(s + a_5)} + \frac{\lambda_s}{s + a_5} \frac{\lambda_w}{s + a_4} Y_1(s) + \frac{\lambda_w}{s + a_5} Y_2(s)$$

$$V_4(s) = \frac{\lambda_w}{s + a_4} Y_1(s)$$

$$a_6(s) = \lambda_r [\, Y_2(s) + \frac{\lambda_w}{s + a_3} + \sum_{i=4}^{5} V_i(s)\,]$$

$$a_7(s) = \lambda_r [\, 1 + Y_1(s)]$$

$$L_1(s) = (s + a_1) - \frac{\lambda_w \mu_w}{s + a_4}$$

$$L_2(s) = (s + a_2) - \frac{\lambda_w \mu_w}{s + a_5}$$

$$G(s) = s[1 + \sum_{i=1}^{2} Y_i(s) + \frac{\lambda_w}{s + a_3} + \sum_{i=4}^{5} V_i(s) + \sum_{j=6}^{7} a_j(s) \frac{1 - W_j(s)}{s}]$$ (23)

The Laplace transform of the robot-safety system availability with one normally working safety unit, the switch and the robot is given by:

$$AV_{rs}(s = \sum_{i=0}^{1} P_i(s) + \sum_{i=3}^{4} P_i(s) = \frac{1 + Y_1(s) + \frac{\lambda_w}{s + a_3} + V_4(s)}{G(s)}$$ (24)

The Laplace transform of the robot-safety system availability with or without a normally working safety unit is given by:

$$AV_r(s) = \sum_{i=0}^{5} P_i(s) = \frac{1 + \sum_{i=1}^{2} Y_i(s) + \frac{\lambda_w}{s + a_3} + \sum_{i=4}^{5} V_i(s)}{G(s)}$$ (25)

Taking the inverse Laplace transforms of the above equations, we can obtain the time dependent state probabilities, $P_i(t)$ and $P_j(t)$, and robot-safety system availabilities,

AVrs(t) and AVr(t).

Thus, for the failed robot-safety system repair time x is exponentially distributed repair times, the probability function is expressed by

$$w_j(x) = \mu_j e^{-\mu_j x} \qquad (\mu_j > 0, \ j = 6,7)$$ (26)

where

x is the repair time variable and μ_j is the constant repair rate of state j.
Substituting equation (26) into equation (15), we can get

$$W_j(s) = \frac{\mu_j}{s + \mu_j} \qquad (\mu_j > 0, j = 6,7)$$ (27)

By inserting Equation (27) into Equations (9)-(13), setting $\lambda_s = 0.002$, $\mu_s = 0.00015$, $\lambda_w = 0.001$, $\mu_w = 0.0003$, $\lambda_r = 0.00009$, $\mu_6 = 0.0001$, $\mu_7 = 0.00015$; and using Matlab computer program [10], the Figure 3 plots were obtained. These plots show that state probabilities decrease and increase with varying time t.
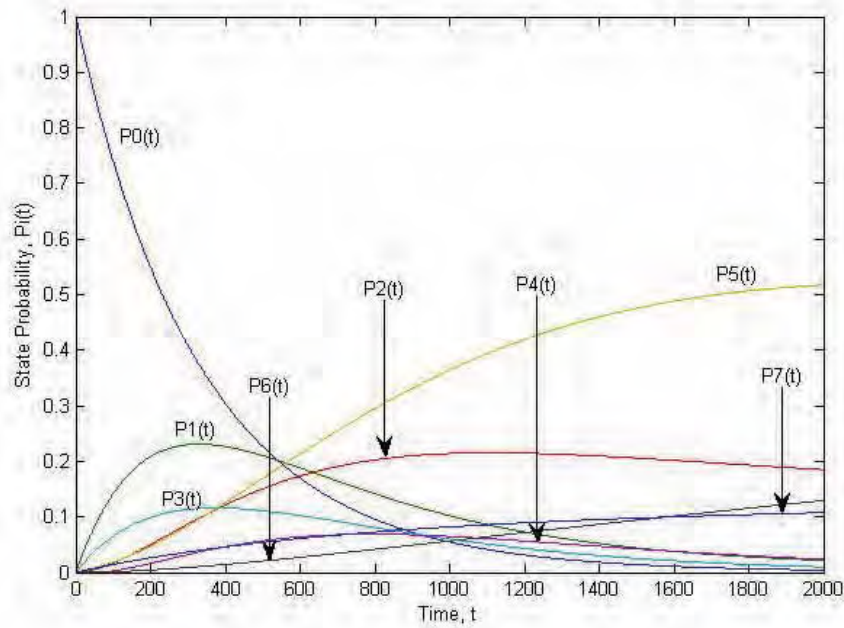
$$\lambda_s = 0.002, \ \mu_s = 0.00015, \ \lambda_w = 0.001, \ \mu_w = 0.0003,$$
$$\lambda_r = 0.00009, \ \mu_6 = 0.0001, \ \mu_7 = 0.00015$$

Figure 3. Time-dependent probability plots for a robot safety system with exponential distributed failed system repair time.

4. Generalized Robot Safety System Steady State Analysis

As time approaches infinity, all state probabilities reach the steady state. Thus, from

Equations (1)-(8) get:

$$a_0 P_0 = \mu_s P_1 + \mu_w P_n + \sum_{j=2n+2}^{2n+3} P_j(x)\mu_j(x)dx \tag{28}$$

$$a_i P_i = \lambda_s P_{i-1} + \mu_s P_{i+1} + \mu_w P_{i+n+1} \tag{29}$$

(for i = 1,2,……..,n-1)

$$a_n P_n = \lambda_s P_{n-1} + \mu_w P_{2n+1} \tag{30}$$

$$a_i P_i = \lambda_w P_{i-n-1} \tag{31}$$

(for i = n+1,n+2,……..,2n-k-1)

$$\text{a}_{2n+1} \text{P}_{2n+1} = \lambda_s \sum_{i=n+1}^{2n} P_i + \lambda_w \text{P}_n \tag{32}$$

where

$$\text{a}_0 = \lambda_s + \lambda_w + \lambda_r$$
$$\text{a}_i = \lambda_s + \lambda_w + \lambda_r + \mu_s \qquad (\text{ for } i = 1,2,\ldots\ldots,\text{n-1})$$
$$\text{a}_n = \lambda_w + \lambda_r + \mu_s$$
$$\text{a}_i = \lambda_s + \lambda_r + \mu_w \qquad (\text{ for } i = \text{n+1,n+2},\ldots\ldots,\text{2n})$$
$$\text{a}_{2n+1} = \lambda_r + \mu_w$$

$$\frac{dP_j(x)}{dx} + \mu_j(x)P_j(x) = 0 \quad (\text{ for } j = \text{2n+2,2n+3}) \tag{33}$$

The associated boundary conditions are as follows:

$$\text{P}_{2n+2}(0) = \lambda_r \sum_{i=n}^{2n+1} P_i \tag{34}$$

$$\text{P}_{2n+3}(0) = \lambda_r \sum_{i=0}^{n-1} P_i \tag{35}$$

Solving Equations (28) - (33), and together with

$$\sum_{i=0}^{2n+1} P_i + \sum_{j=2n+2}^{2n+3} P_j = 1 \tag{36}$$

We get:

$$\text{P}_0 = (1 + \sum_{i=1}^{n} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n} V_i + \sum_{j=2n+2}^{2n+3} a_j E_j[x])^{-1} = \frac{1}{G} \tag{37}$$

$$\text{P}_i = Y_i \text{P}_0 \qquad (\text{ for } i = 1,2,\ldots\ldots,\text{n}) \tag{38}$$

$$\text{P}_i = V_i \text{P}_0 \qquad (\text{ for } i = \text{n+2},\ldots\ldots,\text{2n+1}) \tag{39}$$

$$\text{P}_{n+1} = \frac{\lambda_w}{a_n} \text{P}_0 \tag{40}$$
$$\text{P}_j = a_j E_j[x] \text{P}_0$$

(for j = 2n+2, 2n+3) (41)

where

$L_i = \lim_{s \to 0} L_i(s)$ (for i = 1,2,……..,n)

$D_1 = L_1$

$D_i = L_i - \dfrac{\lambda_s \mu_s}{D_{i-1}}$ (for i = 2,……..,n)

$A_i = \dfrac{\lambda_s^i}{\prod\limits_{h=1}^{i} D_h}$ (for i = 1,2,……..,n-1)

$B_i = \dfrac{\mu_s}{D_i}$ (for i = 1,2,……..,n-1)

$Y_i = \sum\limits_{h=i}^{n-1} A_h \prod\limits_{k=i}^{h-1} B_k + \prod\limits_{h=i}^{n-1} B_h Y_n$

(for i = 1,2,……..,n-1)

$V_i = \dfrac{\lambda_w}{a_i} Y_{i-n-1}$ (for i = n+2,……..,2n)

$V_{2n+1} = \dfrac{\lambda_s \lambda_w}{a_{n+1} a_{2n+1}} + \dfrac{\lambda_s}{a_{2n+1}} \sum\limits_{i=1}^{n-1} \dfrac{\lambda_w}{a_{i+n+1}} Y_i + \dfrac{\lambda_w}{a_{2n+1}} Y_{n-k}$

$Y_n = \dfrac{\lambda_s A_{n-1} + \dfrac{\lambda_s \lambda_w \mu_w}{a_n a_{2n+1}} + \dfrac{\lambda_s \mu_w}{a_{2n+1}} \sum\limits_{i=1}^{n-1} \dfrac{\lambda_w}{a_{i+n+1}} \sum\limits_{h=i}^{n-1} A_h \prod\limits_{k=i}^{h-1} B_k}{L_n - \lambda_s B_{n-1} - \dfrac{\lambda_s \mu_w}{a_{2n+1}} \sum\limits_{i=1}^{n-1} \dfrac{\lambda_w}{a_{i+n+1}} \prod\limits_{h=i}^{n-1} B_h}$

$a_{2n+2} = \lambda_r (Y_n + \sum\limits_{i=n+2}^{2n+1} V_i + \dfrac{\lambda_w}{a_{n+1}})$

$a_{2n+3} = \lambda_r (1 + \sum\limits_{i=1}^{n-1} Y_i)$

$$G = 1 + \sum_{i=1}^{n-1} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i + \sum_{j=2n+2}^{2n+3} a_j E_j[x] \qquad (42)$$

$$E_j[x] = \int_0^\infty \exp[-\int_0^x \mu_j(\delta)d\delta]dx$$

$$(43)$$

$$= \int_0^\infty x w_j(x)dx \qquad\qquad (\text{for } j = 2n+2, 2n+3)$$

where

$w_j(x)$ is the failed robot safety system repair time probability density function

$E_j[x]$ is the mean time to robot safety system repair when the failed robot safety system is in state j and has an elapsed repair time x.

The generalized steady state availability of the robot safety system with one normally working normally safety unit, the switch and the robot is given by

$$\text{SSAVrs} = \sum_{i=0}^{n-1} P_i + \sum_{i=n+1}^{2n} P_i = \frac{1 + \sum_{i=1}^{n-1} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n} V_i}{G} \qquad (44)$$

Similarly, the generalized steady state availability of the robot safety system with or without a working safety units is

$$\text{SSAVr} = \sum_{i=0}^{2n+1} P_i = \frac{1 + \sum_{i=1}^{n} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i}{G} \qquad (45)$$

For different failed robot-safety system repair time distributions, we get different expressions for G as follows:

1) For the failed robot-safety system Gamma distributed repair time x, the probability

density function is expressed by

$$w_j(x) = \frac{\mu_j^\beta x^{\beta-1} e^{-\mu_j x}}{\Gamma(\beta)} \quad (\beta > 0, j = 2n+2, 2n+3) \qquad (46)$$

where

x is the repair time variable, $\Gamma(\beta)$ is the gamma function, μ_j is the scale pa-parameter and β is the shape parameter.

Thus, the mean time to robot-safety system repair is given by

$$E_j(x) = \int_0^\infty x w_j(x)dx = \frac{\beta}{\mu_j} \qquad (\beta > 0, j = 2n+2, 2n+3) \tag{47}$$

Substituting equation (47) into equation (42), we get

$$G = 1 + \sum_{i=1}^{n-1} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i + \sum_{j=2n+2}^{2n+3} a_j \frac{\beta}{\mu_j} E_j[x] \tag{48}$$

2) For the failed robot-safety system Weibull distributed repair time x, the probability

density function is expressed by

$$w_j(x) = \mu_j \beta x^{\beta-1} e^{-\mu_j(x)^\beta} \qquad (\beta > 0, j = 2n+2, 2n+3) \tag{49}$$

where

x is the repair time variabl, μ_j is the scale parameter and β is the shape pa-rameter.

Thus, the mean time to robot-safety system repair is given by

$$E_j[x] = \int_0^\infty x W_j(x)dx = (\frac{1}{\mu_j})^{1/\beta} \frac{1}{\beta} \Gamma(\frac{1}{\beta}) \qquad (\beta > 0, j = 2n+2, 2n+3) \tag{50}$$

Substituting (50) into equation (42), we can get

$$G = 1 + \sum_{i=1}^{n-1} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i + \sum_{j=2n+2}^{2n+3} a_j (\frac{1}{\mu_j})^{1/\beta} \frac{1}{\beta} \Gamma(\frac{1}{\beta}) \tag{51}$$

3) For the failed robot-safety system Rayleigh distributed repair time x, the probability

density function is expressed by

$$w_j(x) = \mu_j x e^{-\mu_j x^2/2} \qquad (\mu_j > 0, j = 2n+2, 2n+3) \tag{52}$$

where

x is the repair time varable, μ_j is the scale parameter.

Thus, the mean time to robot-safety system repair is given by

$$E_j(x) = \int_0^\infty xW_j(x)dx = \sqrt{\frac{\pi}{2\mu_j}} \qquad (\mu_j > 0, j = 2n+2, 2n+3) \tag{53}$$

Substituting (53) into equation (42), we can get

$$G = 1 + \sum_{i=1}^{n-1} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i + \sum_{j=2n+2}^{2n+3} a_j \sqrt{\frac{\pi}{2\mu_j}} \tag{54}$$

4) For the failed robot system Lognormal distributed repair time x, the probability

density function is expressed by

$$w_j(x) = \frac{1}{\sqrt{2\pi}x\sigma_{y_j}} e^{[\frac{-(Inx-\mu_{y_j})^2}{2\sigma_{y_j}^2}} \qquad (\text{for } j = 2n+2, 2n+3) \tag{55}$$

where

x is the repair time variable, Inx is the natural logarithm of x with a mean μ and

variance σ^2. The conditions μ and σ^2 on parameters are:

$$\sigma_{y_j} = In\sqrt{1 + (\frac{\sigma_{x_j}}{\mu_{x_j}})^2} \tag{56}$$

$$\mu_{y_j} = In\sqrt{\frac{\mu_{x_j}^4}{\mu_{x_j}^2 + \sigma_{x_j}^2}} \tag{57}$$

Thus, the mean time to robot-safety system repair is given by

$$E_j(x) = e^{(\mu_{y_j} + \frac{\sigma_{y_j}^2}{2})} \qquad (\text{for } j = 2n+2, 2n+3) \tag{58}$$

Substituting (58) into equation (42), we can get

$$G = 1 + \sum_{i=1}^{n-1} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i + \sum_{j=2n+2}^{2n+3} a_j \, e^{(\mu_{y_j} + \frac{\sigma_{y_j}^2}{2})} \quad \text{(for } j = 2n+2,2n+3) \tag{59}$$

5) For the failed robot system exponentially distributed repair time x, the probability

density function is expressed by

$$w_j(x) = \mu_j \, e^{-\mu_j x} \qquad (\mu_j > 0, \ j = 2n+2,2n+3) \tag{60}$$

where

x is the repair time variable and μ_j is the constant repair rate of state j.

Thus, the mean time to robot-safety system repair is given by

$$E_j(x) = \int_0^\infty x w_j(x) dx = \frac{1}{\mu_j} \qquad (\beta > 0, j = 2n+2, 2n+3) \tag{61}$$

Substituting equation (61) into equation (42), we can get

$$G = 1 + \sum_{i=1}^{n-1} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i + \sum_{j=2n+2}^{2n+3} a_j \frac{1}{\mu_j} \tag{62}$$

4.1 The Robot-Safety System Steady State Analysis For A Special Case

For n = 2, from Equations (37)-(45), we get

$$P_0 = \frac{1}{1 + \sum_{i=1}^{2} Y_i + \frac{\lambda_w}{a_3} + \sum_{i=4}^{5} V_i + \sum_{j=6}^{7} a_j E_j[x]} \tag{63}$$

$$P_i = Y_i \, P_0 \qquad\qquad \text{(for } i = 1,2) \tag{64}$$

$$P_3 = \frac{\lambda_w}{a_3} P_0 \tag{65}$$

$$P_i = V_i P_0 \qquad\qquad \text{(for } i = 4,5) \tag{66}$$

$$P_j = a_j E_j [x] P_0 \tag{67}$$

where

$$Y_2(s) = \frac{\lambda_s \dfrac{\lambda_s}{L_1} + \dfrac{\lambda_s \lambda_w \mu_w}{a_3 a_5} + \dfrac{\lambda_s \lambda_w \mu_w}{a_4 a_5} \dfrac{\mu_s}{L_1}}{L_2 - \lambda_s \dfrac{\mu_s}{L_1} - \dfrac{\lambda_s \lambda_w \mu_w}{a_4 a_5} \dfrac{\mu_s}{L_1}}$$

$$Y_1 = \frac{\lambda_s}{L_1} + \frac{\mu_s}{L_1} Y_2$$

$$V_5 = \frac{\lambda_s \lambda_w}{a_3 a_5} + \frac{\lambda_s \lambda_w}{a_4 a_5} Y_1 + \frac{\lambda_w}{a_5} Y_2$$

$$V_4 = \frac{\lambda_w}{a_4} Y_1$$

$$a_6 = \lambda_r \left[Y_2 + \frac{\lambda_w}{a_3} + \sum_{i=4}^{5} V_i \right]$$

$$a_7 = \lambda_r [1 + Y_1]$$

$$L_1 = a_1 - \frac{\lambda_w \mu_w}{a_4}$$

$$L_2 = a_2 - \frac{\lambda_w \mu_w}{a_5}$$

$$G = 1 + \sum_{i=1}^{2} Y_i + \frac{\lambda_w}{a_3} + \sum_{i=4}^{5} V_i + a_j E_j [x] \tag{68}$$

$$\text{SSAVrs} = \sum_{i=0}^{1} P_i + \sum_{i=3}^{4} P_i = \frac{1 + Y_1 + \dfrac{\lambda_w}{a_3} + a_4}{G} \tag{69}$$

$$\text{SSAVr} = \sum_{i=0}^{5} P_i = \frac{1 + \sum_{i=1}^{2} Y_i + \dfrac{\lambda_w}{a_3} + \sum_{i=4}^{5} V_i}{G} \tag{70}$$

For exponentially distributed failed robot-safety system repair Equation (61) into

Equations (69) and (70), setting:

$\lambda_s = 0.0002$, $\lambda_w = 0.001$, $\mu_w = 0.0003$, $\lambda_r = 0.00009$, $\mu_6 = 0.0001$, $\mu_7 = 0.00015$;

and using matlab computer program [10], the Figure 4 plot were obtained. The plot shows, as expected, that SSAV_r is greater than SSAV_{rs} and both of them increase slightly with the increasing values of the safety unit repair rate.
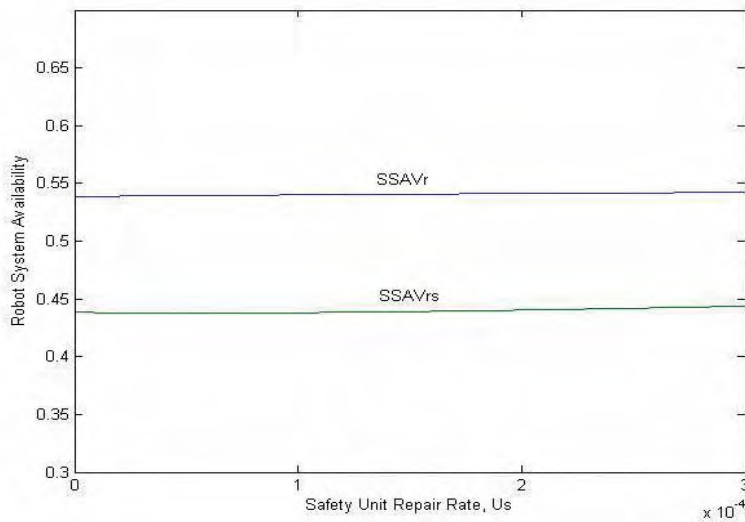
5. Robot-Safety System Reliability and MTTF Analysis

Setting $\mu_j = 0$, (for $j = 2n+2, 2n+3$), in Figure 2 and using the Markov method[11], we write the following equations for the modified figure:

$$\frac{dP_0(t)}{dt} + a_0 P_0(t) = \mu_s P_1(t) + \mu_w P_{n+1}(t) \tag{71}$$

$$\frac{dP_i(t)}{dt} + a_i P_i(t = \lambda_s P_{i-1}(t) + \mu_s P_{i+1}(t) + \mu_w P_{i+n+1}(t) \tag{72}$$

(for $i = 1,2,……..,n-1$)

$\lambda_s = 0.0002$, $\lambda_w = 0.001$, $\mu_w = 0.0003$, $\lambda_r = 0.00009$,
$\mu_6 = 0.0001$, $\mu_7 = 0.00015$

Figure 4. Robot system steady state availability versus safety unit repair rate (μ_s) plots with exponentially distributed failed system repair time

$$\frac{dP_n(t)}{dt} + a_n P_n(t) = \lambda_s P_{n-1}(t) + \mu_w P_{2n+1}(t) \tag{73}$$

$$\frac{dP_i(t)}{dt} + a_i P_i(t) = \lambda_w P_{i-n-1}(t) \qquad (\text{for } i = n+1, n+2, \ldots, 2n) \tag{74}$$

$$\frac{dP_{2n+1}(t)}{dt} + a_{2n+1} P_{2n+1}(t) = \lambda_s \sum_{i=n+1}^{2n} P_i(t) + \lambda_w P_n(t) \tag{75}$$

$$\frac{dP_{2n+2}(t)}{dt} = \lambda_r \sum_{i=n}^{2n+1} P_i(t) \tag{76}$$

$$\frac{dP_{2n+3}(t)}{dt} = \lambda_r \sum_{i=0}^{n-1} P_i(t) \tag{77}$$

where

$a_0 = \lambda_s + \lambda_w + \lambda_r$
$a_i = \lambda_s + \lambda_w + \lambda_r + \mu_s \qquad (\text{for } i = 1, 2, \ldots, n-1)$
$a_n = \lambda_w + \lambda_r + \mu_s$
$a_i = \lambda_s + \lambda_r + \mu_w \qquad (\text{for } i = n+1, n+2, \ldots, 2n)$
$a_{2n+1} = \lambda_r + \mu_w$

At time $t = 0$, $P_0(0) = 1$ and all other initial conditions state probabilities are equal to zero.
By solving Equations (71) – (77) with the aid of Laplace transforms, we get:

$$P_0(s) = P_0(s) = \left[s\left(1 + \sum_{i=1}^{n} Y_i(s) + \frac{\lambda_w}{s + a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i(s) + \sum_{j=2n+2}^{2n+3} \frac{a_j(s)}{s} \right) \right]^{-1} = \frac{1}{G(s)} \tag{78}$$

$$P_i(s) = Y_i(s) P_0(s) \qquad (\text{for } i = 1, 2, \ldots, n) \tag{79}$$

$$P_i(s) = V_i(s) P_0(s) \qquad (\text{for } i = n+2, n+2, \ldots, 2n+1) \tag{80}$$

$$P_{n+1}(s) = \frac{\lambda_w}{s + a_{n+1}} P_0(s) \tag{81}$$

$$P_j(s) = \frac{a_j(s)}{s} P_0(s) \quad \text{(for } j = 2n+2, 2n+3) \tag{82}$$

$$G(s) = s[1 + \sum_{i=1}^{n} Y_i(s) + \frac{\lambda_w}{s + a_{n+1}} + \sum_{i=n+2}^{2n+1} V_i(s) + \sum_{j=2n+2}^{2n+3} \frac{a_j(s)}{s}] \tag{83}$$

The Laplace transform of the robot-safety system reliability with one normally working safety unit, the switch and the robot is given by:

$$R_{rs}(s) = \sum_{i=0}^{n-1} P_i(s) + \sum_{i=n+1}^{2n} P_i(s) = \frac{1 + \sum_{i=1}^{n-1} Y_i(s) + \frac{\lambda_w}{s + a_{n+1}} + \sum_{i=n+2}^{2n} V_i(s)}{G(s)} \tag{84}$$

Similarly, the Laplace transform of the robot safety system reliability with or without a working safety unit is

$$R_r(s) = \sum_{i=0}^{2n+1} P_i(s) = \frac{1 + \frac{\lambda_w}{s + a_{n+1}} + \sum_{i=1}^{n} Y_i(s) + \sum_{i=n+2}^{2n+1} V_i(s)}{G(s)} \tag{85}$$

Using Equation (83) and Reference [11], the robot-safety system mean time to failure with one normally working safety unit, the switch and the robot is given by

$$\text{MTTF}_{rs} = \lim_{s \to 0} R_{rs}(s) = \frac{1 + \sum_{i=1}^{n-1} Y_i + \frac{\lambda_w}{a_{n+1}} + \sum_{i=n+2}^{2n} V_i}{\sum_{j=2n+2}^{2n+3} a_j} \tag{86}$$

Similarly, using Equation (84) and Reference [11], the robot safety system mean time to failure with or without a working safety unit

$$\text{isMTTF}_r = \lim_{s \to 0} R_r(s) = \frac{1}{\lambda_r} \tag{87}$$

5.1 Robot-Safety System MTTF Analysis for a Special Case

Substituting n = 2 into Equation (86) and (87), we get

$$\text{MTTF}_{rs} = \frac{1 + Y_1 + \dfrac{\lambda_w}{a_3} + V_4}{\displaystyle\sum_{j=6}^{7} a_j} \tag{88}$$

$$\text{MTTF}_r = \frac{1}{\lambda_r} \tag{89}$$

where

$$Y_2 = \frac{\lambda_s \dfrac{\lambda_s}{L_1} + \dfrac{\lambda_s \lambda_w \mu_w}{a_3 a_5} + \dfrac{\lambda_s \lambda_w \mu_w}{a_4 a_5} \dfrac{\mu_s}{L_1}}{L_2 - \lambda_s \dfrac{\mu_s}{L_1} - \dfrac{\lambda_s \lambda_w \mu_w}{a_4 a_5} \dfrac{\mu_s}{L_1}}$$

$$Y_1 = \frac{\lambda_s}{L_1} + \frac{\mu_s}{L_1} Y_2$$

$$V_5 = \frac{\lambda_s \lambda_w}{a_3 a_5} + \frac{\lambda_s \lambda_w}{a_4 a_5} Y_1 + \frac{\lambda_w}{a_5} Y_2$$
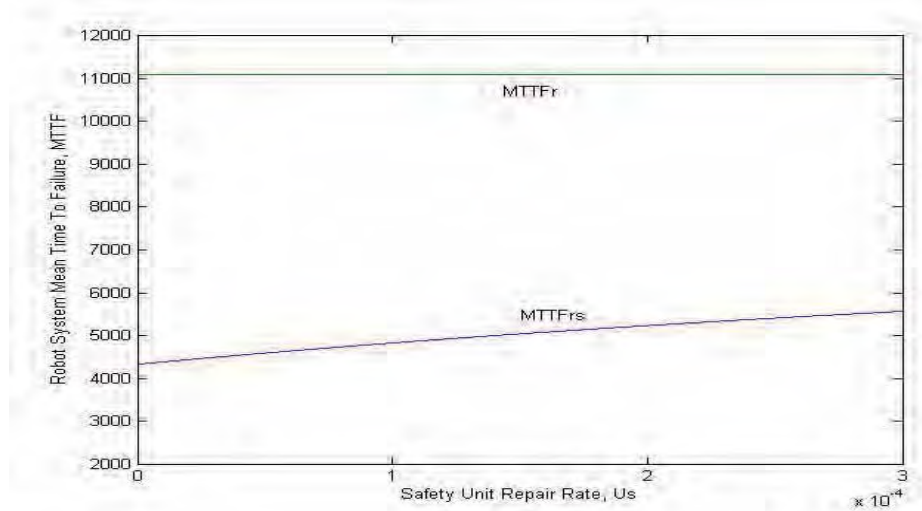
$$V_4 = \frac{\lambda_w}{a_4} Y_1$$

$$a_6 = \lambda_r \left[Y_2 + \frac{\lambda_w}{a_3} + \sum_{i=4}^{5} V_i \right]$$

$$a_7 = \lambda_r \left[1 + Y_1 \right]$$

$$L_1 = a_1 - \frac{\lambda_w \mu_w}{a_4}$$

$$L_2 = a_2 - \frac{\lambda_w \mu_w}{a_5}$$

For λ_s =0.0002, λ_w =0.001, μ_w = 0.0003, λ_r = 0.00009, and using Equations (88)-(89) and Matlab computer program [10], in Figure 5 MTTF$_{rs}$ and MTTF$_r$ plots were obtained. λ_s =0.0002, λ_w =0.001, μ_w = 0.0003, λ_r = 0.00009

Figure 5. The robot-safety system mean time to failure plots for the increasing value of the safety unit repair rate (μ_s).

These plots demonstrate that MTTF$_r$ is greater than MTTF$_{rs}$, but just MTTF$_{rs}$ increases with the increasing value of μ_s.

6. References

Zeldman, M.I., *What Every Engineer Should Know About Robots*, Marcel Dekker, New York, 1984

Ruldall, B.H., *Automation and Robotics Worldwide: Reports and Survey*, Robotica, Vol.14, 1996, pp. 243-251.

Dhillon, B.S., Fashandi, A.R.M., *Safety and Reliability Assessment Techniques in Robotics*, Robotica, Vol.15, 1997, pp. 701-708.

Dhillon, B.S., Fashandi, A.R.M., Liu, K.L., *Robot Systems Reliability and safety: A Review*, Journal of Quality in Maintenance Engineering, Vol. 8, No,3, 2002, pp. 170-212.

Nicolaisen, P., *Safety Problems Related to Robots*, Robotics, Vol.3, 1987, pp. 205-211.

Nagamachi, M., *Ten Fatal Accidents Due Robots in Japan, in Ergonomics of Hybrid Automated Systems*, edited by H.R. Korwooski, M.R., Parsaei, M.R., Slsevier, Amsterdam, 1998, pp. 391-396.

Dhillon, B.S., *Robot Reliability and Safety*, springer, NewYork, 1991.

Gaver, D.P., *Time to failure and availability of paralleled system with repair*, IEEE Trans. Reliab. Vol.12, 1963, pp.30-38.

Grag, R.C., *Dependability of a complex system having two types of components*, IEEE Trans Reliab. Vol. 12, 1963, pp.11-15.

Hahn, Brian D., *Essential MATLAB for Scientists and Engineers*, Oxford: Butterworth-Heinemann, 2002.

Dhillion, B.S., *Design Reliability: Fundamentals and Applications*, CRC, Boca Raton, Florida, 1999.

Corresponding Author List

Kazuhiko Kawamura
Center for Intelligent Systems
Vanderbilt University
USA

I-Ming Chen
School of Mechanical and Aerospace Engineering
Nanyang Technological University
Singapore

Peter Mitrouchev
Integrated Design Centre
Domaine Universitaire
France

Serdar Kucuk
Kocaeli University
Turkey

Tuna Balkan
Mechanical Engineering Department
Middle East Technical University
Turkey

Ibrahim A. Sultan
School of Science and Engineering
University of Ballarat
Australia

Saeed Behzadipour
University of Alberta
Canada

Scott Nokleby
University of Ontario Institute of Technology
Canada

Xin-Jun Liu
Institute of Manufacturing Engineering
Tsinghua University
P. R.China

Spyros G. Tzafestas
Intelligent Robotics and Automation Laboratory
National Technical University of Athens
Greece

Masatoshi Nakamura
Saga University
Japan

Samir Lahouar
Laboratoire de Mécanique des Solides
France

Yung Ting
Chung Yuan Christian University
Taiwan, R.O.C.

Luis T. Aguilar
Centro de Investigación y Desarrollo de
Tecnología Digital (IPN)
México

Muhammad Suzuri Hitam
University College of Science and Technology
Malaysia

Recep Burkan
Erciyes University
Turkey

Abdessemed Foudil
University of Batna
Algeria

Mirosław Galicki
Institute of Medical Statistics
Friedrich Schiller University Jena
Germany

Houssem Abdellatif
Institute of Robotics
Hannover Center of Mechatronics
Germany

Raffaele Di Gregorio
Department of Engineering
University of Ferrara
Italy

Hongliang Cui
Stevens Institute of Technology
USA

Mehrdad Moallem
Department of Electrical & Computer Engineering
University of Western Ontario
Canada

Dan Zhang
University of Ontario Institute of Technology
Canada

Haruki Ueno
National Institute of Informatics
Japan

Friedrich Lange
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Germany

De Xu
Institute of Automation
Chinese Academy of Sciences
P. R. China

Theodor Borangiu
University Politehnica of Bucharest
Romania

Xiao Nan-Feng
Intelligent Systems Research Lab.
Deakin University
Australia

Fusaomi Nagata
Tokyo University of Science
Japan

Satoru Goto
Saga University
Japan

Luis Filipe Baptista
Department of Marine Engineering/IDMEC
Escola Náutica Infante D. Henrique
Portugal

Antonio Visioli
Dipartimento di Elettronica per l'Automazione
University of Brescia
Italy

Alla A. Nesenchuk &
Victor A. Nesenchuk
United Institute of Informatics Problems
of the Belarusian National Academy of Sciences
Belarus

Balbir S. Dhillon
Department of Mechanical Engineering
University of Ottawa
Canada