University of Tennessee, Knoxville
Trace: Tennessee Research and Creative Exchange

Doctoral Dissertations | Graduate School

12-2004

An Adaptive Tool-Based Telerobot Control System

Ge Zhang
University of Tennessee, Knoxville

Recommended Citation
Zhang, Ge, "An Adaptive Tool-Based Telerobot Control System. " PhD diss., University of Tennessee, 2004.
https://trace.tennessee.edu/utk_graddiss/4577

This Dissertation is brought to you for free and open access by the Graduate School at Trace: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of Trace: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a dissertation written by Ge Zhang entitled "An Adaptive Tool-Based Telerobot Control System." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mechanical Engineering.

William R. Hamel, Major Professor

We have read this dissertation and recommend its acceptance:

Jay I. Frankel, J. Wesley Hines, J.A.M Boulet

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Ge Zhang entitled "An Adaptive Tool-Based Telerobot Control System." I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mechanical Engineering.

William R. Hamel, Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Vice Chancellor and Dean of
Graduate Studies

An Adaptive Tool-Based Telerobot Control System

A Dissertation

Presented for the

Doctorate of Philosophy

Degree

The University of Tennessee, Knoxville

GE ZHANG

December, 2004

Thesis
2004b
.Z54

Copyright ©, 2004 by Ge Zhang

All rights reserved

To My Family

Acknowledgements

I wish to thank all those who helped me in completing my PhD in Mechanical Engineering. Especially, I would like to thank my supervisor Dr. Hamel who gave me a chance to study telerobotics and guided me through this work with his experience, knowledge, and patience. This dissertation could not be finished without his encouragement and support. I would like to thank Dr. Frankel, Dr. Hines, Dr. Smith, and Dr. Boulet for serving on my committee and for providing advice and academic guidance.

I would like to thank Mr. Mark Noakes for his friendship and his advice on digital filter design. With his assistance, a major problem in the modeling of tooling interaction forces was resolved. I wish to thank Dr. Sewoong Kim for his help in developing kinematics and Jacobians for the TII manipulator system. I also would like to thank all the students who have worked with me in the Robotics and ElectroMechanical Systems Laboratory: Renbin Zhou, Pamela Murray, Shou Yang, Sriram Sridharan, Karthi Perumal, and Kelley P. Brown.

This research originated from the research project: Robot Task Scene Analyzer and Human-Machine Cooperative Telerobotics and was supported by a Department of Energy grant.

Abstract

Modern telerobotics concepts seek to improve the work efficiency and quality of remote operations. The unstructured nature of typical remote operational environments makes autonomous operation of telerobotic systems difficult to achieve. Thus, human operators must always remain in the control loop for safety reasons. Remote operations involve tooling interactions with task environment. These interactions can be strong enough to promote unstable operation sometimes leading to system failures. Interestingly, manipulator/tooling dynamic interactions have not been studied in detail. This dissertation introduces a human-machine cooperative telerobotic (HMCTR) system architecture that has the ability to incorporate tooling interaction control and other computer assistance functions into the overall control system. A universal tooling interaction force prediction model has been created and implemented using grey system theory. Finally, a grey prediction force/position parallel fuzzy controller has been developed that compensates for the tooling interaction forces. Detailed experiments using a full-scale telerobotics testbed indicate: (i) the feasibility of the developed methodologies, and (ii) dramatic improvements in the stability of manipulator – based on band saw cutting operations. These results are foundational toward the further enhancement and development of telerobot.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Summary

This dissertation describes the development of a methodology to integrate tooling interaction force information into advanced human-machine cooperative telerobotic systems (HMCTR) for operation in complex task environments. An overall telerobotic system architecture is developed for The HMCTR system. With modular design, computer assistance and new control algorithm modules can be easily integrated into the overall control system built on this system architecture. An advanced HMCTR system based on the proposed architecture was built at the Robotics and ElectroMechanical Systems Laboratory at UTK. It provides an excellent test bed for implementing new ideas in telerobotic research area. A universal grey interaction force prediction model is developed in this study for force prediction control, which is the force control loop in the proposed grey prediction force/position parallel fuzzy controller. It utilizes the predicted interaction force information in the control system design. Since there is an integral part designed in the force control loop, along with a fuzzy controller been used in the overall control system, therefore this newly developed control algorithm is robust against environmental disturbances and robot modeling errors, and is capable of handling unexpected contact.

The benefits of implementing the developed control system include: greatly reducing the stress to the human operators during remote operation and increasing the work efficiency and quality of the tool based teleoperation since the system is more stable

and promotes less tool misuse than those of traditional non-tool-based teleoperation. Because this tool based system is fundamentally easier to operate, it reduces training time and thus reduces the total costs.

1.2 Outline of Dissertation

The history and background of the development of telerobotics is introduced in Chapter 2. An overall telerobotic system architecture for advance human-machine cooperative telerobotic system is outlined in Chapter 3. Chapter 4 gives an introduction of grey system theory, and a grey prediction model for interaction force is developed and tested. Basic concepts of force/position parallel control and fuzzy logic control strategy is described in Chapter 5. Also in this chapter, a grey prediction force/position parallel fuzzy controller is developed and simulated by computers using a two-link manipulator model. Chapter 6 describes the HMCTR system hardware and software which serve as the test bed for implementing tooling interaction force control. System experiment results are given in Chapter 7 . Overall conclusions, contributions, and suggestions for future work are outlined in Chapter 8.

Chapter 2

Background

2.1 History

A brief introduction of the history of the development of manipulator systems is given in this section followed by some applications. These introductions are based on the materials in [1] and [2].

2.1.1 Development of Manipulator System

From primitive to current time, telemanipulation, the technical origin of teleoperation, exists in everyday life of human being. The prehistoric man used sticks to dig the ground for animals and roots which were the main sources of food. The BBQ tongs is used to turn over the steaks to avoid burning the hand in a sunny weekend day. These are examples of simplest telemanipulator which can be considered as the extensions of the human hand. Telemanipulation, by using telemanipulator tools, allows work to be carried out remotely by hand in a limited range with the aim of increasing dexterity or avoiding hurt in some circumstances such as turning over the BBQ steaks. Throughout the history of human's adaptation to the natural environment, special tools have been developed for remote handling. The origins of these special tools are sticks and spears used by primitive man to hunt for food or to kill enemies. Later, the first blacksmith performed real telemanipulation by using blacksmith tongs to handle heated metal remotely. The blacksmith tongs can provide three degree of freedom (DOF) in position and three DOF in orientation which make the telemanipulation more advanced.

From then, various forms of remote handling systems have been developed and been used for people who deal with hazardous environment.

Intensive research and development of remote handling systems began around 1940s when scientists in atomic physics were aware of the dangers of irradiation of natural radioisotopes and they also found that the intensity of radiation reduced rapidly against distance (i.e., intensity of radiation proportion to the inverse square of distance). When the radiation source being handled was in low activity, laboratory remote handling tongs with grippers has been created and been used to handle objects at a safe distance up to 50 cm away. It is necessary for researchers to stay behind a protective barrier to manipulate the radioactive materials as the activity of radiation source has increased. The researchers transferred the first radioactive products out of the first nuclear reactor in 1943 by using system of cables, trolleys, pokers and grippers such that these radioactive materials could be handled around at a safe distance. At the end of this pioneering period the remote handling tongs was mounted on shielding walls with a sphere joint so that the scientists could handle radioactive material behind this protective wall and view the process through mirror setup. (Figure 2.1.1) [1]

As experiments and developments in nuclear science became more complicated, it was required that the human operator could perform increasingly complex tasks precisely and safely behind a thick protective barrier. The laboratory remote handling tongs with long handle and mirror setups was no longer effective to accomplish these kinds of research tasks because of the limitations of the remote handling tongs such as working with the mirror image of the scene and the inversion of the movements of the gripper in relation to those of the handle. It was clear that the need for a new manipulation system,

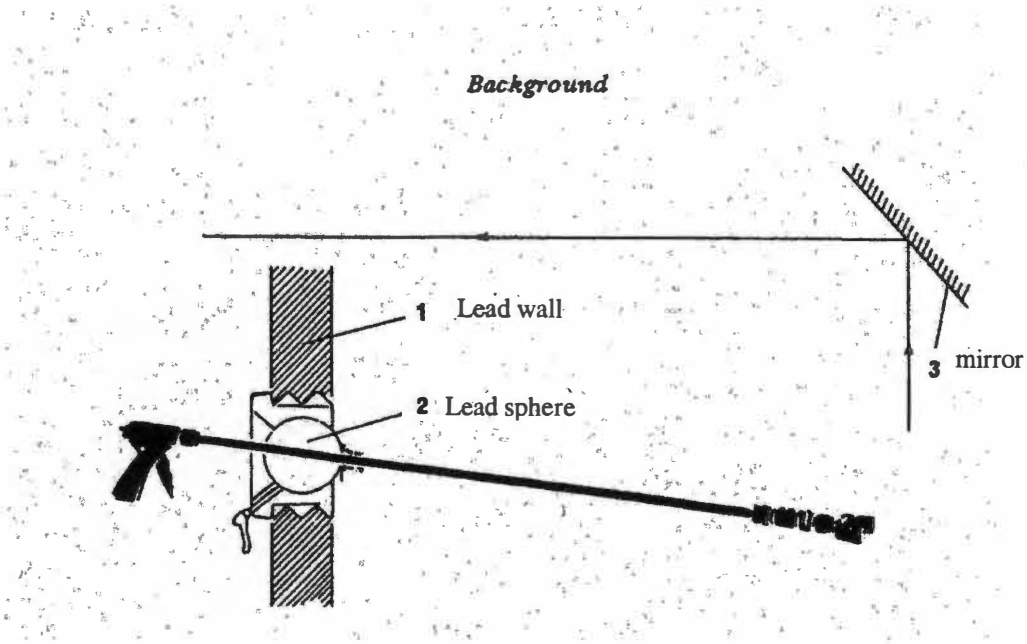


Figure 2.1.1 Sphere mounted remote handling tongs.

which could provide more complete dexterity to handle the radioactive materials, was very imperative. In a period of 15 years from 1947, a group of scientists led by Ray Goertz at Argonne National Laboratory near Chicago, IL, developed much of the telemanipulator technology known today. The first modern mechanical master-slave manipulator (MSM) (Figure 2.1.2) was built in 1948. This manipulator was the ancestor of the thousands of telemanipulators that are used in nuclear, space, underwater, and any other types of hazardous remote operations around the world today. The mechanism developed at that time allowed exact reproduction of the movements of master side on the gripper of the slave side. This let the operator feel comfortable while handling the objects through a lead glass window. The work efficiency of a dual-arm MSM with window viewing is about 5 to 10 times slower than that of direct contact operation by hand using conventional tools. This is considered as a standard of comparison for the performance evaluation of alternative systems by most of the remote handling technologists [2].

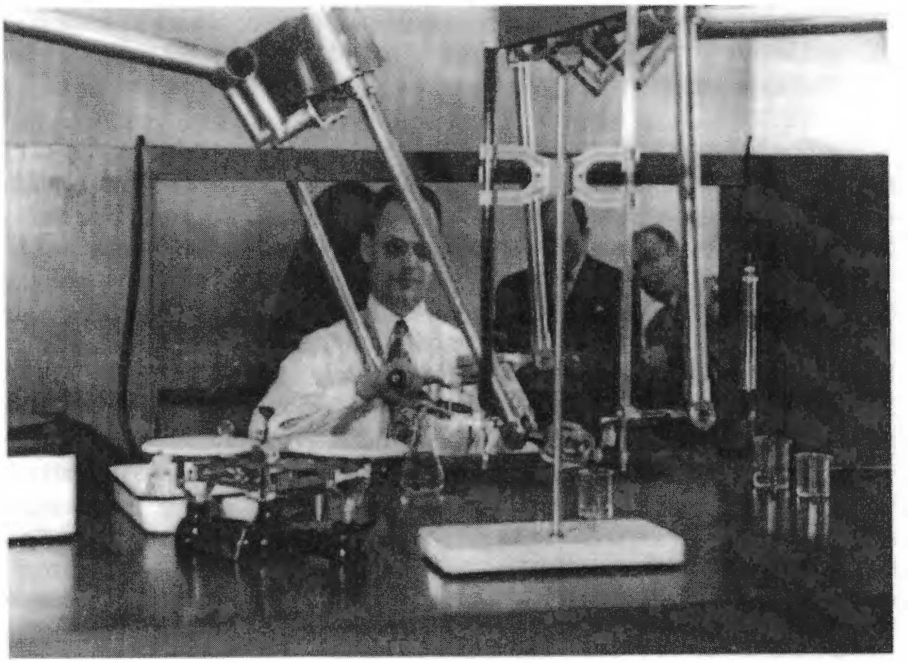


Figure 2.1.2 Mechanical master-slave manipulator (Courtesy of ORNL).

Even though mechanical MSMs can provide noteworthy telemanipulation capabilities, they are very limited to be applied because of their pure mechanical characteristics. The master and slave side are mechanically connected with the metal-tape drive transmission. This limitation makes the physical distance between safe area and the remote hazardous work area less than 10 meters and narrows the application of mechanical MSMs. Due to the limitation of mechanical MSMs, a fly-by-wire MSM, in which the physical separation of the master and slave would not be constrained, is highly desired. This expectation led to the birth of the first electrical master-slave manipulator (EMS), model E1, in 1954. Model E3, an electrical manipulator with bilateral servocontrol, was also developed by Argonne group in 1958 and used experimentally in a radiochemical installation in 1959. The Argonne team made great progress in developing

the integrated EMS system. They developed all the fundamental concepts despite the lack of supporting technology such as electrical control at that time. During that period, there were another two important innovations on master-slave manipulator system development. In 1954, a manipulator was mounted on a vehicle which could run on a caterpillar tracks and could be manipulated by an operator directly or remotely through a television camera. This type of system constitutes the first teleoperator which has mobility different from a telemanipulator, and hence created the concept of teleoperation which is a generalization of telemanipulation in larger space, not only in terms of range but also in terms of mobility [1]. Another advanced innovation at that time was the birth of the first manipulator with force sensor and DC motors in 1965 at Brookhaven National Laboratory. The first force-reflecting servomanipulator system (M2 system), which uses distributed digital electronics to implement position-to-position force reflection with multiplexed serial communications between master and slave, was built jointly by Central Research Laboratories and Oak Ridge National Laboratory. The M2 system (Figure 2.1.3) was used widely to demonstrate complex tasks for nuclear, space, and military applications. It played an important role in demonstrating the potential efficiency of electrical servomanipulator systems for teleoperations in highly uncertain and unstructured task environments [2]. The advanced servomanipulator (ASM) was developed after M2. It used a revolutionary design method, mechanical module design which improved the remote maintainability of the manipulator. ASM was also designed to provide the foundation of telerobotics.

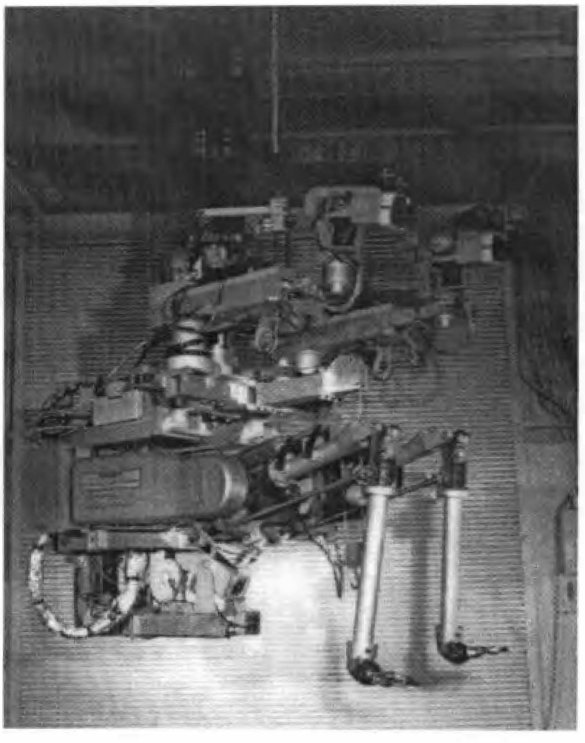


Figure 2.1.3 M2 servomanipulator system (Courtesy of ORNL).

Most of the early developed EMSs used only electric actuators which provide very low power intensity. To increase the power intensity, a group led by Ralph Mosher at General Electric developed an electro-hydraulic servo manipulator with force feedback in 1958. Later, MBA built a prototype of a unilateral servocontrolled manipulator without force feedback in 1969. Wilson constructed a water-operated hydraulic Servoarm with force feedback in 1976 and more recently, Schilling Robotics developed the powerful hydraulic manipulators, such as Titan II and Titan III, which are widely used in subsea applications.

In the late of the period of American effort on developing preliminary master-slave manipulators, Europe began their own research and development led by France and built the first European master-slave manipulator in 1959. Great Britain also developed

MSMs in 1960 and Italian team built their manipulator, Mascot, collaborating with Argonne National Laboratory. The Soviet Union also developed several mechanical master-slave models before 1958 but made no progress for a long period of time.

2.1.2 Development of Telerobotics

In the history of human dealing with hazardous environments, various forms of remote handling techniques have been developed to improve the work efficiency of remote operations. From simple telemanipulation which allows direct remote operation by hand to teleoperation which adds mobility to telemanipulation system. From the standard teleoperation to advanced or computer-aided teleoperation, finally, to telerobotics which integrates automation with teleoperation.

The basics of the technology of teleoperation were created and became general awareness 20 years after the first mechanical master-slave manipulator was built by Goetz's group. The idea of robotics was introduced to practice in late 60s. Around 1970, the first computer-controlled manipulator was born at Purdue University [1]. ASM, mentioned in Section 2.1.1, provided a preliminary foundation for modern telerobotics. It could provide trajectory teach-playback and automated tool-changing functions. Almost about the same time of ASM developed, one of the earliest experimental demonstrations of telerobotic functions was carrying on in France led by Jean Vertut and his teammate using their MA-23 electrical servomanipulator system. The telerobotic functions they tested include computer assists and robotic teach and playback. These were the early concepts of computer-assisted teleoperations. The Draper Laboratory research group at MIT also claimed they created the idea of computer-assisted teleoperation in their NASA project of grapping satellites using mechanical master-slave manipulators.

The history of manipulator development is the history of increasing work efficiency of remote operations. Automation was considered to integrate with teleoperation in order to increase the work efficiency during late 1960s and early 1970s when digital electronics became more cost effective. Since the hostile task environments of most teleoperations are highly unstructured and uncertain, total automation in teleoperations is impossible. Human must stay in the control loop to secure safe operations and only specific subtasks could be executed automatically. Integrating selective automation of specified subtasks with standard teleoperation creates the foundation of modern telerobotics. What consists of a telerobotic system? As explained in [2], a telerobot system, in the context of remote operation in hostile environments, is a combined teleoperator and robot, which is fundamentally a teleoperator but can be modified to execute specific subtasks automatically. Telerobotics has been an active research area since 1970s. New manipulator, control strategy, and control algorithm have been developed to make the telerobotic system robust and hence improve the performance since then. A robust telerobot defined in [2] as a system (1) has effective teleoperability, (2) allows onsite subtask automation, (3) reliably monitor and detect fault conditions, (4) smoothly switch control modes, and (5) can be used in realistic hazardous task environments.

2.1.3 Development of Human-Machine Cooperative Telerobotics

In recent years, much research has focused on human-machine cooperative telerobotics (HMCTR) which incorporates computer assistance using imperfect sensor data to improve work efficiency and to reduce operator fatigue [1,2-7]. A HMCTR system based on the philosophy that the human operator remains at all phases of operation and is

assisted, but never superseded with sensor and model information, was built in the Robotic and ElectroMechanical Systems Laboratory (REMSL), University of Tennessee, Knoxville. The basic concept of this kind of system is shown in Figure 2.1.4 [2].

2.2 Application Domain

Many applications in remote, dangerous, or inaccessible areas, such as nuclear, underwater, space, or microscopic environment (Figure 2.2.1), require remote manipulation and sensing technologies.

2.2.1 Nuclear Applications

Modern teleoperation technologies have first been developed because of the awareness of the dangers of ionizing radiation in nuclear research activities. From the first mechanical master-slave manipulator to the modern telerobotic systems, significant research have been done to improve the work performance of the remote operations and

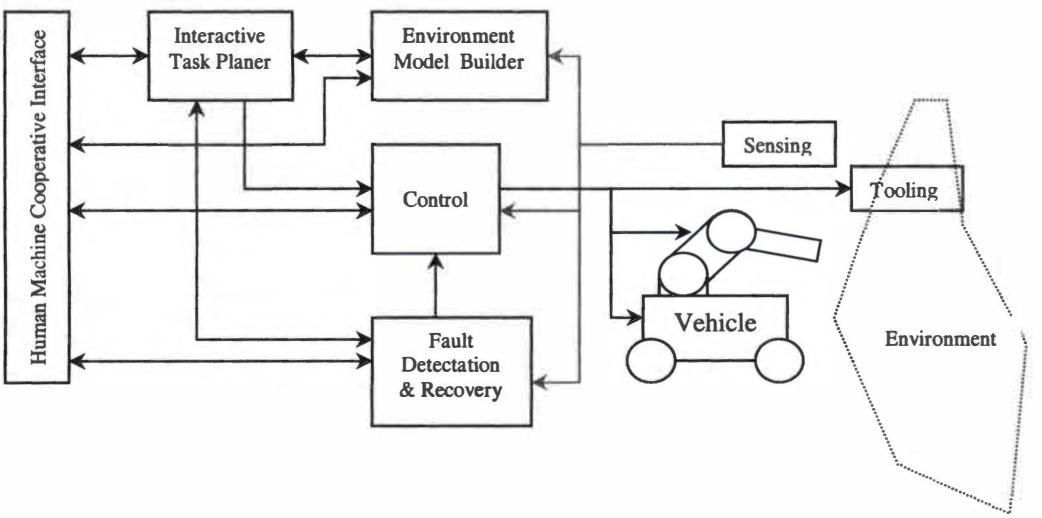
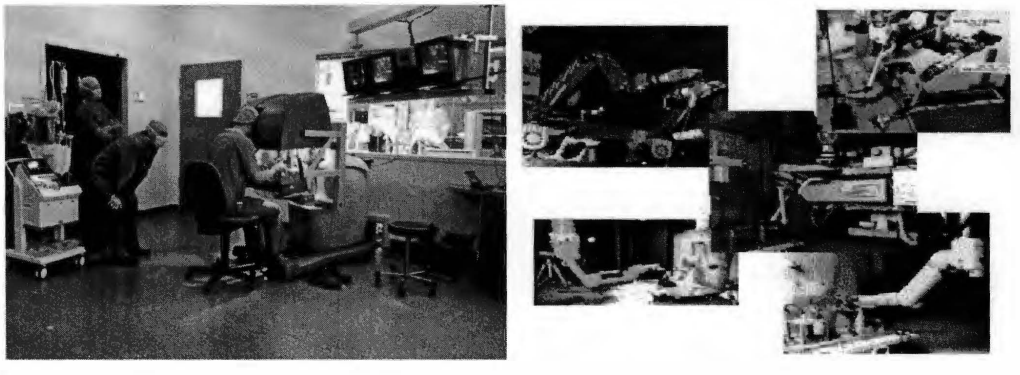


Figure 2.1.4 Component of human-machine cooperative telerobotics.

(a) Space (b) Under Water



(c) Medical (d) Nuclear

Figure 2.2.1 Typical remote operations.

to protect the nuclear researchers and workers from radiation. In current years, it is imperative in nuclear domain to improve the remote manipulation efficiency since there are hundreds of contaminated and inoperative facilities such as hot cells, reprocessing canyons, glove boxes, and reactor facilities exist at U.S. Department of Energy (DOE) sites. Many of these facilities are no longer satisfying the current environmental regulations. In order to meet the current regulations and eliminate the potential hazards to the public and environment, the contaminated facilities will need various levels of decontamination and decommissioning (D&D) in the future. The typical D&D manipulations include equipment disassembling, pipe cutting, decontamination of

equipments and parts before removal, and transporting the decontaminated equipment and parts out of the facilities. Because of high level radiation and harmful chemical hazards within or on the surface of these contaminated and inoperative facilities, many of these remediation activities will have to be performed remotely in order to protect human workers from dangerous exposures in work sites. The hazardous and unstructured natures of these remote operations make the remote manipulation complex, inefficient, and tedious. It is much slower than direct contact manipulation [2]. In order to improve the work efficiency and quality, it is highly desirable to perform the remote operation autonomously wherever applicable. But the variety and variability in the remote environment makes it almost impossible to execute the remote operation fully automatic even with the best teleoperation system in which human beings are an integral part of the control and decision making loop. Even though an experienced operator can finish most remote operation tasks with up-to date teleoperation system, these advanced systems are hard to manipulate, and make simple remote operations tedious and time consuming [3].

2.2.2 Teleoperation in Space Operations

NASA began their research in teleoperation in late 1960s. The first example of teleoperation in space took place in 1967. The first manipulator arm mounted on Surveyor III and was sent to the moon to take lunar soil sample. Ray Goertz at that period was also in charge of a NASA project which emphasized the grasping of satellite in outer space by using mechanical master-slave manipulators. Following his research, the team in Draper Laboratory at MIT developed the idea of computer-aided teleoperation. Being motivated by overcoming the major disadvantage of the teleoperation in space, which is

the time delay in transfer information, several groups from USA began to study the transmission delay effect and computer control at the same time. Their research led to the birth of an advanced manipulator system installed on Viking spacecraft, which landed on the surface of Mars in 1976. This manipulator system could be programmed to solve a certain number of problems by its own computer, and it also could take commands from human operators on earth. This represents the meet of teleoperation and robotics [1]. Currently the Canadarm mounted on US space shuttle and the Dexterous arm mounted on International Space Station represents the state of the art telerobot system in space application.

2.2.3 Teleoperation in Underwater Operations

The research in underwater teleoperation began in early 1960s. In 1961 a telemanipulator was installed on a deep sea submarine, the Bathyscaphe Trieste, for undersea exploration. During 1960s the US Navy developed a series of manipulators and unmanned cable-controlled vehicles for underwater operations. The most famous underwater operation at that time was retrieving a nuclear bomb from the bottom of the sea near the coast at Palomares in Spain by using the cable controlled underwater research vehicle. Presently, most of the underwater operations exist in the construction and maintenance of offshore oil wells.

2.2.4 Teleoperation in Medical Applications

Medical robotics, such as telesurgery, laparoscopic surgery [10], orthopedic surgery [11], microsurgery and stereotactic neurosurgery [12], along with radiotherypy [13], are promising applications of robotics. Among them, telesurgery is a typical

application of teleoperation in medical field. Teleoperation can be useful in such a case that a patient imperatively requires an operation in place where no specialized surgeon is available (e.g. on a battlefield, or in some rural area). It is also useful when the operation area is not safe to the surgeon because of the infectious diseases, or long operation under radioactive rays [14]. The first use of hospital telerobotic assisted surgery was performed in Canada by Dr. Mehran Anvari, founding director of the Center for Minimal Access Surgery (CMAS), and Dr. Craig Mckinley, a general surgeon at North Bay General Hospital [15]. On February 28th, 2003, Dr. Anvari in CMAS, located at St. Joseph's Healthcare Hamilton, successfully collaborated with Dr. Mckinley to complete a laparoscopic surgery on a female patient located at North Bay General Hospital, nearly 400 kilometers away.

2.3 Literature Review

Manipulator control can be conceptually divided into two categories: position control and force control. Position control has been the major and most successful control strategy of commercial applications. It has also been widely used in non-industrial applications such as teleoperation in hostile environments. Since real manipulation tasks, such as grasping, cutting, drilling, contour-following operation, and assembly of mechanical parts, involve interaction of the manipulator with the environment, therefore large contact forces may occur. Traditional position control system is not suitable for controlling the manipulator when it contacts with the environment because it tries to follow the desired position trajectory and attempts to reject the contact forces as disturbance. This introduces larger interaction forces and

causes saturation, instability, and physical damage [8]. Force control techniques have been developed by researchers and engineers since the early day of manipulator development. They have always been an active research area. Fundamental issues have been extensively addressed by many researchers in the literature: Impact analysis and control [9,16-20], compliant control [8], [21-23], [33-37], force/position control in constraint motion [24-32], impedance control [38]-[47], Sliding mode control [59], Parallel control [55-58], and fuzzy neural network force control [48-54].

In the following sections, general force control architecture and some basic force control techniques will be discussed briefly.

2.3.1 Overall Force Feedback Architecture

Figure 2.3.1 illustrates the general force feedback control architecture that most the force feedback control systems follow. The desired motion commands can be desired force, velocity, or position depending on whichever control method being selected.

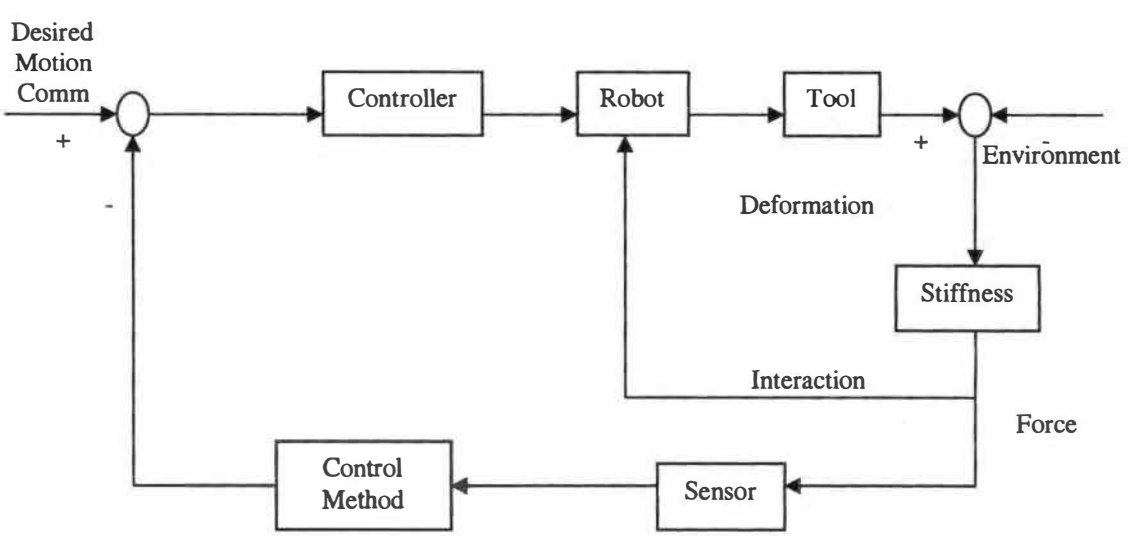


Figure 2.3.1 Overall force feedback architecture.

2.3.2 Impact Force control

In remote manipulation, the manipulator moves from free space to contact with environment, and fast speed generates high productivity, but high speed may cause the end-effector and tool to collide with the environment causing damage to the tool and/or manipulator. Impact force control [60] handles the problems of force regulation and contact transition stabilization. The transient force response during impact can be controlled to limit peak impact force and the system can be stabilized by the impact force control using a suitable control strategy.

There are two ways to describe impact phenomena. The first one is to model the local dynamics of the object or the relation between the deformation and the impact. The other way is to calculate the speed change after collision and neglect the behavior during the impact. In order to avoid damaging the tool or the telerobot, it is assumed that only elastic collisions are allowed. The Hertz impact theorem [61] provides a good elastic model of the collision. In this model, the normal force is:

$$F = k\alpha^{3/2} \tag{1}$$

where

α – the deformation of the environment

$$k = (k_1 + k_2)\frac{4\sqrt{\bar{r}}}{3\pi}, \qquad \bar{r} = \frac{r_1 r_2}{r_1 + r_2} \tag{2}$$

$$k_i = \frac{E_i(1-\nu_i^2)}{\pi}, i = 1,2 \tag{3}$$

ν_i – Possion's ratio

E_i – Young's modulus

r_1, r_2 – radii of the contact masses

The upper bound of the impact velocity:

$$V_0 = \min(\frac{10.7m^2}{k^2 T_s^5}, 0.89 m^{-1/2} k^{-1/3} F_{limit}^{5/6}) \quad (4)$$

m – the mass of end effector and tool

T_s – the sampling time

Even though the Hertz impact theorem provides a basic idea of the limitation of the impact velocity, the real application is more complicated. For example, to calculate the impact velocity, it requires a model of the environment, but in remote operations it is difficult to have the accurate model of the environment before hand.

2.3.3 Impedance Force Control

The impedance force control [39] scheme is based on the assumption that the environment can be modeled as an admittance for any manipulation task. For admittance, force is the input and motion is the output. Because two interactive systems must complement each other, the manipulator system must behave like an impedance system which accepts motion inputs and yields force output (Figure 2.3.2). The desired mechanical impedance for the end-effector is given by

$$M_d \ddot{\bar{r}} + C_d \dot{\bar{r}} + K_d \bar{r} = \bar{f} \quad (5)$$

where M_d, C_d, and K_d are the desired inertia matrix, damping-coefficient matrix, and stiffness matrix respectively. The end-effector position is denoted by $\bar{r}$ and $\bar{f}$ is the external force vector. The mechanical impedance, the feedback gain from the position error to the force, is made large for the directions in which position should be controlled

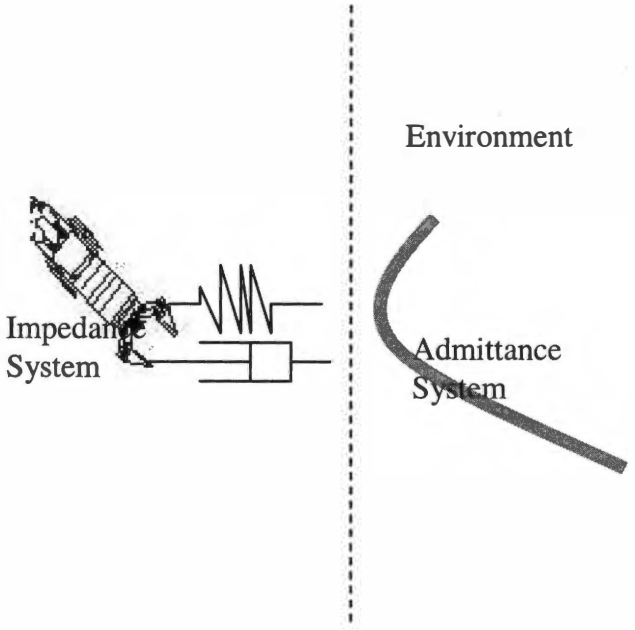


Figure 2.3.2 An impedance system.

accurately in order to reduce the effect of external force. On the other hand, the impedance is made small for the direction in which it is desirable to avoid pushing too aggressively on the environment. The major advantage of impedance control is that it builds an adjustable balance of the system between external force and position errors. However, the impedance control cannot specify and control both position and force variable, and force regulation cannot be achieved without accurate environment model.

2.3.4 Hybrid Force/Position Control

The basic concept of the hybrid control approach is that the controller selects the directions in which the position of the end-effector should be controlled and the directions in which the force exerted by the end-effector on environment should be controlled. Based on the assumption that the position and force control directions are

orthogonal, a selection matrix is used to select the proper contribution to the control law for each task coordinate. However, a major drawback of the hybrid approach is that the control structure must be changed according to each phase of a given task. This requires detailed knowledge of the environment. Figure 2.3.3 is a conceptual architecture of hybrid controller introduced in [29].

2.3.5 Parallel Control

The parallel approach to force/position control of manipulators attempts to combine simplicity and robustness of the impedance and admittance method with the ability of controlling both position and force variables of the hybrid control approach. The parallel approach uses two controllers acting in parallel and manages the conflicting situations between force and position control using a priority strategy: The force control loop is designed to prevail over the position control loop because the primary objective of

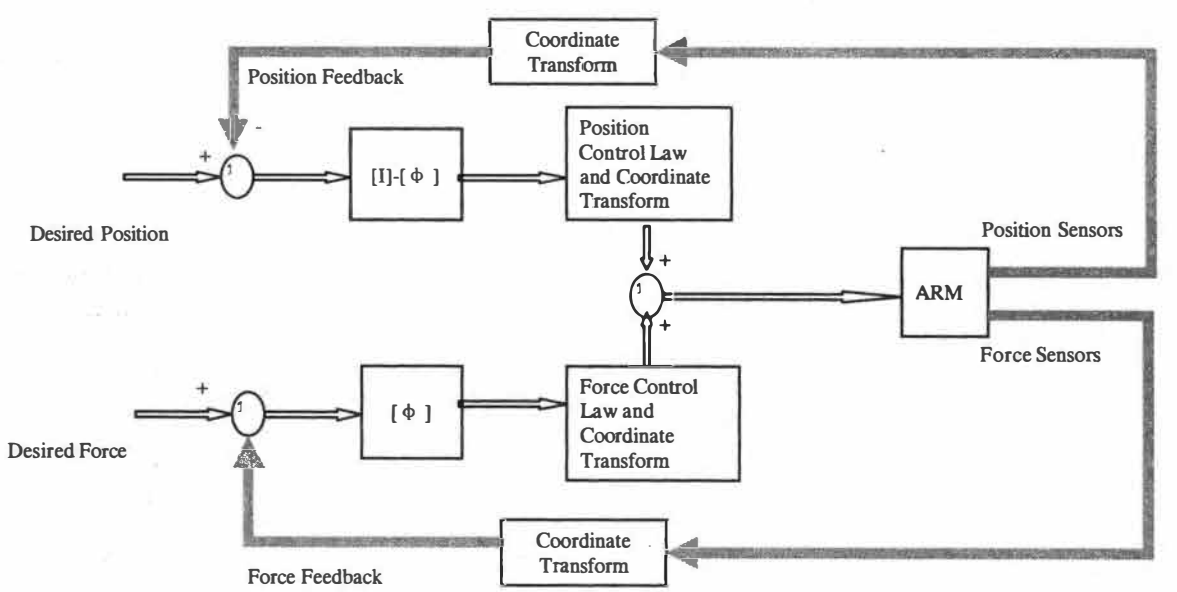


Figure 2.3.3 Conceptual architecture of hybrid controller

this approach is to make system capable of accommodating its motion to environment constraints. Because of the integral control over the force error, the parallel approach is not so sensitive to environment modeling errors and has the ability to handle unexpected contact [58] (Figure 2.3.4 is a duplication from [58]).

2.3.6 Fuzzy and Neural Network Control

The manipulator system is a highly nonlinear and cross-coupled dynamic system. There are structured and non-structured uncertainties in the system. The typical remote task environments are highly unstructured. It is almost impossible to get accurate models of the manipulator system and the environment. All of these factors make it difficult to develop a telerobot controller using the traditional feedback control design techniques. Fuzzy systems deliberately make use of vague, imprecise or uncertain information to

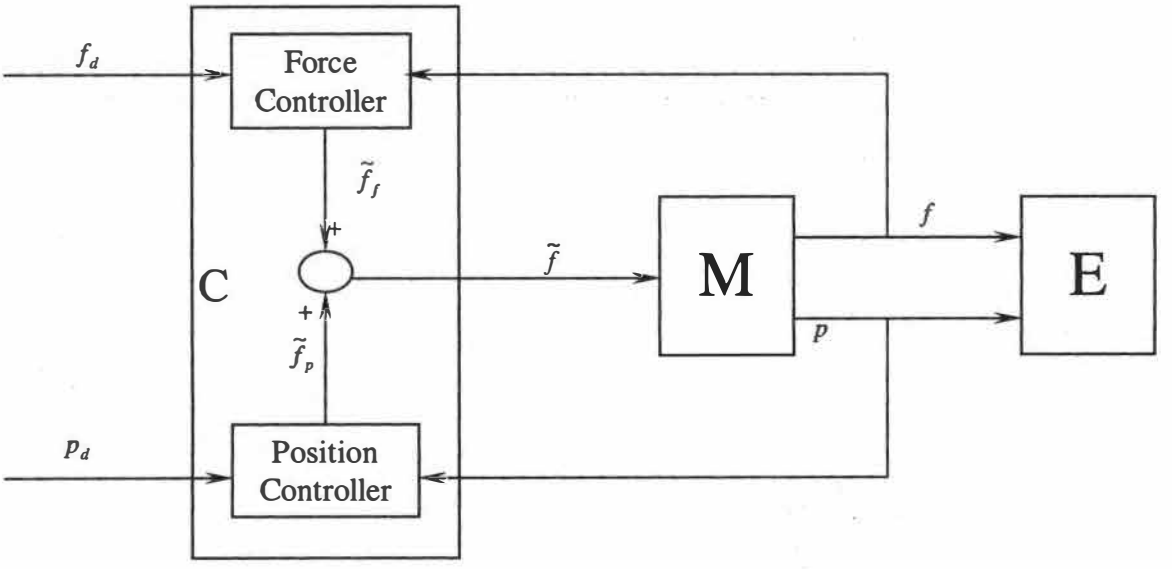


Figure 2.3.4 Basic architecture of parallel controller

develop models in a manner similar to human thinking. Fuzzy manipulator control has been intensively studied [63-67]. In fuzzy systems, expert knowledge can be relatively easily expressed using linguistic if-then rules with antecedents and consequents, however, tweaking membership functions can be difficult and time consuming.

Probably the most innovative technical development in the last decade in the field of control is the introduction of artificial neural networks (NN's). A foundation for neural networks in control was provided in seminal results by Narendra *et al.* Werbos and others [62]. The applications of NN in feedback control systems have been intensively studied recently [68]-[73] due to the following advantages in applying neural networks to control domain [74]: First of all, they are suited for the control of nonlinear systems because neural networks can flexibly and arbitrarily map nonlinear functions,; second, neural networks are particularly well suited to multivariable applications since they can map interactions and cross-couplings readily while incorporating many inputs and outputs; third, neural networks can either be trained offline and subsequently deployed online, or they can be trained online as a part of an adaptive control scheme; finally, neural networks are inherently parallel processing devices that have fault tolerant characteristics, fast data processing, and can be structured for graceful degradation. But the drawbacks of NN are the inability of integrating knowledge into or extracting from it because it acts as a black box to users.

2.4 Motivation

In real applications of telerobotics, the interaction between the manipulator system and environment can be divided into two parts: interaction between the tooling

and environment and interaction between manipulator and tooling as shown in Figure 2.4.1. But most current research only considers the interaction between manipulator system and the environment giving essentially no consideration to tooling effects. This approach may be suitable for manipulator systems which use high stiffness unpowered tools. Typical tools used in remote operations fall into two categories: unpowered and powered. The unpowered tool includes passive devices such as wrenches, and the powered tools include saws, drills, and impact wrenches etc. The interaction between powered tools and environment can be very strong and can affect the overall dynamics of the entire telerobot system. As we have noticed in pipe cutting experiments with the HMCTR system in REMSL, sometimes these interactions can be severe enough to make the manipulator control system unstable. Motion oscillations under these conditions can reduce the tool efficiency and even cause tool failure or damage the manipulator system. A comprehensive literature review has revealed a major gap exists no investigations have considered generalizing tool-environment dynamic interaction into the overall telerobot control system design.

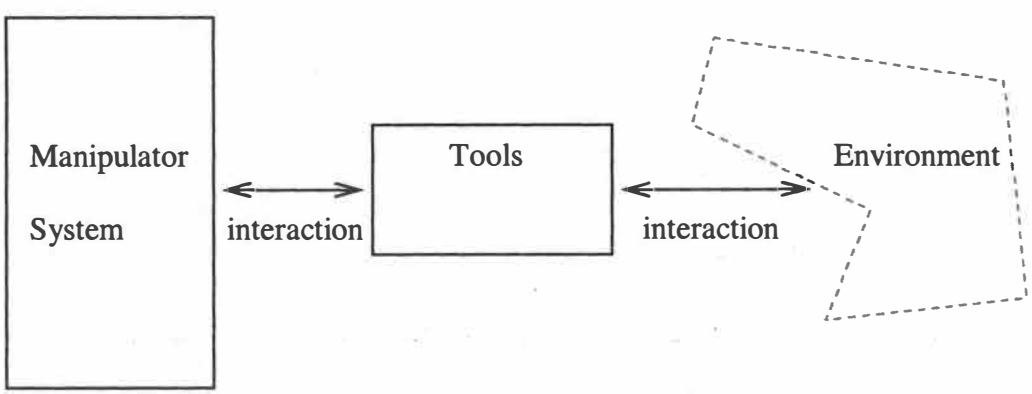


Figure 2.4.1 Manipulator system interactions

Chapter 3

Telerobotic System Architecture

3.1 Introduction

Telerobotic system has been studied and developed for more than 30 years. Many efforts have been focused on improving the work efficiency. An integrated telerobotic system must keep the human operator stay at all phases of operation and be assisted but never superseded by computer control. In this chapter, an overall architecture of advance telerobotic system which includes the task environment as an integral part will be created and discussed in detail.

3.2 Telerobotic System Architecture

Telerobotic system has been successfully applied to nuclear facility maintenance and cleanup, underwater and space operations, and microsurgery etc. A advanced telerobotic system should improve the overall work efficiency of remote operations thus it must be easy to operate, allow onsite programming of subtask automation, seamless control modes switching, online fault detection and correction. As shown in Figure 3.2.1, a telerobotic system consists of human operators with human machine interface, operating software which is computer controller, hardware interface which creates communication between the slave systems in remote task space and human-machine interface, and finally slave system in a remote work space.

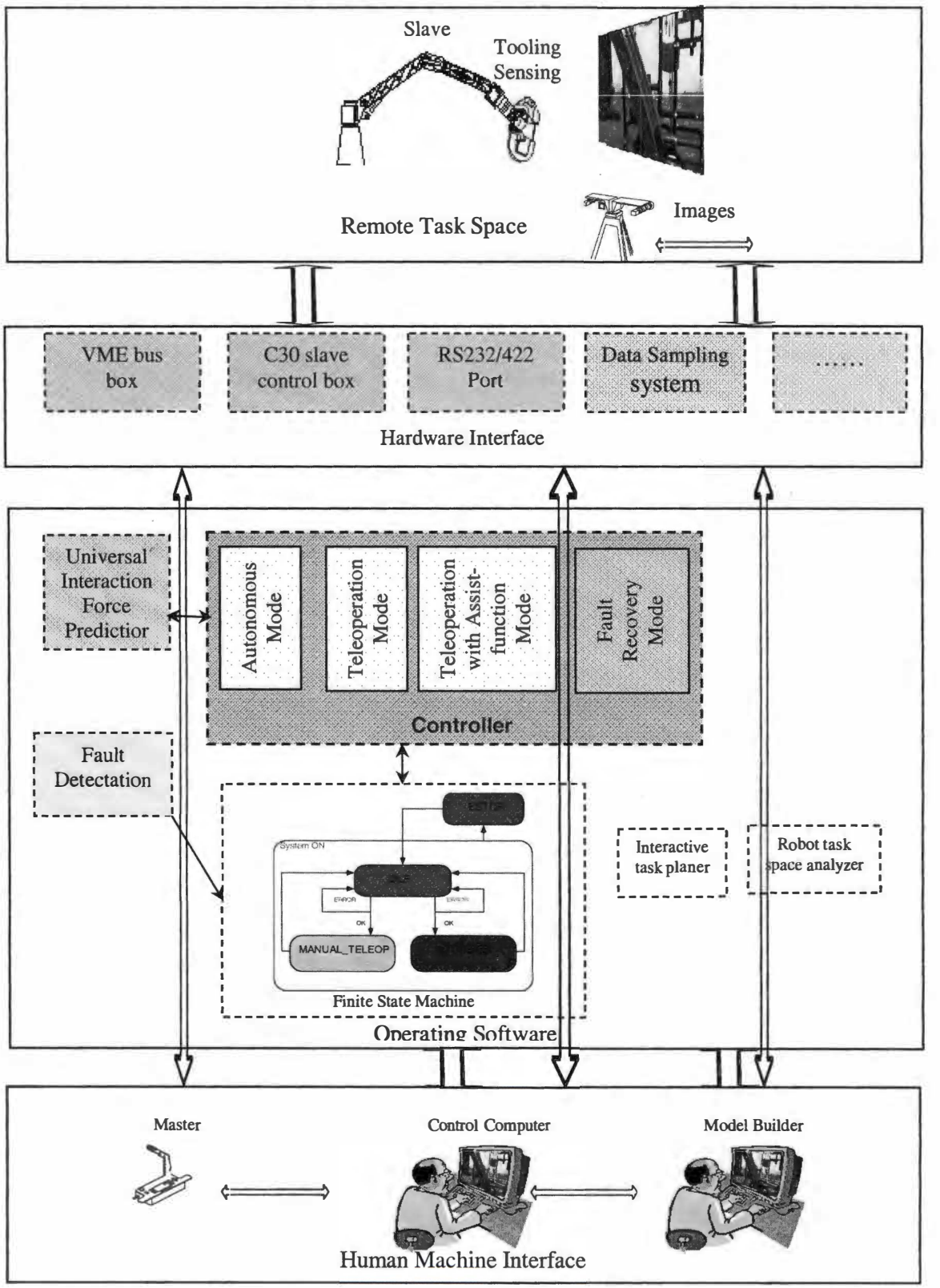


Figure 3.2.1 Overall telerobotic system architecture

Human-machine interface

A human-machine interface of a telerobotic system includes a human operator of cause, a master manipulator which can be used by the operator to control the slave manipulator directly, a control computer which can take input commands from master manipulator or task plan information from model builder computer and generate control commands to the slave manipulator, and a model builder computer which analyzes the remote task space and generates the computer models of the task environment which can be used in subtask automation. A proper designed human-machine interface is significant to a integral telerobotic system since it is the glue that keep the telerobotic system together in the sense of flexibility and robustness [2]. No matter how good the telerobot is, the system will not perform accurately unless the human-machine interface is well defined. A poor human-machine interface will introduce mental stress to the operator and lower the system performance.

Hardware interface

Hardware interface consists of cables, VME bus, slave manipulator control box, RS232/422 port, and data sampling system, etc. It creates communication between master controller, computers, and slave manipulator and sensors.

Slave manipulator system in remote task space

The slave manipulator system in remote task space includes a slave manipulator system, tooling system, and sensing system.

Operating software

Computer assistant control of telerobotic system has become one of the most important issues of modern telerobotics. Operating software is the core of the computer

assistant control of telerobotic system. It usually consists of two major parts, telerobot control system and remote task space modeling system.

Since remote task space is usually unstructured and complex, model the entire operational space at a time is almost impossible. The remote task space modeling system only models the region of interest (ROI) which selected by the human operator during teleoperation. The generated model is used by the task planner to generate a task plan which can be executed by the telerobot control system automatically.

A general control system of a telerobot usually has a finite state machine (FSM) and a continuous controller. FSM receives inputs from the human operator, task plan, and feedback information and then issues stimuli to activate different control modes. The continuous controller basically has four control modes such as pure manual teleoperation mode, subtask autonomous mode, teleoperation with assistant function mode, and fault recovery mode. Fault detection, isolation, interpretation, and interactive recovery strategies are very important in modern telerobotic system. A robust telerobotic control system in real remote operation should also consider the interaction between tooling and the task environment. This introduces a universal interaction force predictor and predictive force/position control into the control system of telerobot which are main contributions of this study.

Chapter 4

Modeling Tool Interaction

4.1 Introduction

Modeling or estimating tool interaction is complicated and difficult. Both dynamics of the tool system and the environment should be known in priori. It is almost impossible to develop a dynamic model of teleoperation task environment because it is highly unstructured and uncertain. Mathematical modeling of a tool system requires detailed knowledge of each specific tool system. Bandsaw dynamics was developed and analyzed in [75-77]. Analytical models in different drilling process were built in [78-81]. These mathematic models are highly nonlinear systems and require extensive computation to solve them. There are also some non-analytical methods for modeling dynamic system such as least-square regression, neural networks, and fuzzy logic. Least-square regression and neural network techniques use experimental data to build the input-output relationships of the dynamic system. They require large amount of data to get a model to better describe the system within the range of training data. Any prediction well beyond the range of training data is not trustful and therefore is not useful. Fuzzy inference system is based on fuzzy if-then rules which are mainly from expert knowledge rather than experimental data set. But it is difficult to find the appropriate membership functions and fuzzy rules, especially when the system is complicated and rapidly changing. All these drawbacks impede the implementation of these techniques for modeling tool interactions in telerobotic system.

4.2 Grey Prediction Model

Grey system theory is relatively new. It was first introduced by Professor Julong Deng to deal with systems with poor information in 1982. A grey system is one in which part of the information is known and part is unknown. It has been widely applied to control, medical, agriculture, military, and engineering problems [82-86]. Any random variation in a system is treated as the variation of the grey value within a certain range and any random process is consider as time-varying grey process by the grey system theory. Instead of using a statistical model, grey theory uses grey generating techniques such as accumulated generating operation (AGO) to turn the stochastic raw data into a more regular series. Grey prediction is one of the most important cores of grey system theory. It utilizes past and current known or indeterminate information to establish a grey model to extend the past information to the future so that the grey model can be used to predict future variation tendency of the system output. The key operation in the construction a grey model is the use of discrete time sequence data to build up an ordinary differential equation. Accumulated generating operation and inverse accumulated generating operation (IAGO) are basic tools for finding the grey differential model. The general form of a grey differential model is GM (n, m), where n is the order of the ordinary differential equation of grey model and m the number of grey variables. The variables n and m define the order of AGO and IAGO, and the computing time increases exponentially as n and m increases but prediction accuracy may not be improved with large n or m values. Hence, the most extensively used grey prediction model is GM (1, 1) [87]. Compare to other non-analytical method such as neural network, regressive analysis etc., grey modeling has the following characteristics: (i) small data set

(ie. 4-6 data), (ii) less computation, (iii) online real time modeling, and (iv) any kind of data distribution

4.2.1 Accumulated Generating and Inverse Accumulated Operation

In the view of grey system theory, the concept of accumulated generating operation is an important idea which has the ability to turn the stochastic raw data to a regular series.

Let $\{x^{(0)}(k)\}$, $x^{(0)}(k) \geq 0$ and k=1, 2, …, N is any time sequence data. It is irregular as shows in Figure 4.2.1. The accumulated generating operation is:

$$x^{(1)}(k) = \sum_{i=1}^{k} x^{(0)}(i) \qquad (4.2.1)$$

where $x^{(1)}(k)$ is accumulated generating sequence data. As shown in Figure 4.2.2, it increases monotonically. If $x^{(0)}(k)$ is a negative sequence, common in real applications, some mapping methods such as exponential and linear mapping techniques are needed to map the data to a positive area. Generally, one AGO operation makes data show some regularities. If it is not enough, AGO is repeated until the data become regular:

$$x^{(m)}(k) = \sum_{i=1}^{k} x^{(m-1)}(i) \qquad (4.2.2)$$

where m is the number of AGO operations.

Inverse accumulated generating means transforming the AGO sequence data back to raw data sequence:

$$\begin{aligned} x^{(0)}(1) &= x^{(1)}(1) \\ x^{(0)}(k) &= x^{(1)}(k) - x^{(1)}(k-1) \end{aligned} \qquad (4.2.3)$$

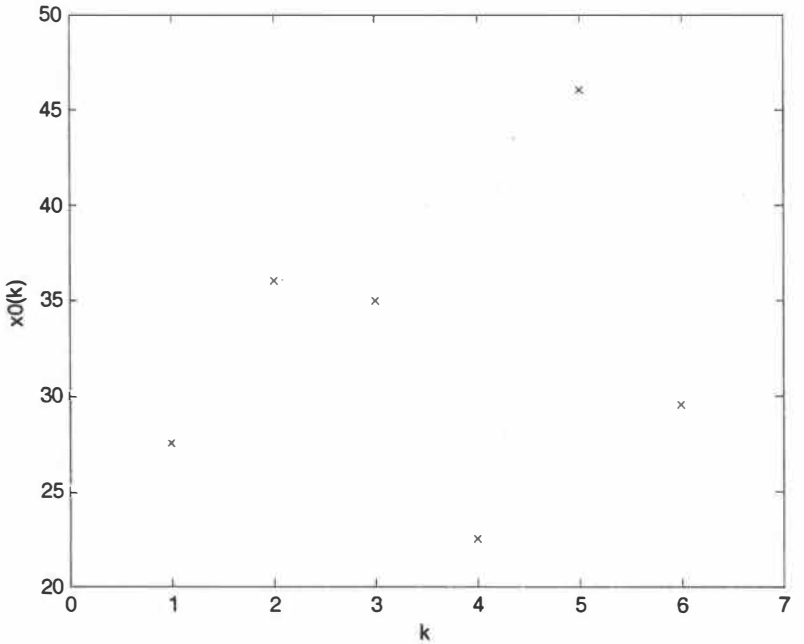


Figure 4.2.1 Raw data series.

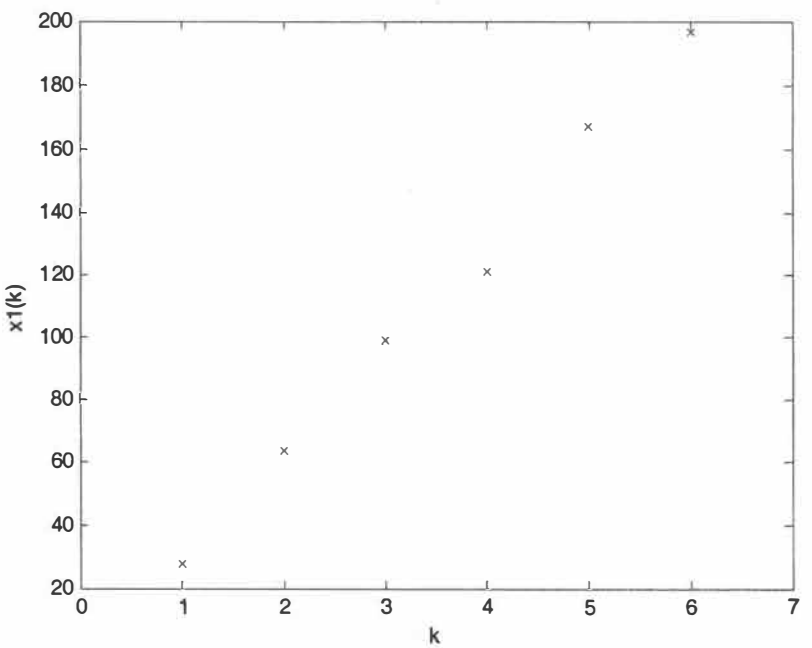


Figure 4.2.2 AGO sequence data.

4.2.2 GM(1,1) Prediction Model

Since the AGO sequence data increases monotonically, it can be approximated by an exponential function which has the dynamics of a first-order differential equation. The first-order differential grey model GM(1,1) is:

$$\frac{dx^{(1)}}{dt} + ax^{(1)} = b \qquad (4.2.4)$$

where a is the developing coefficient and b is the grey input. Set the sampling interval as one unit and then the first derivative of the AGO sequence $x^{(1)}$ becomes:

$$\frac{dx^{(1)}}{dt} = x^{(1)}(k) - x^{(1)}(k-1) = x^{(0)}(k) \qquad k = 2,3,\ldots,N \qquad (4.2.5)$$

and let $z^{(1)}(k)$ be the average of $x^{(1)}(k)$ and $x^{(1)}(k-1)$:

$$z^{(1)}(k) = \frac{x^{(1)}(k) + x^{(1)}(k-1)}{2} \qquad (4.2.6)$$

Next approximate the second term of equation (4.2.4) by (4.2.6) and substituting (4.2.5) into it, the first-order grey differential model has been established:

$$x^{(0)}(k) + az^{(1)}(k) = b$$

$$x^{(0)}(k) = b - \frac{a}{2}[x^{(1)}(k) + x^{(1)}(k-1)] \qquad (4.2.7)$$

coefficients a and b can be calculated by using a least-square technique from the raw time sequence data $x^{(0)}(k)$ and AGO sequence data $x^{(1)}(k)$ since at least 4 sets of data are needed to predict the trend of this grey model in order to get an approximate growing of $x^{(1)}(k)$.

According to the least-square method, the corresponding matrix equations are:

$$\vec{\hat{c}} = \begin{bmatrix} a \\ b \end{bmatrix} = (B^T B)^{-1} B^T \vec{Y}_N \qquad (4.2.8)$$

$$\vec{Y}_N = [x^{(0)}(2), x^{(0)}(3), \ldots, x^{(0)}(N)]^T \qquad (4.2.9)$$

$$\mathbf{B}=\begin{bmatrix} -\frac{1}{2}[x^{(1)}(2)+x^{(1)}(1)] & 1 \\ -\frac{1}{2}[x^{(1)}(3)+x^{(1)}(2)] & 1 \\ \vdots & \vdots \\ -\frac{1}{2}[x^{(1)}(N)+x^{(1)}(N-1)] & 1 \end{bmatrix} \qquad (4.2.10)$$

here, N is the number of data sets used to calculate the grey model coefficients and $\vec{Y}_N$ is the raw data sequence from time interval 2.

From equation (4.2.1), $x^{(1)}(1)=x^{(0)}(1)$. The solution of the grey differential equation GM(1,1) could be written as [82]:

$$\hat{x}^{(1)}(1) = x^{(0)}(1)$$
$$\hat{x}^{(1)}(k+1) = (x^{(0)}(1) - \frac{b}{a})e^{-ak} + \frac{b}{a} \qquad (4.2.11)$$

and therefore, the calculating value of $x^{(0)}$ is determined as follows:

$$\hat{x}^{(0)}(1) = x^{(0)}(1)$$
$$\hat{x}^{(0)}(k+1) = \hat{x}^{(1)}(k+1) - \hat{x}^{(1)}(k) \qquad (4.2.12)$$

where $\hat{x}^{(1)}(k)$ and $\hat{x}^{(0)}(k)$ are estimating value of $x^{(1)}(k)$ and $x^{(0)}(k)$ at point k, respectively.

4.3 Modeling Bandsawing Interaction Forces with Environment

As mentioned in Section 4.1, it is almost impossible to establish an accurate mathematical model for tool interaction with environment in telerobotic tasks since the task environment is unstructured and uncertain and it is hard to get the direct information

from the task environment because it is unreachable or hazardous to human operator. Bandsaw is one of the typical cutting tools using in teleoperation. The interactions between the teeth of the blade and environment can be very severe. Here GM(1,1) model is used to predict the interaction forces and torques of a cutting process.

4.3.1 Multi-Axis Force/Torque Sensor System

Some sampling data is needed to test the GM(1,1) model for predicting interaction force and torque. A multi-axis force/torque sensor system was built (Figure 4.3.1). It includes a Theta US-300-1800 F/T transducer from ATI Industrial Automation, a NI-DAQ 6034e sampling card from National Instruments, a power supply box, and computer system with RTAI Linux operating system installed. The RTAI, real time application interface, empties the kernel of Redhat linux system and makes it become a hard real-time operating system so that the sampling rate can reach 1000 Hz. The Theta US-300-1800 transducer is made of hardened stainless steel, and the standard mounting adapter is made of high-strength aircraft aluminum. Its measuring ranges are $\pm 300 lbf$ in x, y axis, $\pm 875 lbf$ in z axis, and $\pm 1800 in-lb$ about x, y, z, respectively. It also has overload protection and the maximum allowed overload value are 6.1 to 20 times rated capacities. All these features make it applicable for rough applications with high load interactions such as the teleoperation in dismantling and decommissioning missions in the nuclear area.

4.3.2 Preprocess the Sampling Data

The data used to test the GM(1,1) model were sampled at 1000 Hz. They cover the whole cutting process from picking up the saw to cutting the steel 2' pipe, and storing

Figure 4.3.1 Bandsaw with force/torque senor mounted

the saw. Figure 4.3.2 shows the raw data of interaction force in z axis of the transducer. It looks noisy so a fast Fourier transform analysis has been applied. The result is shown in Figure 4.3.3. It is found that there are some high frequency components in the range 200 – 400 Hz. Since the natural frequency of the manipulator system is very low, it is basically a mechanical low pass filter and does not respond to higher frequency disturbances. An on-line low pass filter is needed.

A moving average filter was built to filter the raw data because it is simple, fast, and can be used in on-line filtering. A moving average filter can be modeled mathematically as shown in equation (4.3.1):

$$y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k] \qquad (4.3.1)$$

where y is the filtered signal, x is the input signal, M is the number of input samples for the filter to take in for averaging and M =20 is determined by experiment.

Figures 4.3.4 and 4.3.5 show the filtered data and the frequency spectrum of the filtered data. They show that the high frequency components in raw data have been removed.

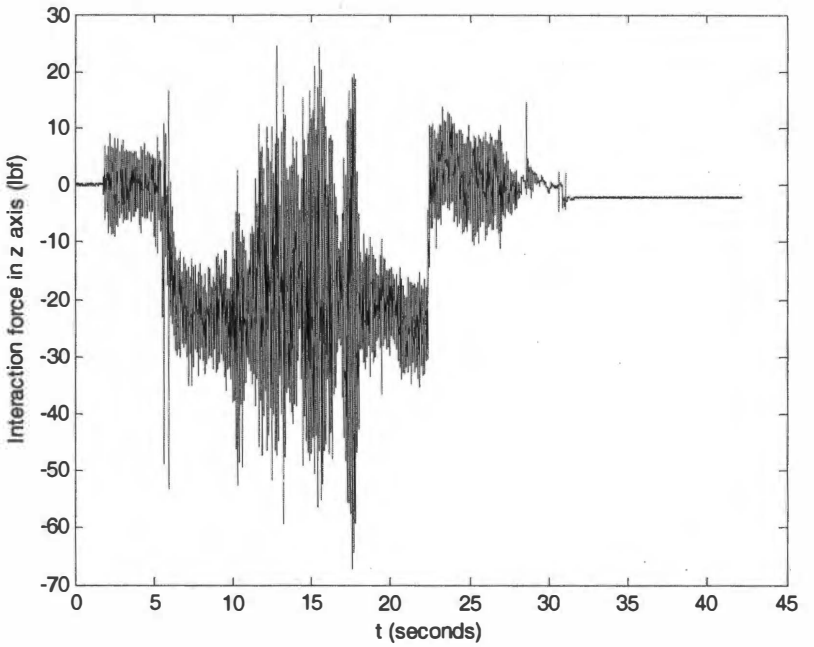


Figure 4.3.2 Raw interaction force in z axis.

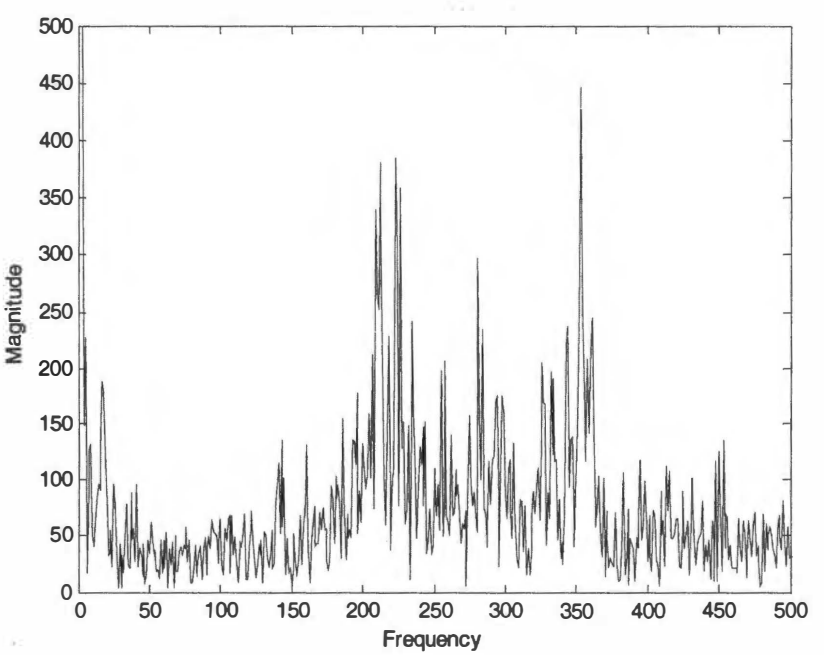


Figure 4.3.3 FFT analysis on raw interaction force in z axis.

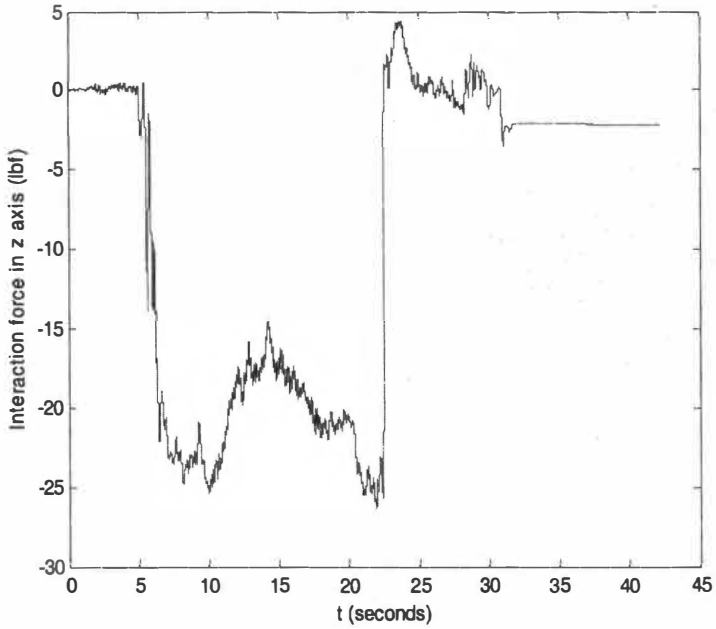


Figure 4.3.4 Filtered interaction force in z axis

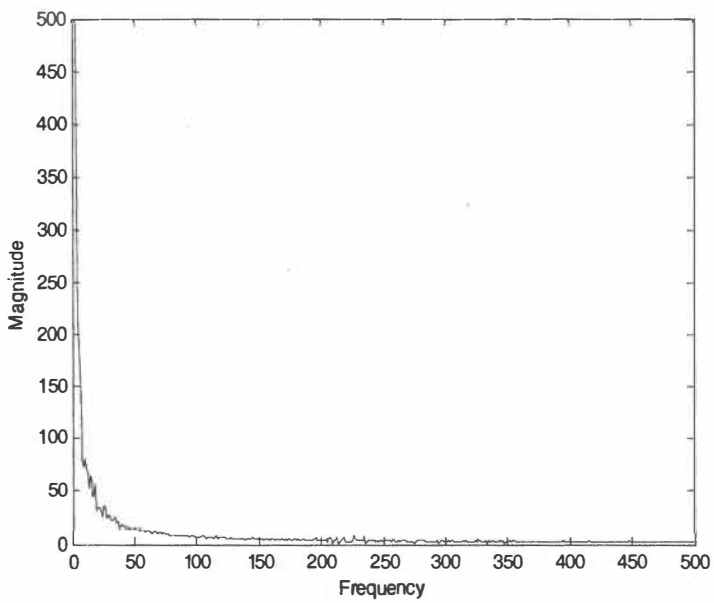


Figure 4.3.5 FFT analysis on filtered interaction force in z axis

4.3.3 Results of GM(1,1) Modeling Interactions

Since the grey model prediction is a local curve fitting extrapolation method, at least four data sets are needed for the GM(1,1) prediction model to provide a satisfied prediction value [83]. According to our experiment results six data sets are good enough for GM(1,1) model to give good prediction of interaction forces and torques. Figure 4.3.6 is the GM(1,1) prediction for interaction force in z axis and Figure 4.3.7 is its zoom in figure.

The complete results of the interaction forces and torques analysis and predictions are shown in Appendix B.

4.4 Conclusions

From above sections we can conclude that grey prediction model GM(1,1) is a universal predictor for dynamic system with poor information. Only six past and present data sets are required to predict the interactions between tooling and environment during a remote operation. It is simple, fast and can be used in real time application.

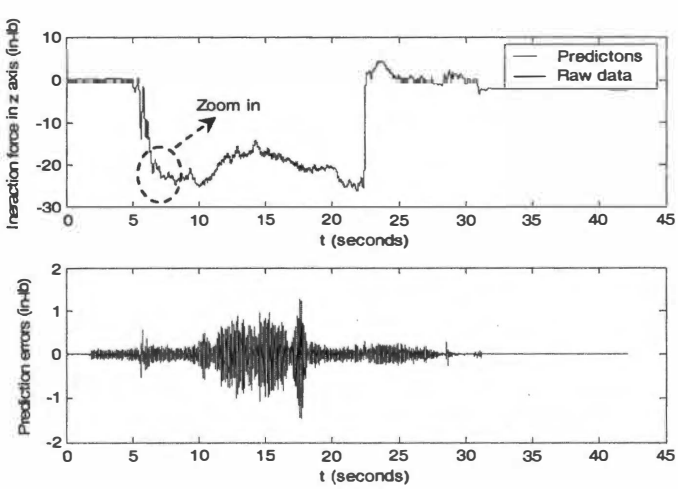


Figure 4.3.6 GM(1,1) prediction results for force in z axis

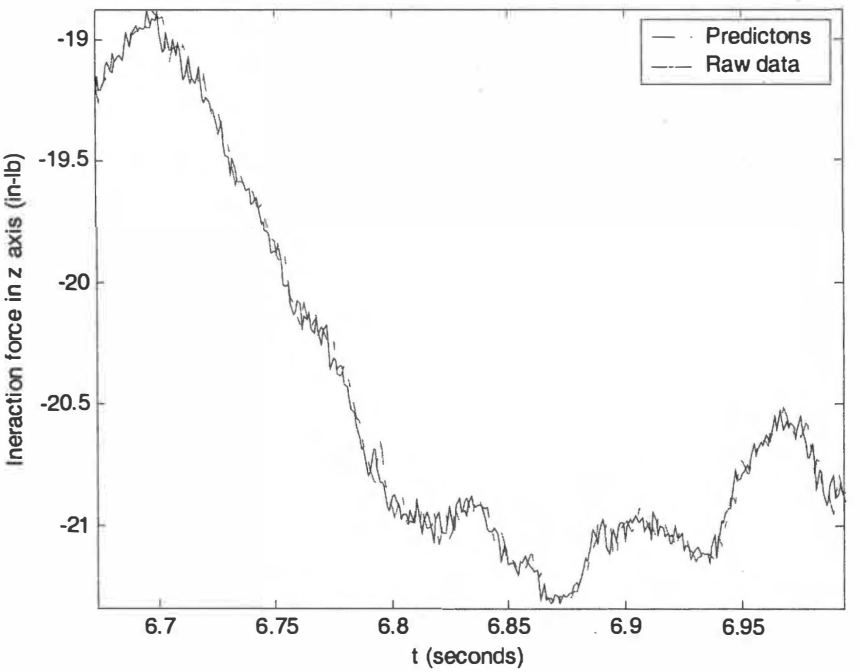


Figure 4.3.7 Prediction results for force in z axis (zoom in)

Chapter 5

Methods for Controlling Tool Interactions

5.1 Introduction

Real manipulation applications often require interaction between the manipulator and the environment. Typical examples of manipulation tasks are mechanical parts assembling, object contour surface following, and mechanical object machining using mechanical tools. During interaction, the end effector of the manipulator cannot move freely in all directions due to the environmental setting constraints on the geometric path. This situation is generally considered as constrained motion. The capability of handling interaction between manipulator and environment is one of the fundamental requirements for the success of a manipulation application. The contact force at the end effector of the manipulator is a significant representative of the state of interaction. Large contact force should be avoided since it may cause damage to both manipulator and the contacted object.

Controlling interaction by means of purely position control strategy requires a highly accurate trajectory planning and a high performance control system that guarantees the end effector following the planned trajectory accurately. So it is crucial to have an accurate manipulator model (kinematics and dynamics) and a detailed knowledge of the mechanical and geometrical features of the environment. However, it is difficult to get high precision model of both manipulator and environment since there are structured and non-structured uncertainties existing in the manipulator system and the typical

remote task environments are highly unstructured. The existence of modeling errors cause actual trajectory deviates from the planned one. This provokes a reaction of the position control system. If the deviation is due to the contact of end effector with environment, then interaction forces arise and the pure position control system takes action to reduce the deviation and follow the planned trajectory. This may generate large contact forces if the end effector contacts with a stiff environment. It then causes system saturation, instability, or mechanical damages.

Since contact force is a significant representative of features of interaction between end effector and environment, it is necessary to include contact force information in control scheme for proper handling of interaction applications of manipulator. Several force control strategies have been developed. Among them, the parallel approach to force/position control of robotic manipulator developed in [58] is one of the most successful control strategies for interaction applications. It has two controllers working in parallel: one is position controller which guarantees the end effector followed the planned trajectory, while the other one is force controller which assures the contact forces stay within a reasonable value. This parallel approach makes the force control action prevail over the position control action by designing the force controller as a PI-type controller and position controller as a PD-type controller so that the deviation from the given force trajectory is limited to a reasonable tolerance.

This chapter begins with a brief introduction of dynamic force/position parallel control scheme. A simplified grey prediction force/position parallel fuzzy controller is then developed, and finally, simulation of a two link manipulator is studied for testing the performance of the developed control strategy.

5.2 Parallel Control Scheme

The dynamic force/position parallel control strategy [58] is based on the basic architecture of parallel controller shown in Figure 2.3.4. First, consider the problem of tracking a joint space trajectory. The dynamic model of an n-joint manipulator can be written as

$$\vec{b}(\vec{q})\ddot{\vec{q}} + \vec{n}(\vec{q},\dot{\vec{q}}) = \vec{u} \tag{5.2.1}$$

$$\vec{n}(\vec{q},\dot{\vec{q}}) = \vec{c}(\vec{q},\dot{\vec{q}}) + \vec{g}(\vec{q}) \tag{5.2.2}$$

where vector $\vec{q} = (q_1, \cdots, q_n)^T$ denotes the joint positions of the manipulator. $\vec{b}(\vec{q})$ is the $(n \times n)$ symmetric and positive definite joint space inertia matrix. $\vec{c}(\vec{q},\dot{\vec{q}})$ is the $(n \times 1)$ vector of Coriolis and centrifugal forces, $\vec{g}(\vec{q})$ is the $(n \times 1)$ vector of gravitational forces, and $\vec{u}$ is the $(n \times 1)$ vector of generalized control torques at the joints.

For interaction control, the motion trajectory and interaction forces are often described in operational space so it is convenient to perform control analysis by using an operational space dynamic model of manipulator. Let the elements of $(m \times 1)$ vector $\vec{x}$ be the generalized coordinates in the operational space, the manipulator dynamics can be rewritten as

$$\vec{B}(\vec{x})\ddot{\vec{x}} + \vec{N}(\vec{x},\dot{\vec{x}}) = \vec{U} \tag{5.2.3}$$

$$\vec{N}(\vec{x},\dot{\vec{x}}) = \vec{C}(\vec{x},\dot{\vec{x}}) + \vec{G}(\vec{x}) \tag{5.2.4}$$

in which $\vec{B}(\vec{x})$ is the $(m \times m)$ symmetric and positive definite inertia matrix if m=n and the manipulator is at a nonsingular configuration. $\vec{C}(\vec{x},\dot{\vec{x}})$ is the $(m \times 1)$ vector of Coriolis

and centrifugal forces, $\vec{G}(\vec{x})$ is the $(m\times 1)$ vector of gravitational forces, and $\vec{U}$ is the generalized force at the end effector of the manipulator.

The relationship between operational space and joint space terms for a non-redundant manipulator in a nonsingular configuration is created by the following equations:

$$\vec{B}(\vec{x}) = \vec{J}^{-T}(\vec{q})\vec{b}(\vec{q})\vec{J}^{-1}(\vec{q}) \tag{5.2.5}$$

$$\vec{C}(\vec{x},\dot{\vec{x}}) = \vec{J}^{-T}(\vec{q})\vec{c}(\vec{q},\dot{\vec{q}}) - \vec{B}(\vec{x})\dot{\vec{J}}(\vec{q})\dot{\vec{q}} \tag{5.2.6}$$

$$\vec{G}(\vec{x}) = \vec{J}^{-T}(\vec{q})\vec{g}(\vec{q}) \tag{5.2.7}$$

$$\vec{u} = \vec{J}^{T}(\vec{q})\vec{U} \tag{5.2.8}$$

where $\vec{J}(\vec{q})$ is the $(m\times m)$ manipulator Jacobian matrix which is nonsingular in this case.

The dynamic model Equation (5.2.3) is highly nonlinear and strongly coupled . It can be decoupled by a new input vector $\vec{f}$ [88]. Rewrite equation (5.2.3) into a control structure, namely

$$\vec{U} = \vec{B}(\vec{x})\vec{M}_d^{-1}\vec{f} + \vec{N}(\vec{x},\dot{\vec{x}}) \tag{5.2.9}$$

which leads to the system described by

$$\ddot{\vec{x}} = \vec{M}_d^{-1}\vec{f} \tag{5.2.10}$$

in which $\vec{M}_d$ is the diagonal positive definite desired inertia matrix which used to ensure dynamic decoupling.

When the manipulator interacts with the environment, a contact force term at the end effector should be added into the dynamic model Equation (5.2.3)

$$\vec{B}(\vec{x})\ddot{\vec{x}} + \vec{N}(\vec{x},\dot{\vec{x}}) = \vec{U} - \vec{f}_i \tag{5.2.11}$$

where $\vec{f}_i$ is the $(m\times 1)$ vector of contact forces that the manipulator end effector imposed on the environment. Substituting equation (5.2.10) into equation (5.2.11) leads to a linear decoupled dynamics for the system.

$$\vec{U} = \vec{B}(\vec{x})\vec{M}_d^{-1}\vec{f} + \vec{N}(\vec{x},\dot{\vec{x}}) + \vec{f}_i \qquad (5.2.12)$$

Equation (5.2.12) is the control law of the manipulator when it is interacting with the environment. From the basic architecture of parallel controller shown in Figure 2.3.4, we get

$$\vec{f} = \vec{f}_p + \vec{f}_f \qquad (5.2.13)$$

in which $\vec{f}_p$ and $\vec{f}_f$ are $(m\times 1)$ vectors of control input due to position and force control action respectively. The position control action can be designed as a resolved acceleration position controller [89]:

$$\vec{f}_p = \vec{M}_d\ddot{\vec{x}}_d + \vec{K}_v(\dot{\vec{x}}_d - \dot{\vec{x}}) + \vec{K}_p(\vec{x}_d - \vec{x}) \qquad (5.2.14)$$

where $\vec{x}_d$ is the $(m\times 1)$ vector of desired end effector position. $\vec{K}_v$, $\vec{K}_p$ are diagonal velocity and position control gain matrix respectively. Since the force control action should be prevail over the position control action, the force control action should be designed as a proportional-integral function:

$$\vec{f}_f = \vec{K}_f\vec{e}_f + \vec{K}_i\int_0^t \vec{e}_f\, d\tau \qquad (5.2.15)$$

$$\vec{e}_f = \vec{f}_d - \vec{f}_i \qquad (5.2.16)$$

where $\vec{f}_d$ is the desired value of the contact force and $\vec{K}_f$, $\vec{K}_i$ are diagonal force and integral control gain matrix, respectively.

The resulting parallel control scheme is shown in Figure 5.2.1.

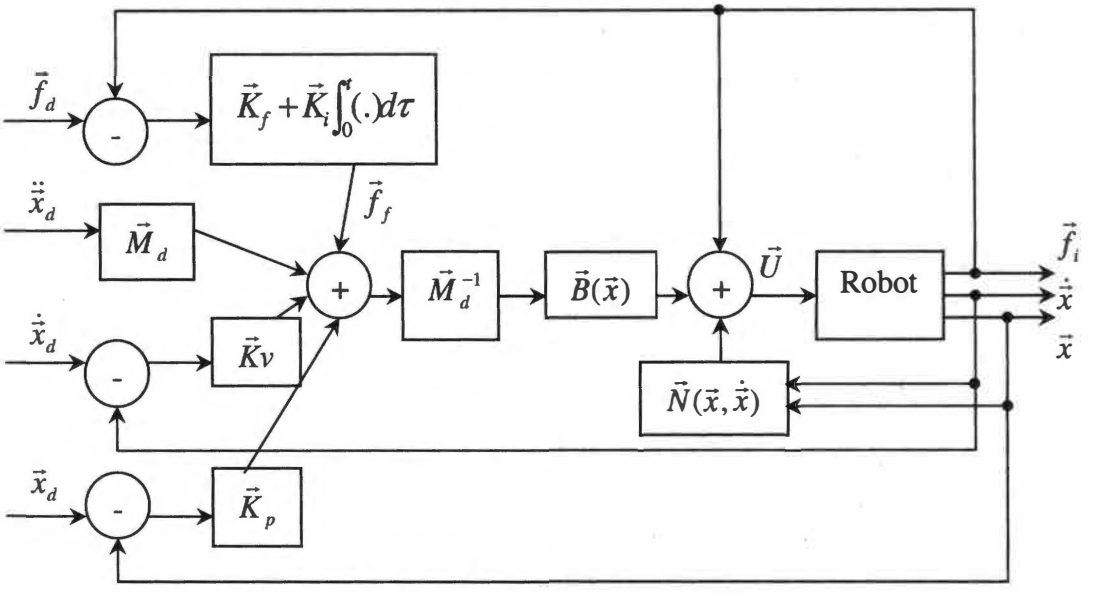


Figure 5.2.1 Block scheme of parallel force/position control.

5.3 Grey Prediction Fuzzy Parallel Control

Direct implementation of equation (5.2.12) is computationally inefficient and the control performance of the law fully depends on the accuracy of the known dynamic model of manipulator system. However, because a manipulator is a multivariable, highly non-linear and coupled complex system, it is hard to model the dynamic characteristics of the manipulator system precisely. A model free controller is created in this section to solve these kinds of problem by using the fuzzy set theory.

5.3.1 Fuzzy Logic Control

Fuzzy logic control is a non-analytical method based on decision-making approaches. The fuzzy set theory was originally developed by Lotfi A. Zadeh in 1960s [90] as a new mathematical method to deal with the uncertainty and imprecision and has

subsequently become the foundation of the novel fuzzy control strategies. The main feature of fuzzy logic controller is its ability to express the automatic control law in a linguistic way and describe the system using a set of fuzzy rules derived from by human expert knowledge.

As shown in Figure 5.3.1, the basic structure of fuzzy controller is very simple conceptually [91]. It consists of an input stage, a fuzzy inference engine, and an output stage. In the input stage, crisp sensor data and any other inputs are preprocessed by, scaling, and smoothing first, and then are fuzzified by appropriate membership functions. The fuzzy inference engine is a simulation of the logical reasoning of human beings. It contains a set of fuzzy if-then rules. These rules map the linguistic control knowledge of an expert to the automatic control strategies. In the output stage, the fuzzy outputs of the fuzzy inference engine are defuzzified to crisp values which can be taken by the actuators.

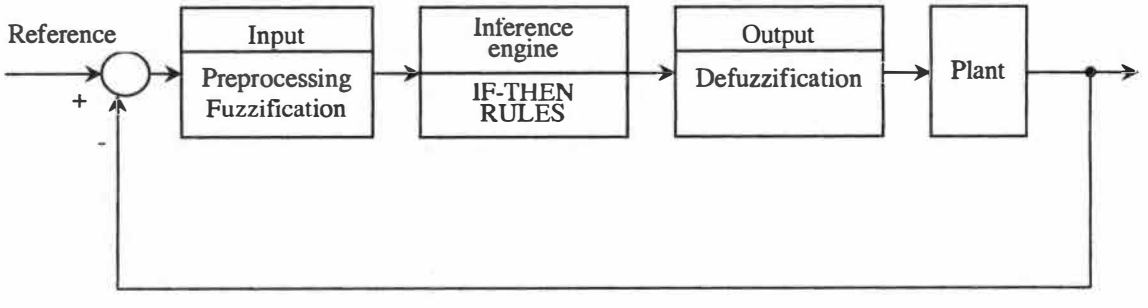


Figure 5.3.1 Conceptual structure of fuzzy control system

Membership functions

The most common used membership function is triangular, although trapezoids and sigmoid membership functions are also used. The shape of the membership is generally less important than the number of fuzzy variables which cover the required range of an input value, or the "universe of discourse". Typical fuzzy variables are defined as follows:

NL ≡ negative large

NM ≡ negative medium

NS ≡ negative small

ZE ≡ zero (5.3.1)

PS ≡ positive small

PM ≡ positive medium

PL ≡ positive large

Figure 5.3.2 shows an example of fuzzy variables cover a universe of discourse using triangular membership functions.

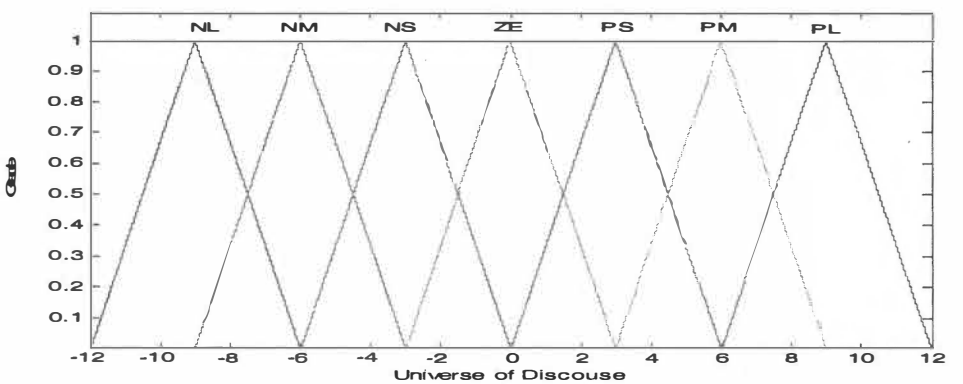


Figure 5.3.2 Fuzzy variables and their membership

In some cases, the membership functions can be modified by "hedges" that are equivalent to adjectives. Common hedges include "about", "near", "close to", "approximately", "very", "slightly", "too", "extremely", and "somewhat". These operations may have precise definitions, though the definitions can vary considerably between different implementations. "Very", for one example, squares membership functions; since the membership values are always less than 1, this narrows the membership function. "Extremely" cubes the values to give greater narrowing, while "somewhat" broadens the function by taking the square root.

Fuzzy Inference engine

As discussed earlier, the fuzzy inference engine is based on a set of logic rules in the form of if-then statements, where the IF part is named the "antecedent" and the THEN part is called the "consequent". Typical fuzzy control systems have dozens of rules. The general form the fuzzy if-then rule can be expressed as

R_i: if a is A and b is B then c is C (5.3.2)

in which R_i is the ith rule, a and b are the variables to be controlled and c is the control input. A, B and C are the corresponding fuzzy variables subsets which span the input and output universe of discourse.

For conveniently explaining the fuzzy if-then rule, let consider a set of rules for controlling a two-link manipulator:

R_1: if e is NM and $\dot{e}$ is NS then τ is PM else

R_2: if e is NM and $\dot{e}$ is ZO then τ is PM else

R_3: if e is NM and $\dot{e}$ is PS then τ is PM else

R_4: if e is NS and $\dot{e}$ is NS then τ is PS else

R_5: if e is NS and $\dot{e}$ is ZO then τ is PS else

R_6: if e is NS and $\dot{e}$ is PS then τ is PS else

R_7: if e is ZO and $\dot{e}$ is NS then τ is PS else

R_8: if e is ZO and $\dot{e}$ is ZO then τ is ZO else (5.3.3)

R_9: if e is ZO and $\dot{e}$ is PS then τ is NS else

R_{10}: if e is PS and $\dot{e}$ is NS then τ is NS else

R_{11}: if e is PS and $\dot{e}$ is ZO then τ is NS else

R_{12}: if e is PS and $\dot{e}$ is PS then τ is NS else

R_{13}: if e is PM and $\dot{e}$ is NS then τ is NM else

R_{14}: if e is PM and $\dot{e}$ is ZO then τ is NM else

R15: if e is PM and $\dot{e}$ is PS then τ is NM

where e is the position error, $\dot{e}$ is the velocity error, and τ is the control torque for the manipulator.

Let's look at rule number 15:

R15: if e is PM and $\dot{e}$ is NS then τ is NM

This rule uses the crisp true value of the position error and velocity error as inputs, in which position is positive medium and velocity error is negative small to some extent, to generate a control torque in the fuzzy set for the manipulator actuator, which is some value of negative medium. This result is used with the results of other rules to finally generate the crisp composite output. Obviously, the greater the truth value of "PM"and "NS", the higher the truth value of "NM", though this does not necessarily mean that the output itself will be set to "NM", since this is only one rule among many.

Generally, the fuzzy rule sets usually have more than one antecedents that are combined by using fuzzy operators, such as AND, OR, and NOT, though again the definitions tend to vary: AND, when mamdani min implication used, it simply uses the minimum weight of all the antecedents, while OR can be interpreted as max $(\vee)$. There is also a NOT operator that subtracts a membership function from 1 to give the "complementary" function.

There are several different ways to get the result of a fuzzy inference engine, but one of the most common and simplest is the "max-min" inference method [91]. Let's illustrate it by using the fuzzy rule set (5.3.3) Suppose at time $t = k$, the crisp control variables $e = e(k)$ and $\dot{e} = \dot{e}(k)$ which are position error and velocity errors of the manipulator respectively. These two crisp data are sent into the fuzzy controller and evaluated by the fuzzy if-then rule set. Rule 7, 9, and 11 are "activated" since their degree of fulfillment (DOF) is not zero. As shown in Figure 5.3.3, in R_{11} the membership of $e(k)$ belong to PS is 0.6 and the degree of membership to ZO of $\dot{e}(k)$ is 1.0. Therefore the DOF of R_{11} at this moment is

$$DOF_{11} = \mu_{PS}(e(k)) \wedge \mu_{ZO}(\dot{e}(k)) = 0.6 \wedge 1.0 = 0.6 \qquad (5.3.4)$$

If mamdani min $(\wedge)$ implication is used to define the connective AND on the antecedent side. The Consequent value of R_{11} NS will be cutting at the height of DOF_{11}. The shaded area in Figure 5.3.3 at R_{11} is the contributes $\mu_{NS}(\tau(k))$ of rule 11. Similarly, the degree of fulfillment of R_7 and R_8 are:

$$DOF_7 = \mu_{ZO}(e(k)) \wedge \mu_{NS}(\dot{e}(k)) = 0.4 \wedge 0.7 = 0.4 \qquad (5.3.5)$$

$$DOF_8 = \mu_{ZO}(e(k)) \wedge \mu_{ZO}(\dot{e}(k)) = 0.4 \wedge 1.0 = 0.4 \qquad (5.3.6)$$

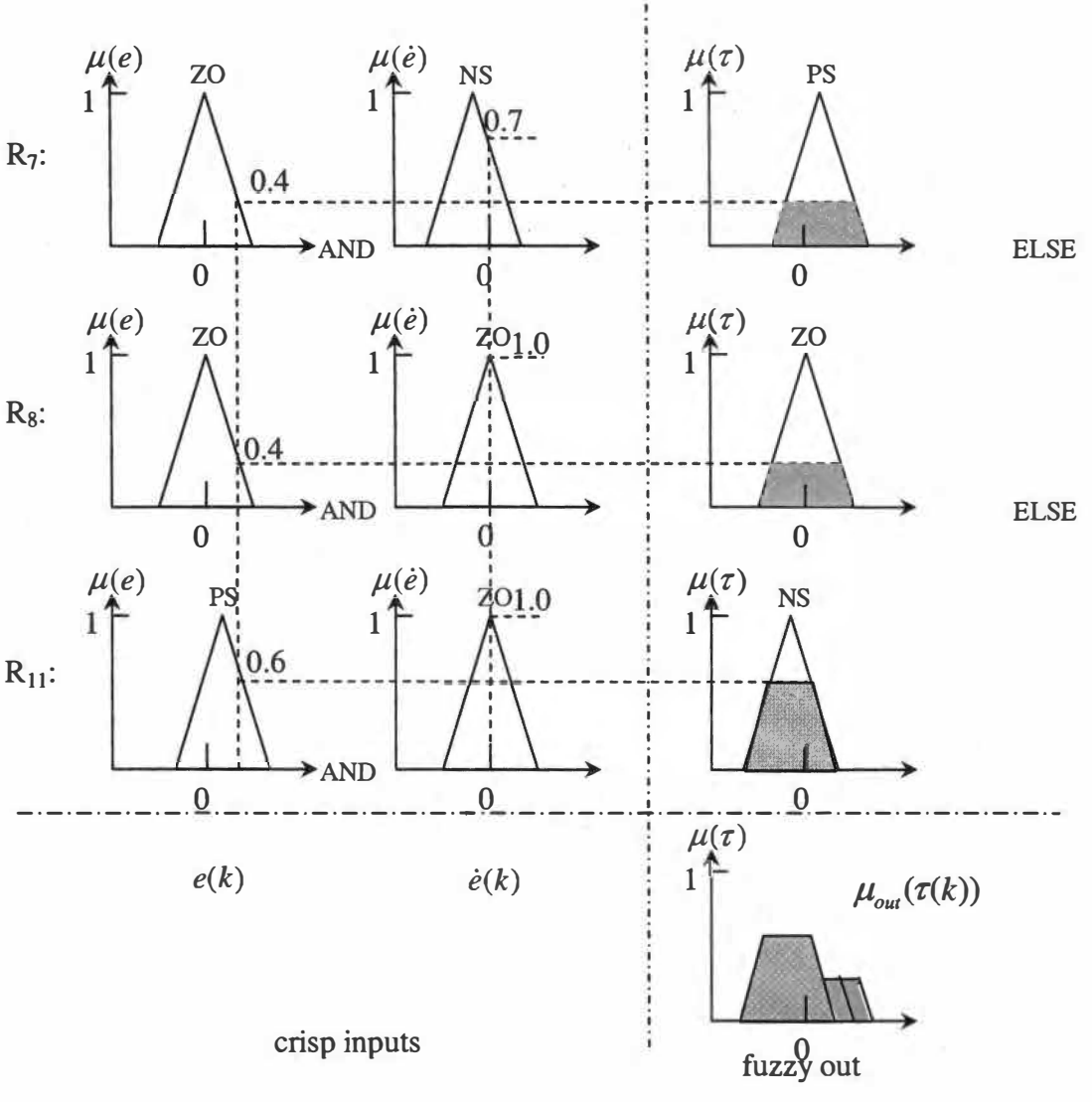


Figure 5.3.3 Max-min fuzzy inference

and their contributes $\mu_{PS}(\tau(k))$ and $\mu_{ZO}(\tau(k))$ are shaded areas corresponding to R_7 and R_8 in Figure 5.3.3 . The other rules in rule set (5.3.3) are not activated since their DOFs are zeros. Then overall fuzzy output of the fuzzy inference engine at this particular time is the union of the three outputs because the connective ELSE is interpreted as OR ($\vee$) in the max-min inference method.

$$\mu_{out}(\tau(k)) = \mu_{PS}(\tau(k)) \vee \mu_{ZO}(\tau(k)) \vee \mu_{NS}(\tau(k)) \tag{5.3.7}$$

It is shown as a shaded part at the bottom of Figure 5.3.3.

Defuzzification methods

The output of fuzzy inference engine is a fuzzy value. It should be defuzzified to obtain a physically meaningful value which can then be used in a physical system. This process is called defuzzification. Several defuzzification methods have been developed over the years such as centroid or center of area (COA), the center of sums (COS), and mean of maxima (MOM) as introduced in [91]. Each method has its own advantages and drawbacks. The selection of defuzzification technique may have a significant impact on the performance of a fuzzy controller.

The "centroid" method is popular, in which the "center of area" of the fuzzy result provides the crisp value. It favors the rule with the output of greatest area:

$$\tau^* = \frac{\sum_{i=1}^{N} \tau_i \mu_{out}(\tau_i)}{\sum_{i=1}^{N} \mu_{out}(\tau_i)} \tag{5.3.8}$$

where τ^* is the crisp output of the fuzzy controller, τ_i is the ith point of a discrete universe of discourse. The potential drawbacks of centroid method are [91]: (1) Slow inference cycle since its favor of "center" value of the universe of discourse and its

complexity. (2) Only take into account the overlapping area once because UNION is used to create the resultant membership function.

Center of sums defuzzification method has been created to overcome the problem of centroid. It adds all the output of each activated rule to form the resultant membership function

$$\tau^* = \frac{\sum_{i=1}^{N} \tau_i \bullet \mu_{R_k}(\tau_i)}{\sum_{i=1}^{N} \sum_{k=1}^{n} \mu_{R_k}(\tau_i)} \tag{5.3.9}$$

in which $\mu_{R_k}(\tau_i)$ is the membership function from the activating of kth rule.

Another approach is the mean of maxima, which takes the value of the biggest contributor which is the crisp value of highest degree of membership in $\mu_{out}(\tau)$. It obviously favors the rule with the greatest output value. When there are more than one element having the maximum value of membership, the mean value of the maxima is taken into account:

$$\tau^* = \frac{1}{M} \sum_{m=1}^{M} \tau_m \tag{5.3.10}$$

where τ_m is the mth point in the university of discourse has the maximum value of membership function $\mu_{out}(\tau)$. M is the total number of such points. The mean of maxima is easy to use and faster than the centroid. It allows the controller to reach the edges of the universe of discourse but it does not take into account the overall shape of the resultant membership function.

Fuzzy rule generation

The most difficult and tedious work in fuzzy control is the generation of an adequate rule-base. Most fuzzy rules are derived from human expert's knowledge and by

iterative trial-error process. It is time consuming and the results rules are specific to each application. A major part of fuzzy control research has been concentrated on fuzzy rule generation. Procyk and Mamdani [92] first introduced a self-organized fuzzy logic controller (SOFLC). The basic idea of this SOFLC is to observe the environment while issuing appropriate control actions and then use the results of these actions to modify the control signals to improve performance of the control system. Since then this technique has been improved and used in many applications such as robot control [87, 93, 94], spacecraft attitude control [95], and chemical process control [96]. Most of these SOFLCs generate and modify the fuzzy rules online but the system response will has larger oscillation during initial learning while starting from zero initial rules. This may not be suitable for some system control. Off-line rule generation techniques are still needed. In this section, the fuzzy rule-base for manipulator control has been generated by using a method introduced in [97] which based on the system performance index.

The objective of the control system is to drive a physical system into a desire state and the system dynamics should not be sensitive to the variations of the system parameters. The control performance is evaluated by a performance index I defined by the system errors ($\vec{e}$) which must be driven to zeros. To achieve the desired control target, the control input should be changed in the direction of the negative gradient of the performance index I [97].

$$I = \sum_{i=1}^{n} \sqrt{e^2(i) + \rho \dot{e}^2(i)} \qquad (5.3.11)$$

where $e(i)$ and $\dot{e}(i)$ are the system error and its derivative at ith time interval, respectively. Here n is the total number of time intervals, and ρ is a weighting coefficient, $0 < \rho < 1$.

To calculate the negative gradient of performance index I, the partial derivative on I with respect to $e(i)$ and $\dot{e}(i)$ is required, namely:

$$\frac{\partial I}{\partial e(i)} = \frac{e(i)}{\sqrt{e^2(i) + \rho\dot{e}^2(i)}} \tag{5.3.12}$$

$$\frac{\partial I}{\partial \dot{e}(i)} = \frac{\rho\dot{e}(i)}{\sqrt{e^2(i) + \rho\dot{e}^2(i)}} \tag{5.3.13}$$

Then the negative gradient of the performance index $-|\nabla I|$ is

$$-|\nabla I| = \begin{bmatrix} -\frac{|e(i)|}{\sqrt{e^2(i) + \rho\dot{e}^2(i)}} & -\frac{\rho|\dot{e}^2(i)|}{\sqrt{e^2(i) + \rho\dot{e}^2(i)}} \end{bmatrix} \tag{5.3.14}$$

So the control input at time interval i can be expressed as

$$u(i) = u(i-1) + \eta\delta u(i) \tag{5.3.15}$$

$$\delta u(i) = \eta(-|\nabla I|)\begin{bmatrix} e(i) \\ \dot{e}(i) \end{bmatrix} \tag{5.3.16}$$

where $\delta u(i)$ is the change of control input at time interval i, and η is the learning rate.

The function in Equation (5.3.16) is used to generate fuzzy rules if the crisp control errors are replaced by their fuzzy value. Suppose that the input/output variables of the fuzzy control system has the same cardinality defined by equation (5.3.1) and

$$F_u = \{NL, NM, NS, ZO, PS, PM, PL\} \tag{5.3.17}$$

is the fuzzy set for the system. To realize function in Equation (5.3.16) in fuzzy variables, let's use integers to represent the fuzzy value, that is

NL=-3, NM=-2, NS=-1, ZO=0, PS=1, PM=2, NL=3

and then the fuzzy set F_u can be express as:

$$F_u = \{-3, \quad -2, \quad -1, \quad 0, \quad 1, \quad 2, \quad 3\} \tag{5.3.18}$$

So the fuzzy if-then rule given in Equation (5.3.2) is modified to:

$$\text{if a is i and b is j then c is f(i, j)} \tag{5.3.19}$$

where i and j are the integers representing fuzzy values and f:{-n,...,n} × {-n,...,n} $\rightarrow \{-n,...,n\}$ is a nonlinear function derived from Equation (5.3.16), namely

$$f(i,j) = Sat(n, \eta(-|\nabla I|)\begin{bmatrix} i \\ j \end{bmatrix}) \tag{5.3.20}$$

Sat(n, x) is a saturation function:

$$Sat = \begin{cases} n & if \quad x > n \\ round(x+0.5) & if \quad 0 < x \le n \\ round(x-0.5) & if \quad -n \le x < 0 \\ -n & if \quad x < -n \end{cases} \tag{5.3.21}$$

The fuzzy control rule base is calculated by function given in Equation (5.3.20). The results are shown in Table 5.3.1

Table 5.3.1 Fuzzy rule base for manipulator system

		Error						
		NL	NM	NS	ZO	PS	PM	PL
Rate of Error	NL	PL	PM	PM	PM	PS	NS	NM
	NM	PL	PM	PS	PS	ZO	NS	NL
	NS	PL	PM	PS	PS	NS	NM	NL
	ZO	PL	PM	PS	ZO	NS	NM	NL
	PS	PL	PM	PS	NS	NS	NM	NL
	PM	PL	PS	ZO	NS	NS	NM	NL
	PL	PM	PS	NS	NM	NM	NM	NL

In summary, the fuzzy controller design is based on empirical methods, basically trial-and-error approach. The general process is as follows:

- Specify system inputs and outputs
- Fuzzify the system inputs
- Create the fuzzy rule set and fuzzy inference engine
- Determine the defuzzification method and defuzzy the fuzzy outputs
- Run the system to test the fuzzy controller, adjust details as required

5.3.2 Grey Prediction Force/Position Parallel Fuzzy Controller

Based on the discussions in Sections 5.2 and 5.3.1, a new force/position parallel fuzzy control strategy is now developed. First, the architecture of force/position parallel fuzzy controller which utilizes the feedback information of interaction force is given in Figure 5.3.4.

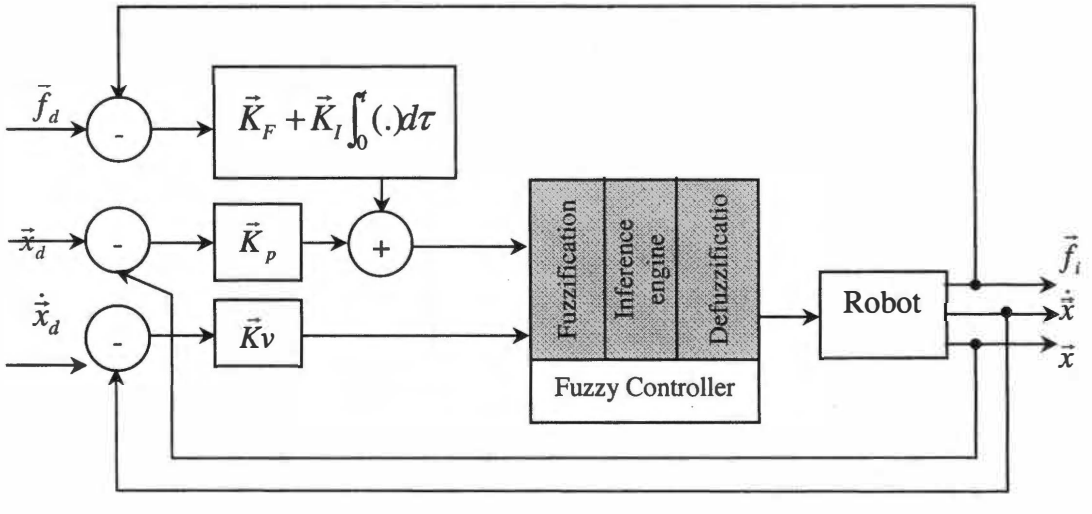


Figure 5.3.4 Architecture of force/position parallel fuzzy control.

It is simple compared with the original force/position parallel controller shown in Figure 5.2.1, and $\vec{K}_F$ and $\vec{K}_I$ are force control gain matrices which have the unit "mm/N".

The control strategy shown in Figure 5.3.4 uses the feedback information of interaction force to design the control system. This is usually referred to as afterward control [82]. The performance of this kind of control system may not be good enough for controlling manipulator with interaction with the environment. A novel control strategy which controls the next step interaction force to the desired force trajectory has been developed. This control scheme utilizes the predicted interaction force information in the control system design. The architecture is shown in Figure 5.3.5.

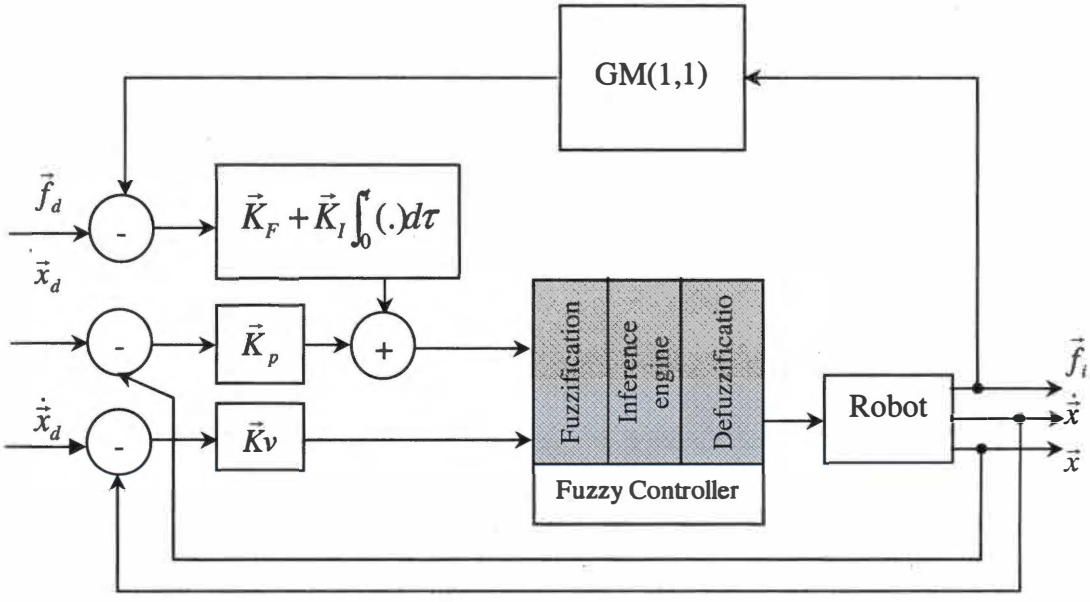


Figure 5.3.5 Architecture of grey prediction force/position parallel fuzzy control

where GM(1,1) is a grey prediction model developed in Chapter 4.

5.4 Simulation

In this section, a two-link planar RR robot manipulator model will is use to test the control performance of the above control strategies. The two-link manipulator is widely used for simulation in the literature since its dynamics is simple enough and still maintains all the nonlinear effects common to general robot manipulators.

5.4.1 Models of Two-Link Manipulator

The two-link manipulator is shown in Figure 5.4.1 where the masses of the two links are denoted by m_1 and m_2, respectively and they are assumed to be concentrated at the ends of the links. The moments of inertia J_1 and J_2 about the centers of gravity of each link are also included in the model. The dynamic model of the two-link manipulator, previously defined, is:

$$\vec{b}(\vec{q})\ddot{\vec{q}} + \vec{n}(\vec{q},\dot{\vec{q}}) = \vec{u} \tag{5.2.1}$$

$$\vec{n}(\vec{q},\dot{\vec{q}}) = \vec{c}(\vec{q},\dot{\vec{q}}) + \vec{g}(\vec{q}) \tag{5.2.2}$$

where the inertia matrix $\vec{b}(\vec{q})$ is expressed as:

$$\vec{b}(\vec{q}) = \begin{bmatrix} (m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos q_2 + J_1 + J_2 & m_2 l_2^2 + m_2 l_1 l_2 \cos q_2 + J_2 \\ m_2 l_2^2 + m_2 l_1 l_2 \cos q_2 + J_2 & m_2 l_2^2 + J_2 \end{bmatrix} \tag{5.4.1}$$

$$\vec{c}(\vec{q},\dot{\vec{q}}) = \begin{bmatrix} -m_2 l_1 l_2 (2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)\sin q_2 \\ m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 \end{bmatrix} \tag{5.4.2}$$

$$\vec{g}(\vec{q}) = \begin{bmatrix} (m_1+m_2) g l_1 \cos q_1 + m_2 g l_2 \cos(q_1+q_2) \\ m_2 g l_2 \cos(q_1+q_2) \end{bmatrix} \tag{5.4.3}$$

where, l_1 and l_2 are the length of the link 1 and link 2, respectively.

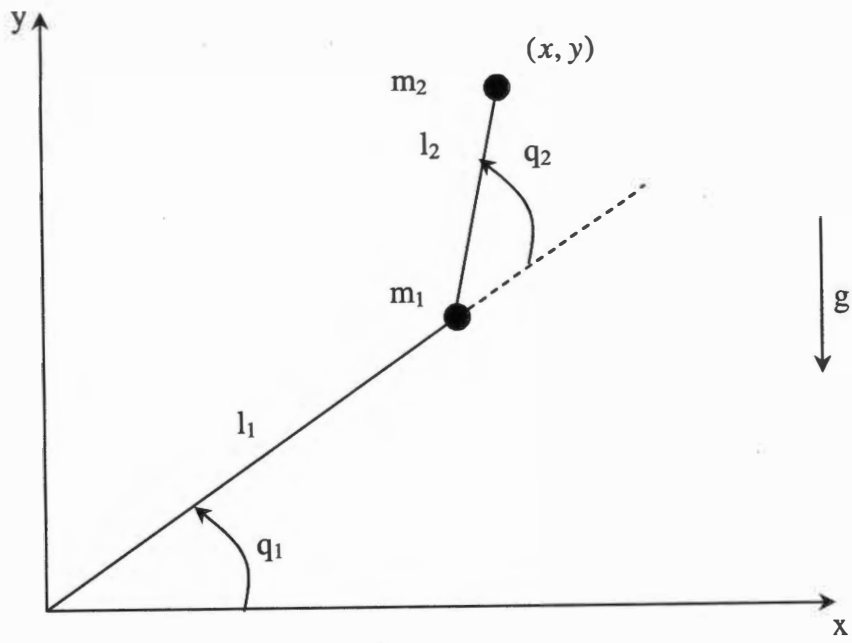


Figure 5.4.1 2-link RR planar manipulator.

The direct kinematics function for the 2-link manipulator is

$$T(\vec{q}) = \begin{bmatrix} 0 & \sin(q_1+q_2) & \cos(q_1+q_2) & l_1\cos q_1 + l_2\cos(q_1+q_2) \\ 0 & -\cos(q_1+q_2) & \sin(q_1+q_2) & l_1\sin q_1 + l_2\sin(q_1+q_2) \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4.4}$$

So the joint space variables q_1 and q_2 can be transformed to operational space variables x and y by:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1\cos q_1 + l_2\cos(q_1+q_2) \\ l_1\sin q_1 + l_2\sin(q_1+q_2) \end{bmatrix} \tag{5.4.5}$$

and the inverse kinematics functions for elbow-up configuration are given as follows:

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} \alpha+\beta \\ \cos^{-1}\left(\dfrac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2}\right) \end{bmatrix} \tag{5.4.6}$$

$$\alpha = \tan^{-1}\frac{y}{x}$$

$$\beta = \cos^{-1}\left(\frac{x^2 + y^2 + l_1^2 - l_2^2}{2l_1\sqrt{x^2 + y^2}}\right)$$

Finally, the Jacobian of the two-link manipulator is

$$J = \begin{bmatrix} -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \qquad (5.4.7)$$

5.4.2 Model of the Environment

It is difficult to devise a precise model of a teleoperation environment since it is highly unstructured. To simplify the simulation process, the movement of the two-link planar manipulator is constrained on axis x and the interaction forces of cutting are modeled as:

$$f_x = k(x - x_c) \qquad (5.4.8)$$

$$f_y = -\zeta f_x \qquad (5.4.9)$$

where k is the elastic coefficient. The constant ζ is strictly positive and x is the end effector position on the x axis while x_c is the position of contact point on x axis. The contact force exerted by the end effector while manipulator contacting with the environment is denoted by f_x while the cutting force denoted as f_y. The directions of the forces are show in Figure 5.4.2.

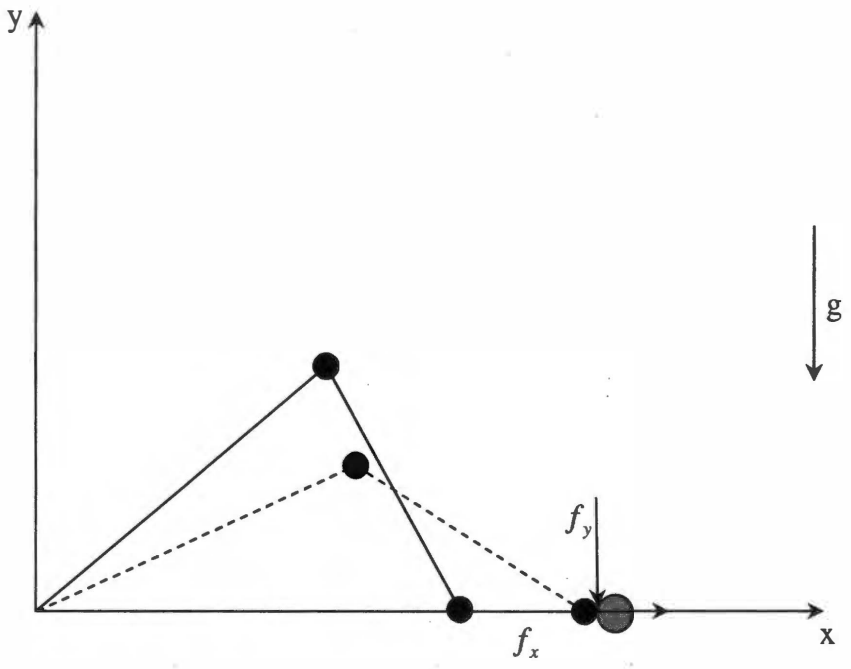


Figure 5.4.2 Interaction forces

5.4.3 Simulation Results

The moving trajectory of the end effector is constrained on x axis:

$$x = 50t + 0.5l_1 \qquad \text{mm} \qquad (5.4.10)$$

$$y = 0 \qquad \text{mm} \qquad (5.4.11)$$

where t is the time in seconds and $l_1 = 1000mm$ is the length of the first link of the manipulator. The trajectory of contacting point x_c is:

$$x_c = \begin{cases} 600 & mm \quad t \le 2s \\ 50t + 500 - 5(t-2) & mm \quad 2s < t \le 2.1s \\ 50t + 500 - 0.5 & mm \quad t > 2.1s \end{cases} \qquad (5.4.12)$$

A fuzzy controller has been built for the two-link manipulator to test the control performance of force/position parallel control strategy and grey prediction force/position

parallel control strategy while the manipulator cutting the environment. The performance of pure position control has also been investigated. A large disturbance force, from 4 to 4.05 seconds, is induced during the simulation. To begin, create the membership functions for the joint variable of the manipulator. These functions are shown in Figures 5.4.3 - 5.4.5.

Using equation (5.3.20), the control rules for this two-link manipulator have been generated as shown in Table 5.4.1.

The control performance is shown in Figures 5.4.6 - 5.4.14.

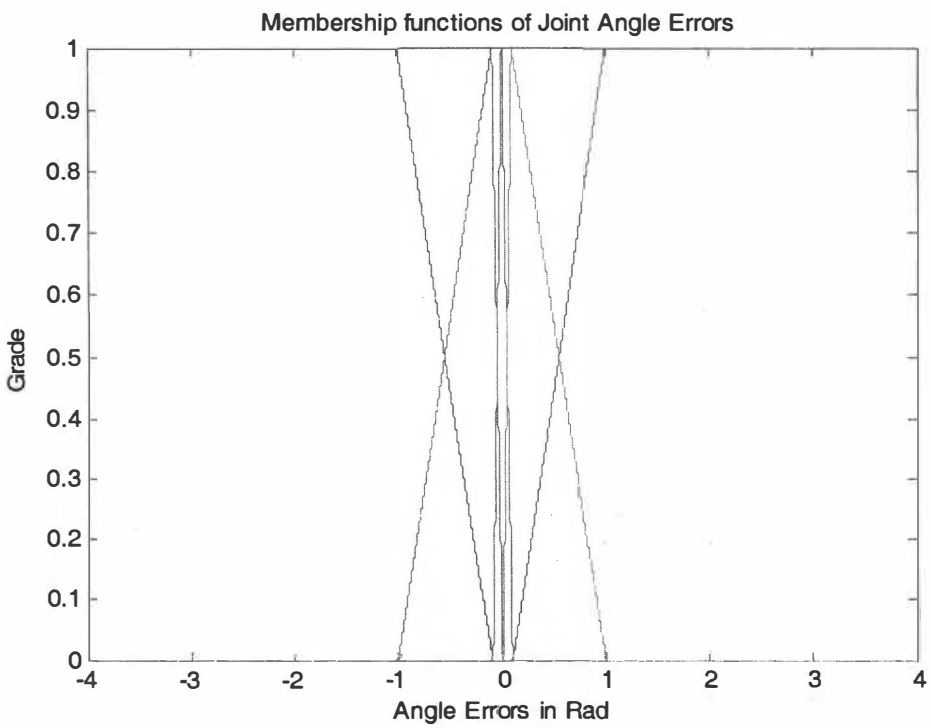


Figure 5.4.3 Membership functions of joint angle errors

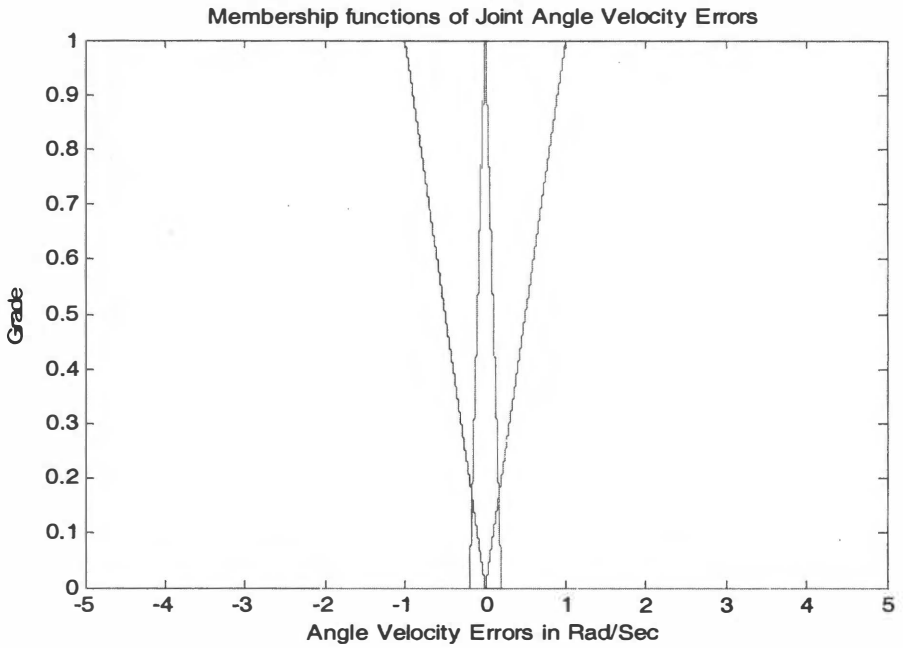


Figure 5.4.4 Membership functions of joint angle velocity errors

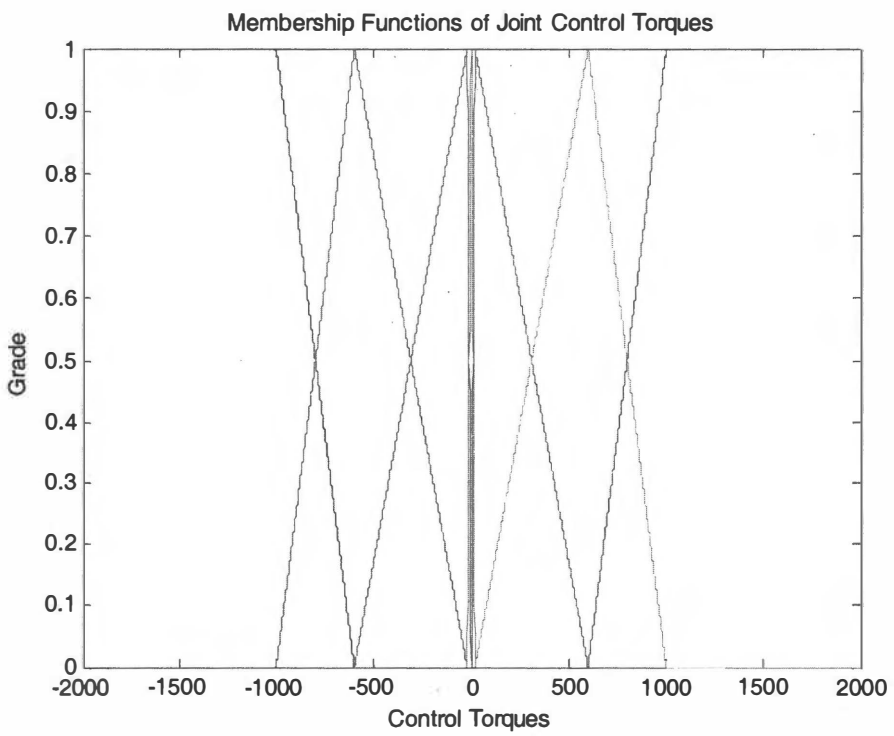


Figure 5.4.5 Membership functions of joint control torques

Table 5.4.1 Fuzzy control rules for two-link manipulator

		Error				
		NM	NS	ZO	PS	PM
Rate	NS	PM	PS	PS	NS	NM
Of	ZO	PM	PS	ZO	NS	NM
Error	PS	PM	PS	NS	NS	NM

Position control

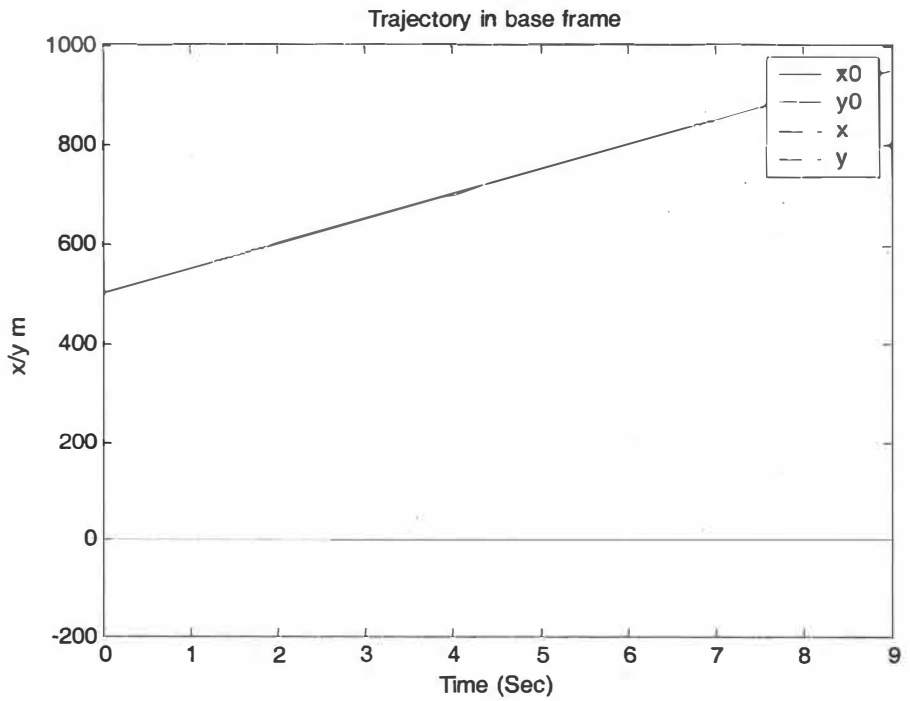


Figure 5.4.6 Trajectory of end-effector in operational space

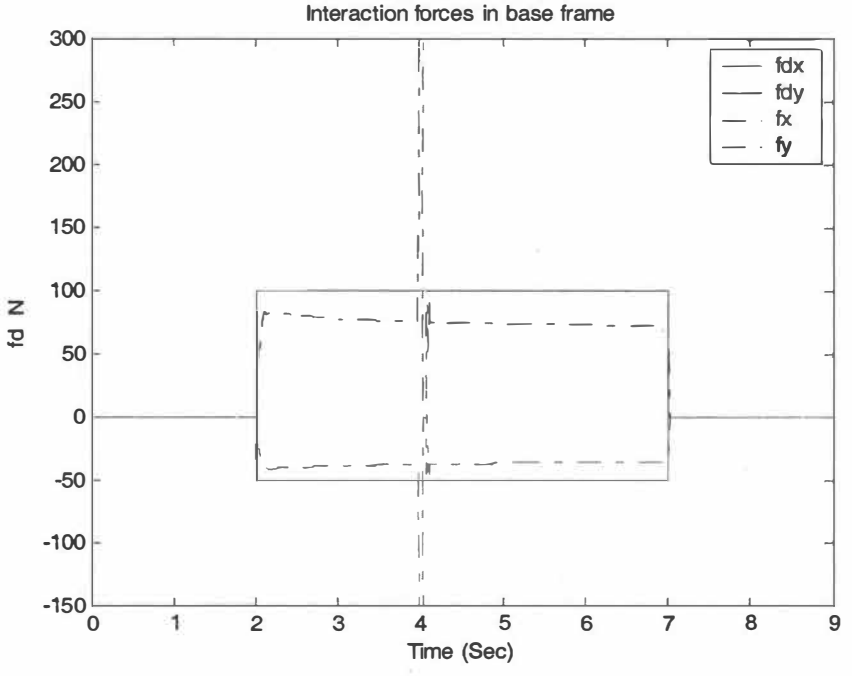


Figure 5.4.7 Interaction forces in operational space

Feedback force/position parallel control

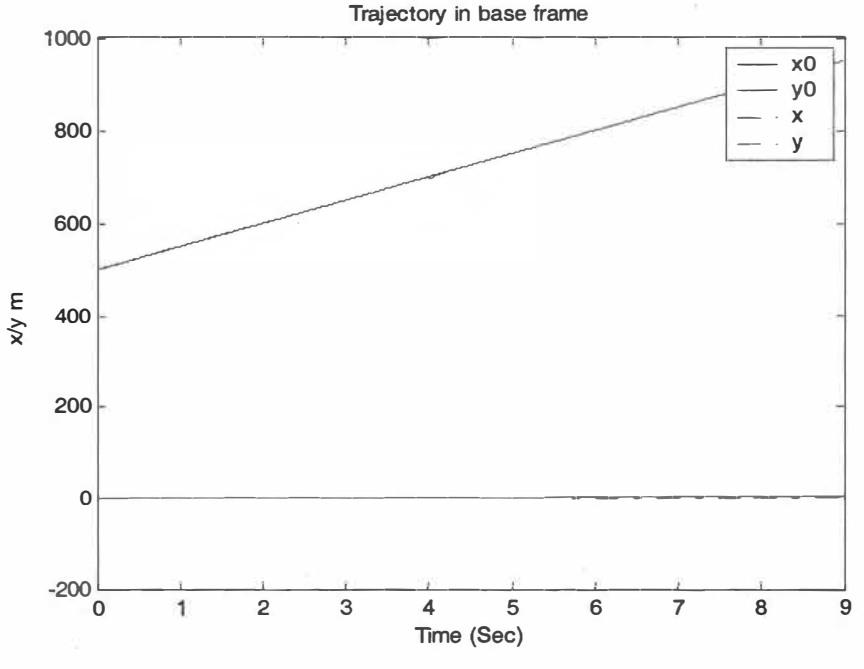


Figure 5.4.8 Trajectory of end-effector in operational space

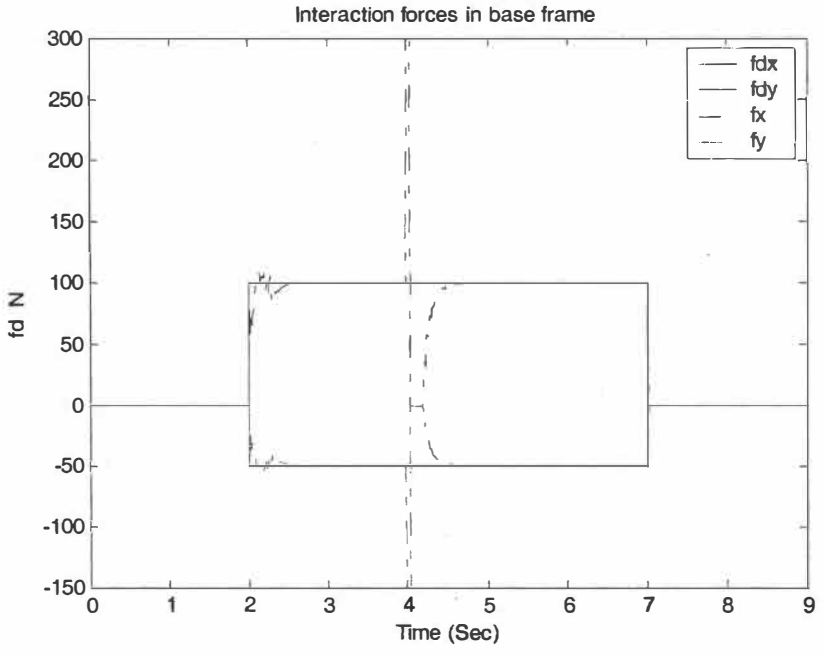


Figure 5.4.9 Interaction forces in operational space

Grey prediction force/position parallel control

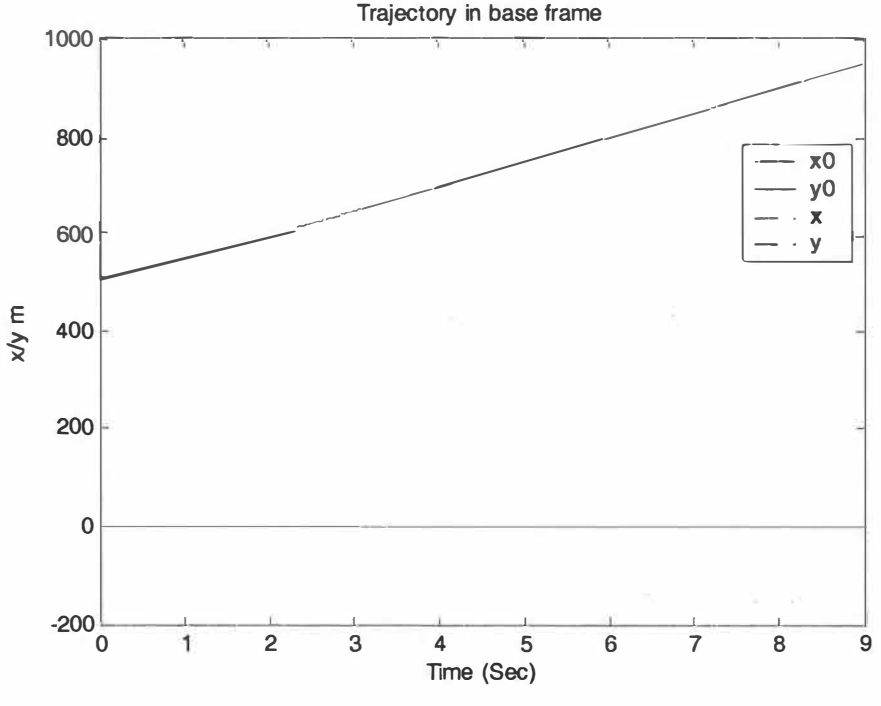


Figure 5.4.10 Trajectory of end-effector in operational space

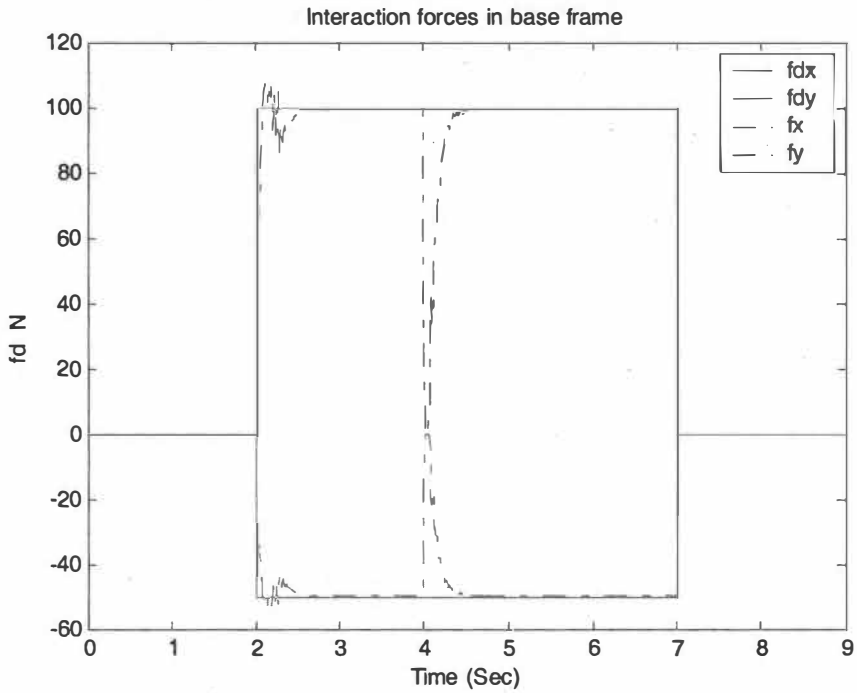


Figure 5.4.11 Interaction forces in operational space

Trajectory following performance form t=4.00 to 4.05 s while disturbance occurs

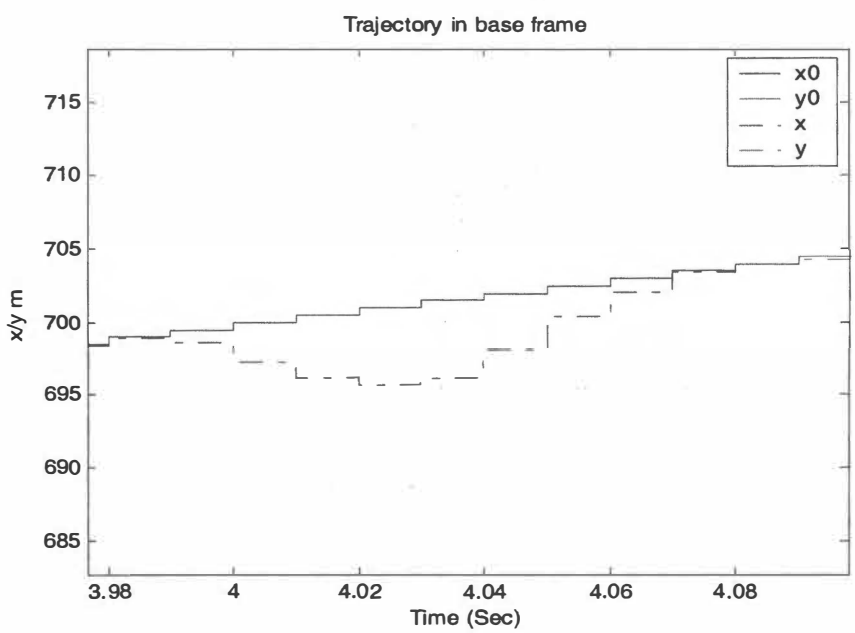


Figure 5.4.12 Pure position control

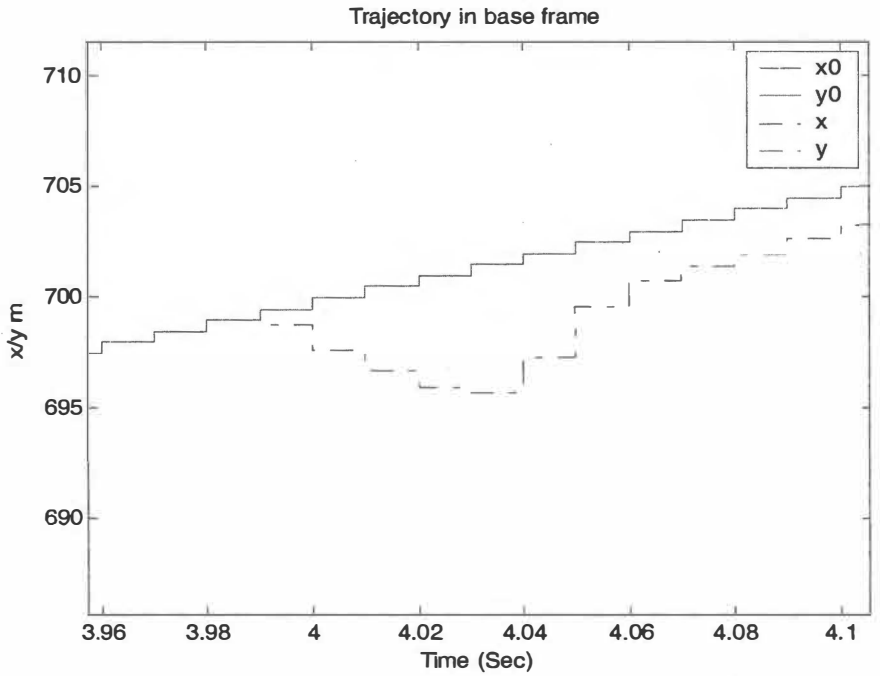


Figure 5.4.13 Feedback force/position control

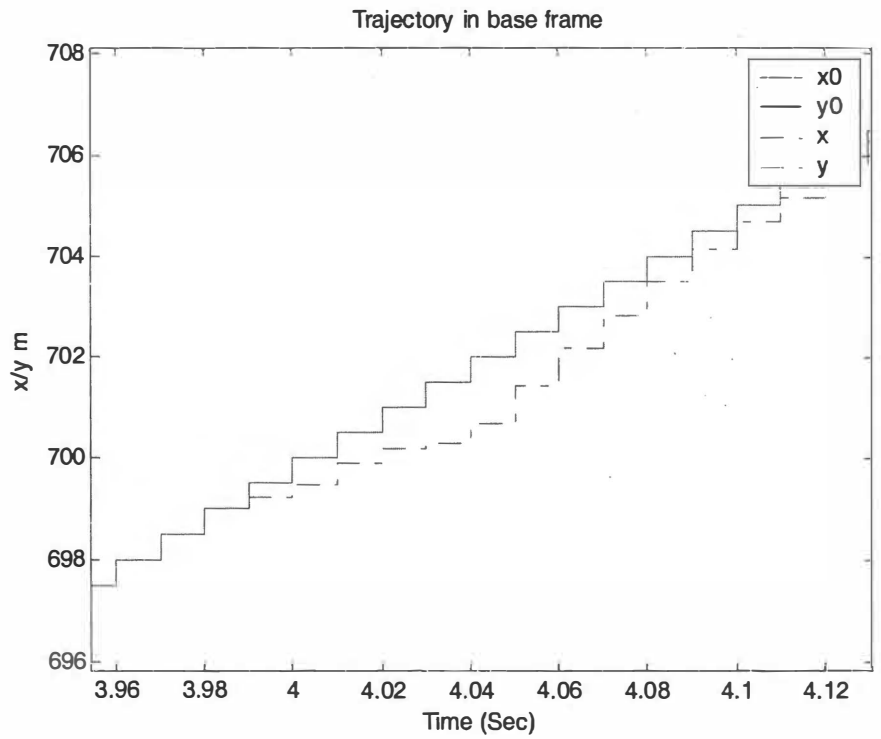


Figure 5.4.14 Grey prediction force/position parallel control

In the above figures, x0,y0 are the desired position trajectories and x, y are the actual position trajectories in operational space. Also fdx, fdy are desired forces and fx, fy are actual feedback forces.

5.4.4 Conclusions

Comparing the results shown in Figures 5.4.6, 5.4.8, and 5.4.10, the position trajectory following performances of pure position control, feedback force/position control, and grey prediction force/position control results are good except when there is a significant force disturbance from time t=4.00 to 4.05s. The actual trajectory deviates from the desired one by about 5 mm but the actual trajectory of the grey prediction force/position parallel control is closer to the desired one than the other two as shown in Figures 5.4.12-5.4.14. The force control performance of the grey prediction control is the best among the three control schemes investigated, when a large force disturbance is imposed. The controller slows down the movement of the end effector before the disturbance force happens in order to avoid a large interaction force as shown in Figure 5.4.11. Pure position control and feedback force/position parallel control do not have this capability. The significant interaction forces can not be reduced before they occur (Figures 5.4.7 and 5.4.9).

Chapter 6

Experimental System Description

6.1 Introduction

An experimental human-machine cooperative telerobotics system (HMCTR) has been built in Robotics and ElectroMechanical Systems Laboratory (REMSL) base on the overall telerobotic system architecture introduced in Chapter 3. In this chapter, the system hardware and software will be described in detail. This chapter is a duplication of author's writing published in [99].

6.2 Hardware

The advanced telerobotic system in REMSL consists of two main components; the Robot Task Space Analyzer (RTSA) and the Human Machine Cooperative Telerobot (HMCTR). In the RTSA, the human operator selects a region of interest (ROI) in the task space of the robot using stereo sensor head and captures the image of ROI. The captured stereo image can be sent to the automated stereo image processing to build a 3-D model of the ROI on background and the same time the human operator can pick up another ROI and build the 3-D model of the environment using the interactive manual modeling Graphical User Interface (GUI) of the RTSA. After all the models of feasible ROIs being generated, the operator builds the task plan file, which describes the execution of task, including the manipulator and tooling motions, using the task planner. The task planner, as an integral part of RTSA, is also an interactive GUI system, and the operator is only required to select the part and tooling position on the 3-D model of the ROI. Detailed

data, such as the required position and rotation of the end-effector or the manipulating procedure, are automatically generated by the task planner. These autonomous functions of RTSA increase the efficiency of the task execution and improve the system reliability by allowing the operator to check for safe operation by examining the 3-D models and the task plan file generated. The schematic in Figure 6.2.1 illustrates the hardware connections of this advanced telerobotic system. The set of hardware in use may be divided into a robot system part, including the Titan II manipulator and its peripherals and controller, and a robot task scene Analyzing (RTSA) part, including the RTSA model builder and task planner computer and sensor head connections.

6.2.1 Robot System Components

Schilling TII manipulator

The Schilling Titan II is a six-degree-of-freedom hydraulic manipulator constructed primarily of titanium and weighing 225 pounds, with a reach of approximately 76 inches and a payload at full extension of 240 pounds. It incorporates a two-finger gripper with a maximum opening of 5 inches. The manipulator in the Robotics and Electromechanical Systems Laboratory is on loan from Oak Ridge national Laboratory (ORNL) and was a part of the Dual Arm Work Platform (DAWP). It is securely mounted on the lab floor and will be used to test the RTSA telerobotic system on a mockup in this lab.

Hydraulic Power Unit

The Hydraulic Power Unit (HPU) provides pressurized hydraulic fluid to the Schilling manipulator at 5 gpm at 3000 psig. It consists of a hydraulic pump, electric motor, fluid reservoir, motor controller, filtration system, and heat exchanger unit. It also

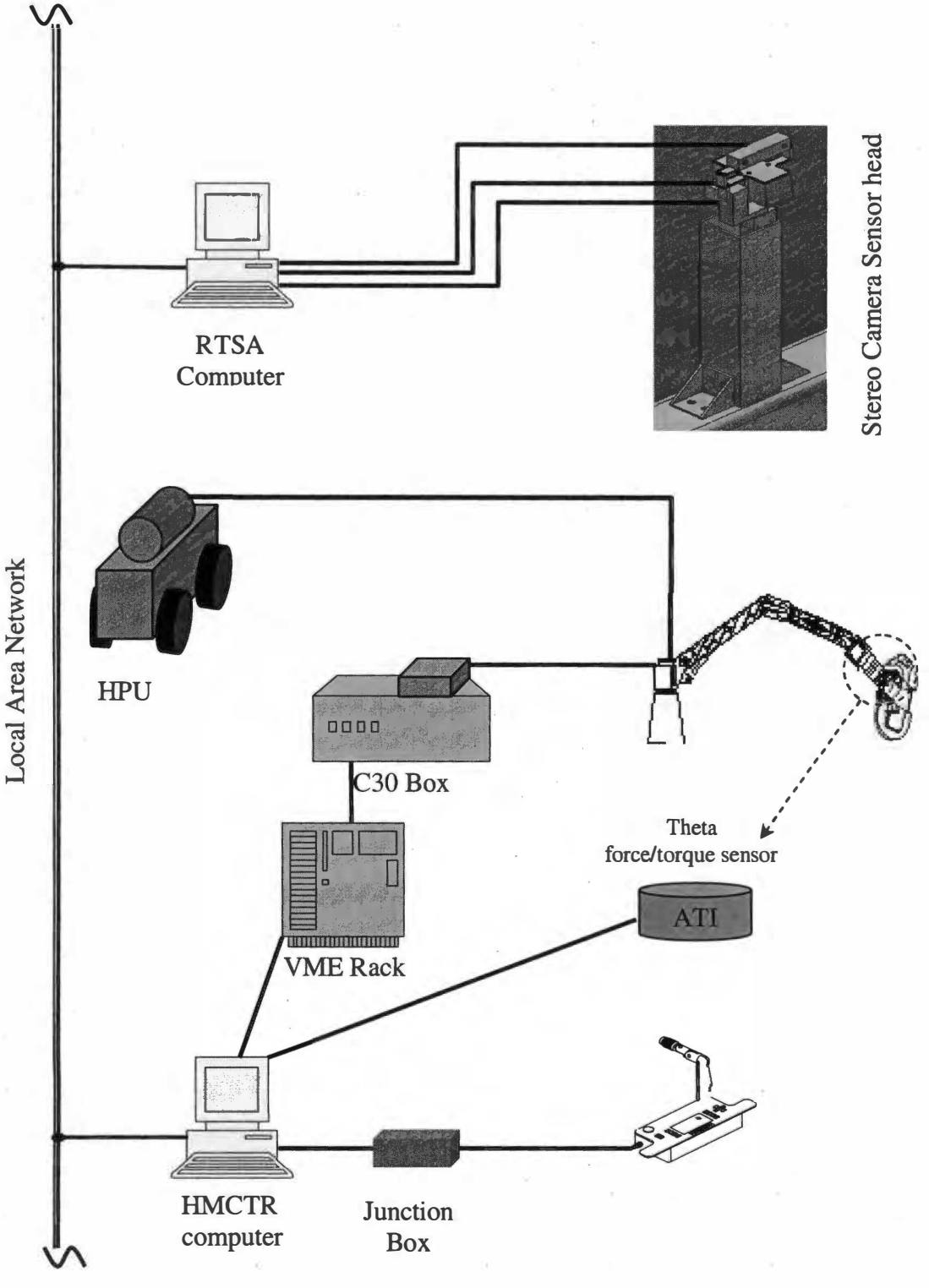


Figure 6.2.1 HMCTR hardware schematic.

has an emergency on-off control pendant and some pressure and temperature sensing in its 15 gallon reservoir.

Mini-Master Controller

The mini-master controller is the input device for manual teleoperation. Its primary component is the mini-master arm itself, a six-degree-of-freedom articulated arm with an approximately 11 inch reach. The box on which it is mounted has a power switch and 12 function keys, an LCD screen, and an RS-232 port with which to communicate with the host computer. The master arm has a freeze button on its terminal end, and two textured bands which may be squeezed to open and close the gripper.

Junction Box

The junction box provides power to the master controller and interface connections with either the unilateral slave controller or the host computer.

C30 Box

The C30 box is a component of the HMCTR host computer control system. It is connected directly to the slave manipulator. The C30 controller is also connected to a card which provides the interface to joint, torque, pressure, and other data from the manipulator. This card is placed in a VME card cage for access by the host computer.

VME Rack

The VME rack is used in the HMCTR host computer system configuration primarily as a relay for data from the C30 controller to the host computer. Typically, the VME rack would house a single board computer on which the host computer controller would run (usually with the Linux operating system with real-time application interface). The HMCTR philosophy has been to base the system on lower cost PC hardware;

however, the Schilling host computer C30 interface card is only compatible with VME technology. Thus, a Bit3 bus to bus adapter pair of card is being used to map memory from the VME address space to the PCI bus in the control PC.

Milwaukee Band Saw

The Milwaukee Portable Band Saw (Figure 6.2.2), Model 6230, will be the primary tool used in RTSA dismantlement demonstrations. It has a ½" wide, 10 tooth per inch blade which can be operated at variable speeds from 0-350 SFPM. The saw has a maximum capacity of 4 3/4" x 4 3/4" for square stock and 4 3/4" diameter round stock.

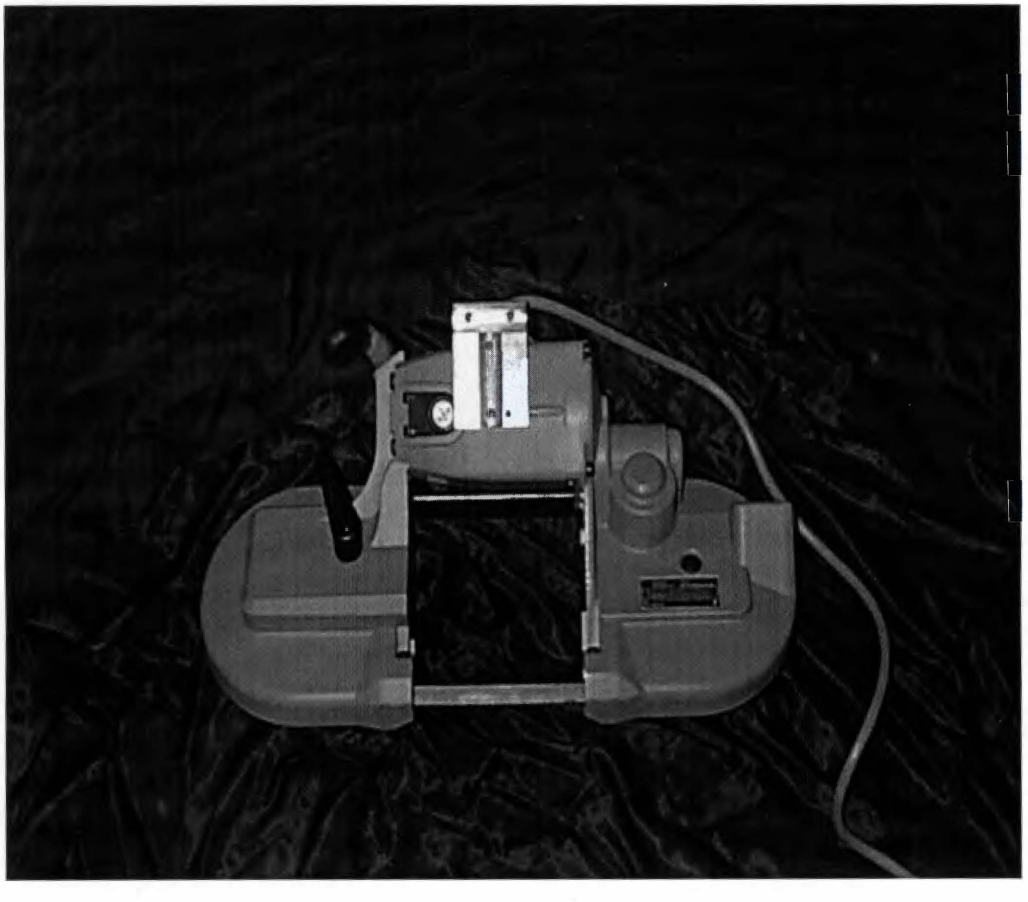


Figure 6.2.2 Milwaukee portable band Saw.

HMCTR Host Computer

The host computer is the real-time computer on which the Schilling controller runs. It is a Dell dual-processor-capable 1.5 GHz Pentium III PC with 256 M RAM. It is capable of being booted with LINUX with or without real-time application interface (RTAI) which pre-empties the Linux kernel to make the system have hard real-time capability. It is interfaced with the mini-master controller through a serial port, with the C30 controller by way of a Bit3 bus to bus adapter, and the ethernet for receiving task plans from RTSA computer.

6.2.2 Robot Task Scene Analyzer System

Pan-Tilt Sensor Head

The pan-tilt sensor head is comprised of the three main hardware components, viz. the BumbleBee™ stereo camera, the AccuRange 4000 LIR one-dimensional laser range finder, and the PTU 46-70 N pan-tilt unit. The sensor head is illustrated in Figure 6.2.3.

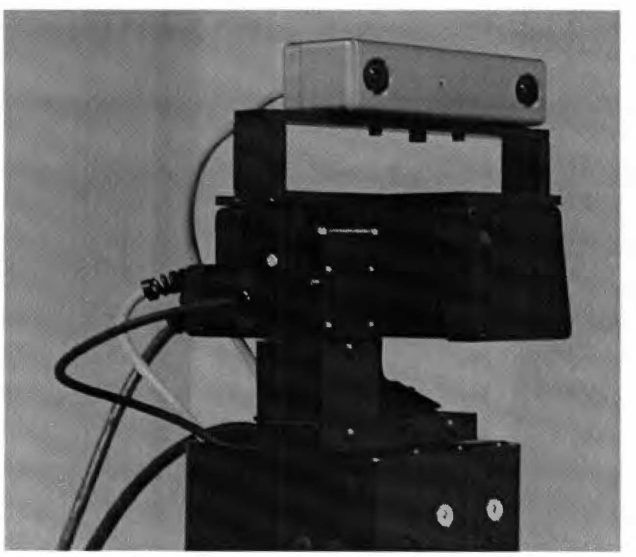


Figure 6.2.3 Pan-tilt sensor head.

BumbleBee™ Stereo Camera System

Bumblebee™ is Point Grey's new two-lens stereo vision camera (Figure 6.2.4). It has high quality 4mm focal length prefocused micro lenses, approximately 70° HFOV. It provides 640x480 resolution and a balance between 3D data quality, processing speed, and size. The camera is ideal for applications such as people tracking, gesture recognition, mobile robotics and other computer vision applications. Bumblebee camera is pre-calibrated for lens distortion and camera misalignments. It does not require in-field calibration and is guaranteed to stay calibrated. The left and right images are aligned within 0.05 pixel RMS error. The calibration information is preloaded on the camera, allowing the software to retrieve the image correction information. This allows seamless swapping of the cameras, or retrieving the correct information when multiple cameras are on the bus. Bumblebee is supplied as a full development kit, including the camera head, interface card, 4.5m cable, device driver, image acquisition software, and Triclops library.

Figure 6.2.4 BumbleBee™ stereo camera.

Laser Range Pointer (AccuRange 4000 LIR)

The AccuRange 4000LIR (Figure 6.2.5) is an optical distance measurement sensor with an accuracy of 0.1 inch and a useful range of zero to 50 feet for most diffuse reflective objects. It operates by emitting a collimated laser beam that is reflected from the target surface and collected by the sensor. It is a Class IIIb laser product, available in power levels of 8 mW (Standard) or up to 20 mW High power Laser optionally. AccuRange 4000 LIR uses near infrared light (780 nm wavelength). It is suitable for a wide variety of distance measurement applications that demand high accuracy and fast response times. It is rigidly connected to the pan-tilt frame and points in the same direction as the camera gaze vectors.

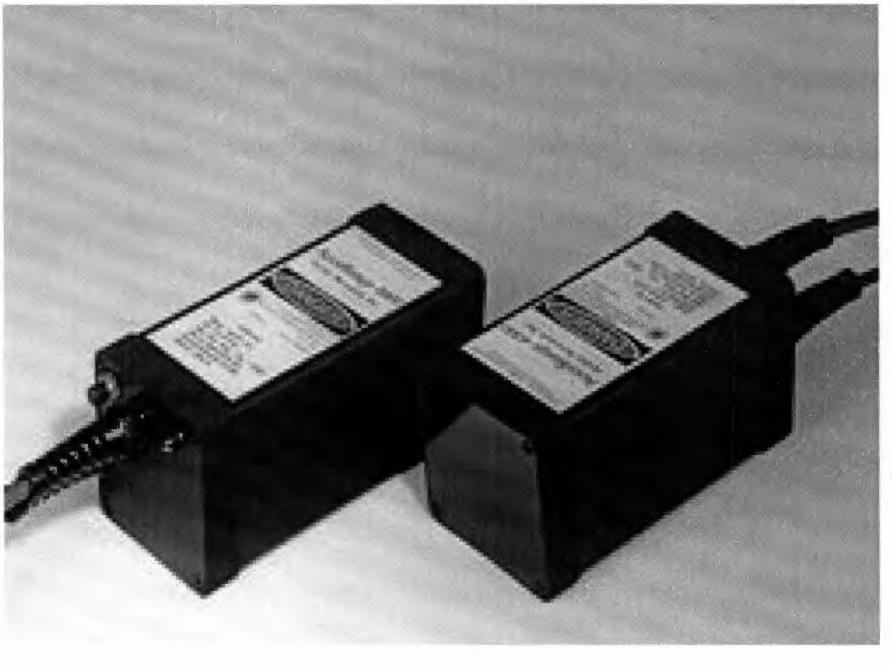


Figure 6.2.5 Laser range pointer.

PTU 46-70 N Pan-Tilt Unit

The Pan/Tilt unit (Figure 6.2.6) is one of the most important components of the sensor head. It carries the cameras and the laser range finder through the required range of motion in precise relation to the manipulator. It has two axes about which it can rotate to point the cameras and the laser range finder to a desired location in the robot task dimensional space.

The Directed Perception, Inc, model PTU 46-70 N pan-tilt unit was selected because it can provide low-cost, fast and accurate positioning of cameras or other payloads. It has a pan range of $\pm 160^{\circ}$ and a modified tilt range of up 31° and down 78° with the resolution of 0.012°. It has a precise control of position, speed and acceleration and on-the-fly speed and position change ability. The changing speed can reach up to $60^{\circ}/\sec ond$. It can be connected to host computer through a RS-232 port.

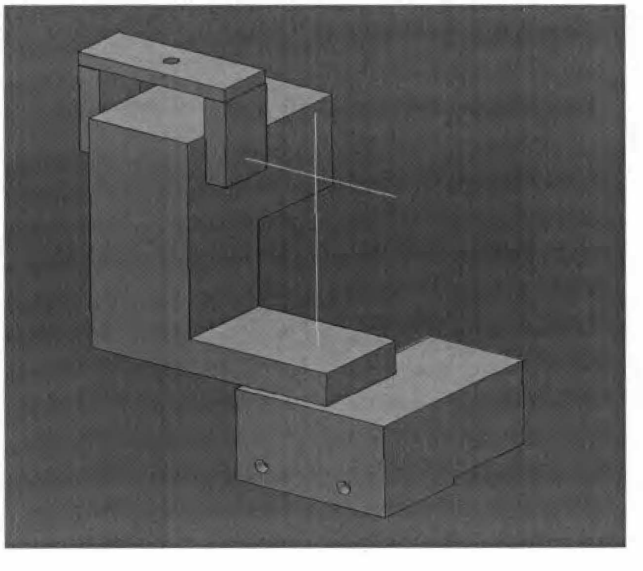


Figure 6.2.6 Autodesk inventor™ model of pan/tilt unit.

RTSA Computer

The main code for the Robot Task Scene Analyzer resides on the RTSA computer. It is a Dell dual-processor 1.5 GHz PC with 256 MB RAM running Windows 2000. An operator using RTSA would run this code here to model a scene and plan the robot task. It is interfaced with the camera controlling, pan-tilt head controlling, and the Ethernet.

6.3 Software

The HMCTR system software was developed in REMSL. It includes: the Robot Task Scene Analyzer, a graphic user interface which can be used to help human operator build a 3-D task space computer model and generate task plan, and the *Constellation*™ control system which reads the task plan file generated by RTSA and generates control commands based on the task plan file, execution modes such as manual teleoperation, subtask autonomous operation, and operation with interaction force control mode etc.

6.3.1 RTSA Software Components

The Robot Task Scene Analyzer (RTSA) software is a windows-based interface that has been written in Visual C^{++} and organized in a tree structure. The tree structure is illustrated in Figure 6.3.1. The following section describes the data flow that takes place within the Robot Task Scene Analyzer software. These descriptions refer to the diagram in Figure 6.3.2. Small squares in the diagram refer to hardware connections such as serial ports. Horizontal parallel lines with intervening labels represent data storage. Small cylinders represent socket connections, which may take place across an Ethernet network. Labeled blocks represent processes that receive, manipulate, and output data. Each of the subsections below describes the process or data storage unit and how it relates with the others.

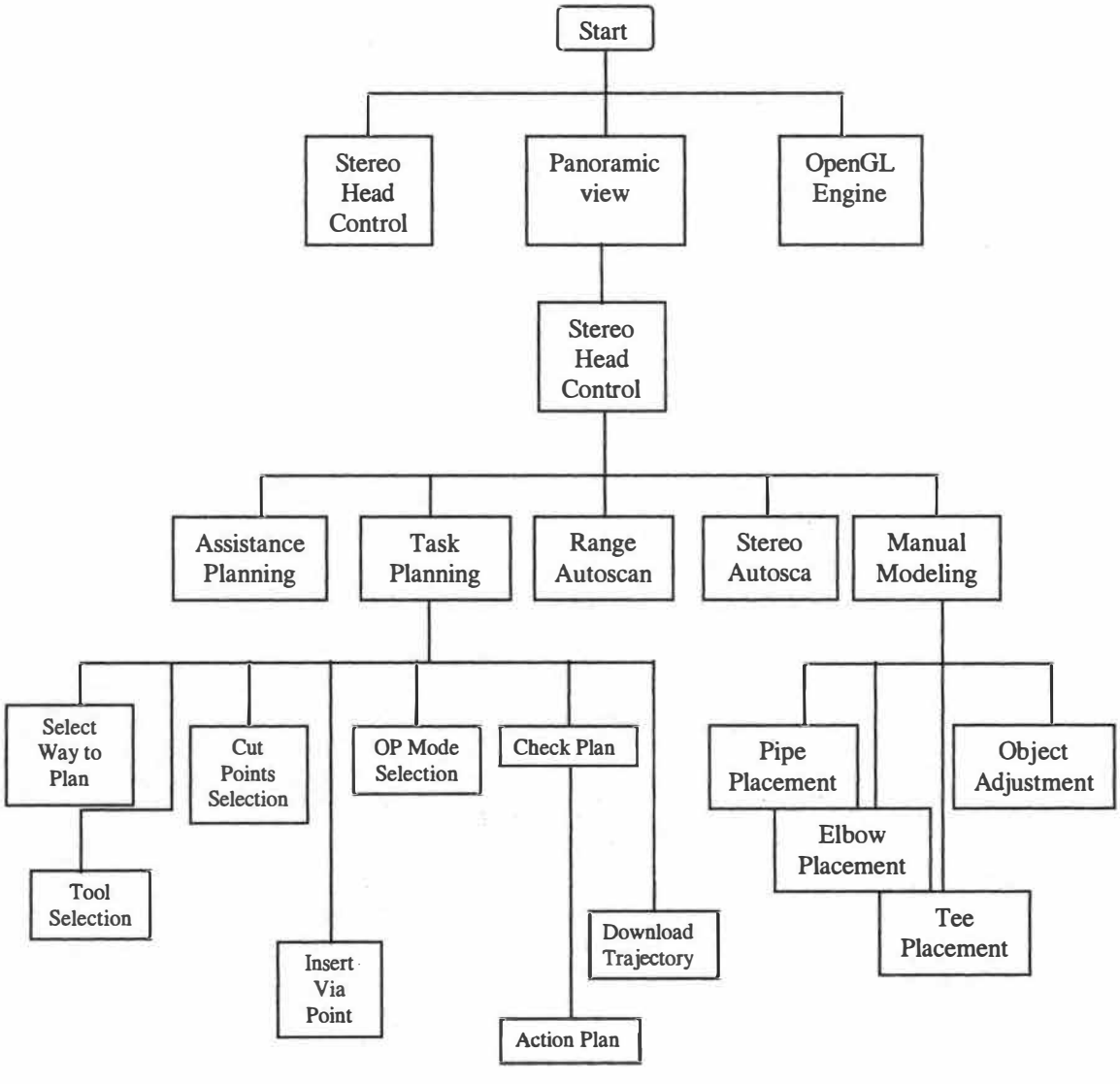


Figure 6.3.1 Robot task scene analyzer windows Tree.

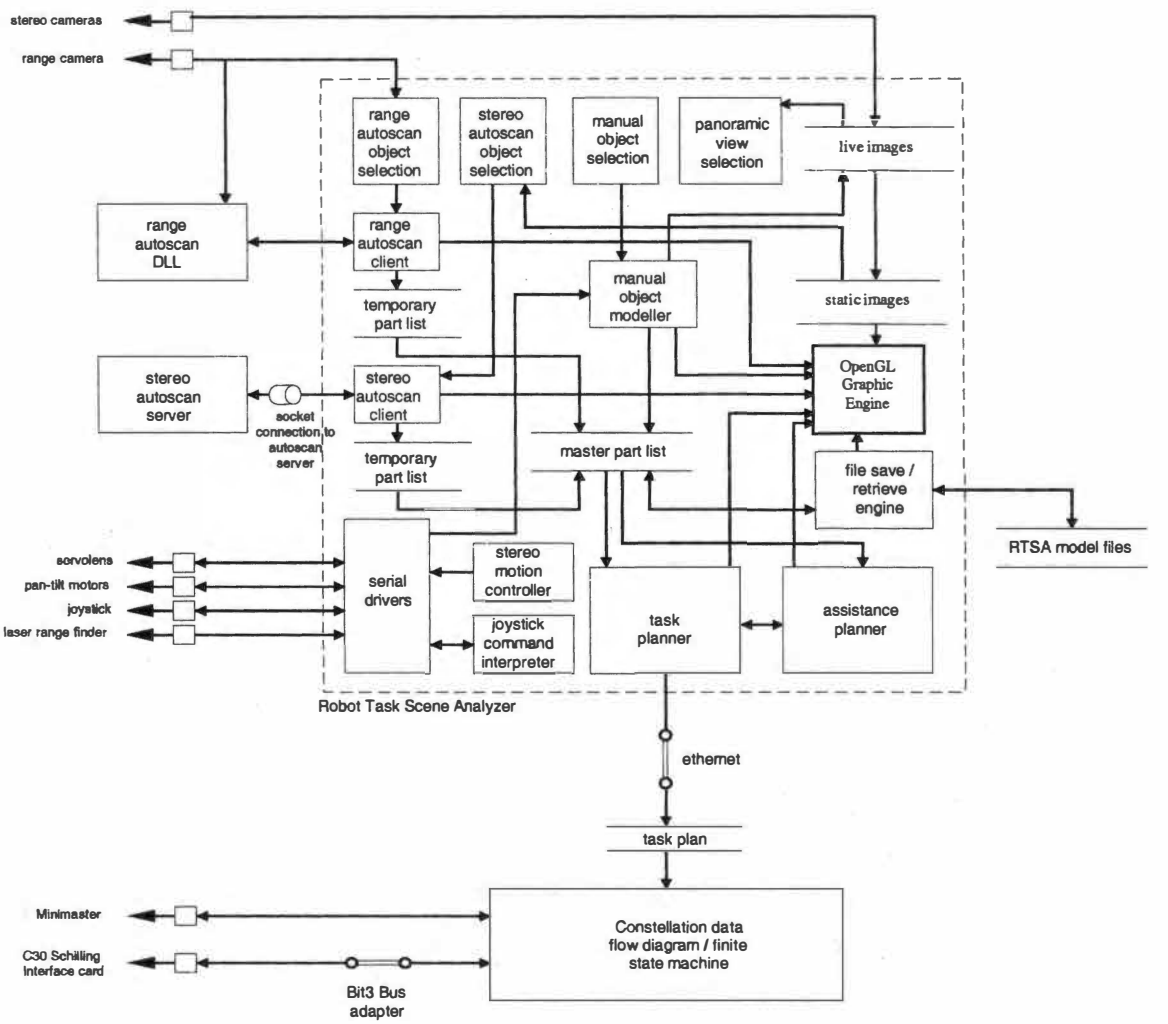


Figure 6.3.2 Robot task scene analyzer software data flow diagram.

Static images

Static images from the stereo camera are captured and stored in a separate buffer. These static images are necessary for texture mapping to the background block in the OpenGL virtual environment and for display to the operator and transmission to the stereo autoscan procedure. This is also the scene that is displayed in the main window of RTSA.

OpenGL engine

The OpenGL engine is used to construct and display the virtual models.

Panoramic view selection

This object corresponds to the window that opens at startup of RTSA. It displays live images from the left and right stereo cameras.

File save/retrieve engine

This set of functions allows a model being built in RTSA to be saved and retrieved. The model is saved by writing the master part list to a file. When the model is retrieved, the current model is erased and the saved information is used to regenerate the original master part list and the previous OpenGL model.

RTSA model files

These files are the model files saved by the file save/retrieve engine functions. They consist of a text file readable by RTSA containing information from the master part list. They appear in the Microsoft file list as RTSA-type files. Double clicking on one of these file icons will open RTSA with that saved model.

Manual object selection

This process corresponds to the manual modeling window. It receives information from the operator about types and attributes of the part to be created. It then relays this information to the manual object modeling process.

Manual object modeler

This process corresponds to several windows that allow the operator to define locations for the part to be created. There is a pipe placement, tee placement, and elbow placement window, each of which has a set of icons for choosing points on the physical mockup. This process receives camera pointing information for calculation of laser spot coordinates and object type information from the manual object selection process. It then uses the information to calculate the part coordinates and creates the part by sending the appropriate commands to the OpenGL engine as well as making the addition to the master part list.

Serial drivers

The serial drivers consist of the codes that establish connections with the pan-tilt motors, joystick, and laser range finder and communicate with these devices through the serial ports. These drivers run as separate threads of execution so they can continuously monitor the ports without affecting the flow of the rest of the software.

Stereo motion controller

The stereo motion controller corresponds to the stereo head control window. It takes input from the operator to adjust the position of the pan-tilt head in situations such as the initial view selection and laser point selection and as such must communicate these commands through the serial drivers.

Joystick command interpreter

This process is another independently running task, which enables the operator to control the camera head from a joystick by using input commands and converting them into signals to be sent to the pan-tilt mechanism.

Master part list

The master part list is a Clist of part information, including part type, size, schedule, location, orientation, etc., as specified by an 'objectInfo' structure in the header file.

Stereo autoscan object selection

This process corresponds to the stereo autoscan window. It displays the static images chosen initially in the panoramic view screen and receives from the operator the part type and attributes to be located by the image processing. When the part information and an image fragment are selected, they are sent to an autoscan client.

Stereo autoscan client

The autoscan client is a separately running thread of execution that gathers the operator defined information and sends it to the autoscan server. The client waits until the server finishes its image processing on the image fragments and sends back the part location information. Then the client places the part information into a temporary Clist and notifies the operator that a list of part locations is ready for insertion into the OpenGL model.

Stereo autoscan server

The stereo autoscan server is an independent executable that runs on a separate computer from RTSA. Once the autoscan server starts, it waits for contact from the stereo

autoscan client. When the client sends part information and an image, the server also calculates the part position and orientation in the camera coordinate frame to send back to the client. Data communication is handled over a socket.

Range autoscan object selection

This process corresponds to the range autoscan window. It displays the range images gathered by a range camera and sent to RTSA and receives from the operator the part type and attributes to be located by the image processing. When the part information and an image fragment are selected, the autoscan window is used to call the range autoscan client.

Range autoscan client

The range autoscan client is a separately running thread of execution that gathers the operator defined information and calls the range autoscan DLL. The client waits until the DLL finishes its image processing on the image fragments and sends back the part location information. Then the client places the part information into a temporary Clist and notifies the operator that a list of part locations is ready for insertion into the OpenGL model.

Range autoscan DLL

The range autoscan DLL is a library of image processing functions that may be called by the range autoscan client. It processes range images and returns part location and orientation information. Since it is a dynamic link library, it may exist on a separate computer to avoid monopolizing CPU time.

Temporary part list

The temporary part lists are Clists of parts returned by the autoscan functions. They exist as member variables and are erased when the operator chooses to insert them into the OpenGL model. This temporary location for autoscan information was created since automatic insertion of parts can interfere with manual insertion of parts at certain stages of manual modeling. In this way, the operator can finish with manual modeling before automatically modeled parts are added to the OpenGL environment.

Task planner

The task planner is a large set of functions responsible for creating a task plan from operator and model information. It provides the operator with a set of windows to select the object to be cut, cutting planes, tools to use, etc. It uses graphical displays of selections of parts and cutting planes in OpenGL and part information from the master part list. The final result is a linked list of actions, which are downloaded as a text file through the Ethernet to the control computer.

Assistance planner

The assistance planner is a large set of functions that interact with the task planner. It provides the operator several teleoperation assistance methods to be included at various stages of a task plan. This planner uses graphical displays of selections of parts and motion constraints in OpenGL and part information from the master part list.

Task plan

The task plan is a text file consisting of the set of actions compiled by the task planner. It is downloaded through ethernet from the task planner to the control computer and written in a format that the controller can parse.

6.3.2 Human-Machine Cooperative Telerobotics Controller

The HMCTR controller is created in the Real-Time Innovations (RTI) product Constellation. It incorporates a finite state machine and continuous controller, which receive task plans and commands from the master controller and command motion of the Schilling manipulator.

Constellation is the development tool created by Real-Time Innovations (RTI) used to create the HMCTR controller. It is a graphical component-based tool that provides some automatic generation of code to build and maintain real-time applications quickly and easily. It allows users to create continuous flow diagrams graphically and link them with finite state machines to dictate the behavior of the system. Programmers may use previously written components that are provided by RTI in a repository or generate special purpose blocks into which user-written code is integrated. The control system is then complied for the desired machine and executes the control strategy in real-time.

Below the application level, there are objects of several types that may be found in Constellation. The Composite Object Group (COG) may encapsulate both sample-data system elements and event-driven elements. The Finite State Machine (FSM) is also a composite object that consists of a state transition diagram that represents an event-driven program. The FSM usually appears in its file with associated continuous flow diagrams that are responsible for providing stimuli and running state transition components. There are also three types of primitive components in Constellation: the state transition component (STC), the data flow component (DFC), and the atomic component (ATC). The state transition component provides actions taken by a FSM in response to a stimulus.

The data flow component is the building block for the sampled-data part of the system that executes routines at every sample period. The atomic component provides generic utilities or functions to other components in the same diagram. Connections between components in Constellation take one of three forms. Pin connections are inputs and outputs of data from components. Bubble connections provide utility of functions from one method to another. Interfaces allow pins and bubbles to be bundled in a unified connection. Another important aspect of Constellation is the ability to define operating modes. A mode is a set of active components which may be enabled and disabled as a unit, allowing the event-driven part of the system to alter the system behavior automatically in response to given stimuli.

This section describes the HMCTR control system being developed in Constellation.

Top Level controller

At the highest level of the HMCTR controller, there is one Composite Object Group (COG), which contains all other components, shown in Figure 6.3.3. In the main COG, there are two primary COGs: the robot discrete controller and the robot continuous controller, as shown in Figure 6.3.4. The discrete controller is responsible for determining the mode of operation of the controller, whether initiated by the operator or by the downloaded task plan. The continuous controller receives data and commands from the discrete controller to enable predefined sets of components so that the controller behaves in the desired way.

Robot Controller (Continuous)

The robot continuous controller shown in Figure 6.3.5, is the data flow diagram which encompasses the various capabilities for modes of operation of the manipulator

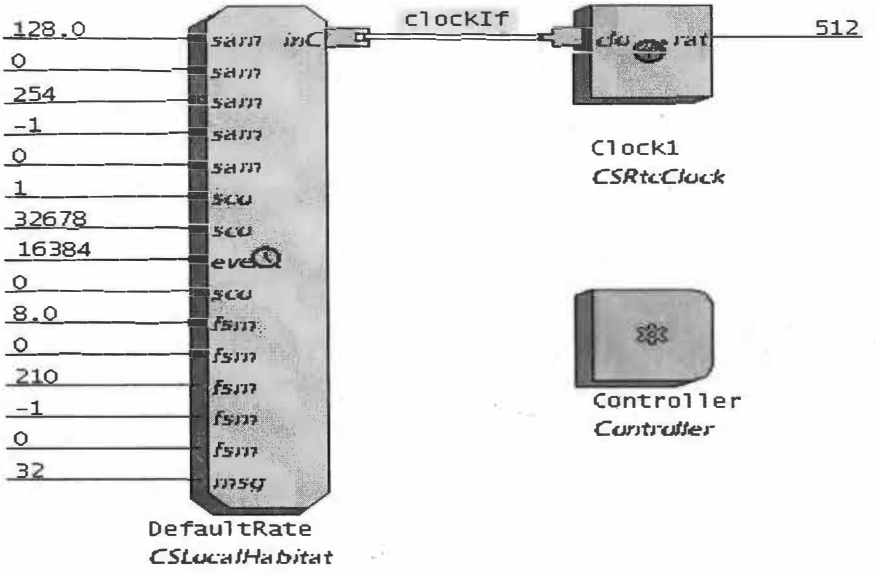


Figure 6.3.3 Top level controller

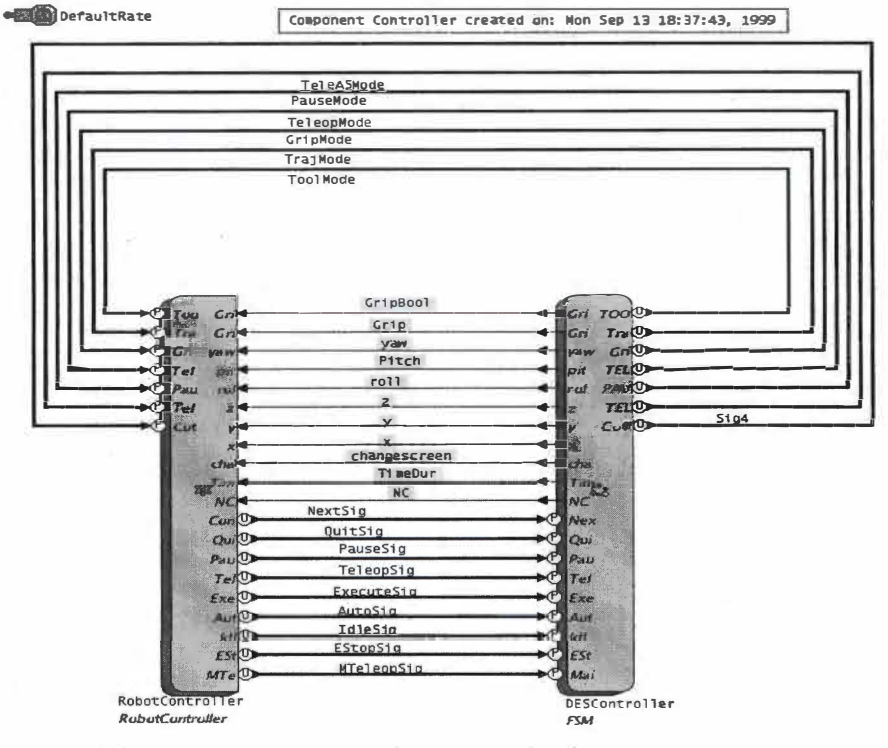


Figure 6.3.4 Robot continuous and discrete control COGs

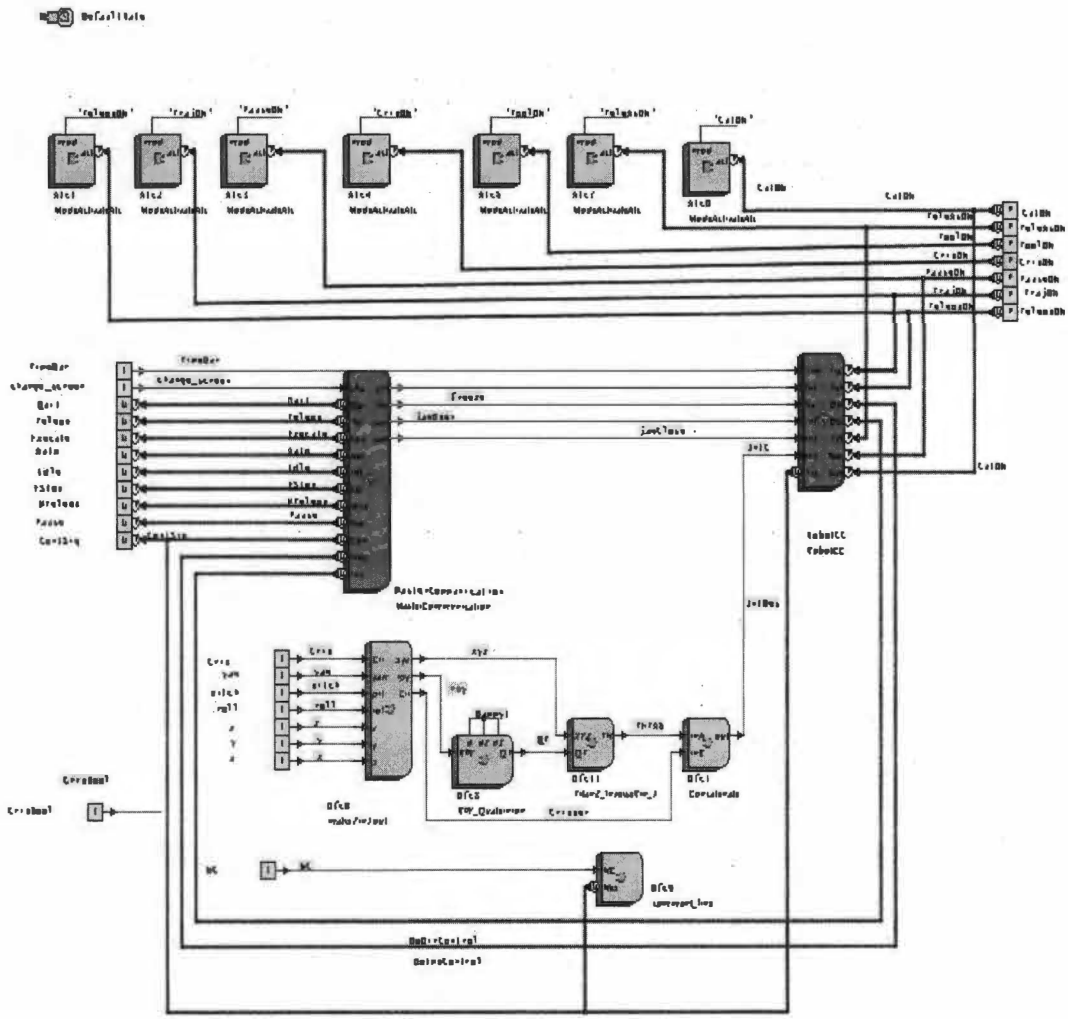


Figure 6.3.5 Continuous control COG elements

and sends and retrieves information from the lower level control loops in the robot closed loop control COG. The MasterCommunication and Robot components are always activated during operation of the system (The red blocks), but in the other four modes of operation available, a unique set of components is enabled. In the PauseON mode, TeleopON or TeleopAsON (teleoperation with assistance function) no additional components are activated. In the TrajON or ForceON mode, Component Dfc0, Dfc7, Dfc8 and XYZQT_G will be activated.

There are two main sources for robot control signals in this diagram: one the signals from discrete controller, and the other one is the MasterCommunication component. These components are mutually exclusive sources for control, and only one at a time will be enabled in a given mode of operation. Each of the two also has access to the SendStimulus component "NEXT" in the FSM subchain so that when it is finished executing, it can signal for the next action in the plan to be initiated.

There are several ActivateMode components in this diagram which cause the mode to be changed when called from various sources throughout the diagram. The components make it possible to activate the PauseON, TeleopON, TeleopAsON, TrajON, or ForceON mode by providing the proper function to users elsewhere in the system.

The MasterComm component is the communication component with the minimaster. It is always enabled so that button information from the master may be used to transition from state to state regardless of the current state of the system.

The RobotCC component in this diagram receives high level position commands from either autonomous or manual inputs and outputs its actual position.

Dfc0, Dfc8, Dfc7, and XYZQT_G receive the final point of the trajectory in Cartesian space from the task plan and convert it into joint space and send it to trajectory generator in the low level continuous controller.

Mastercommunication

The MasterCommunication component is the communication component with the minimaster. It is always enabled so that button information from the master may be used to transition from state to state regardless of the current state of the system. Its 'use' bubbles each correspond to a different button function, rather than to a physical button, so that the same button may call different functions while the master is in different submenus. The MasterComm component also has two continuous outputs, the GripperPos and JntPos signals. Figure 6.3.6 shows the structure of the Mastercommunication component.

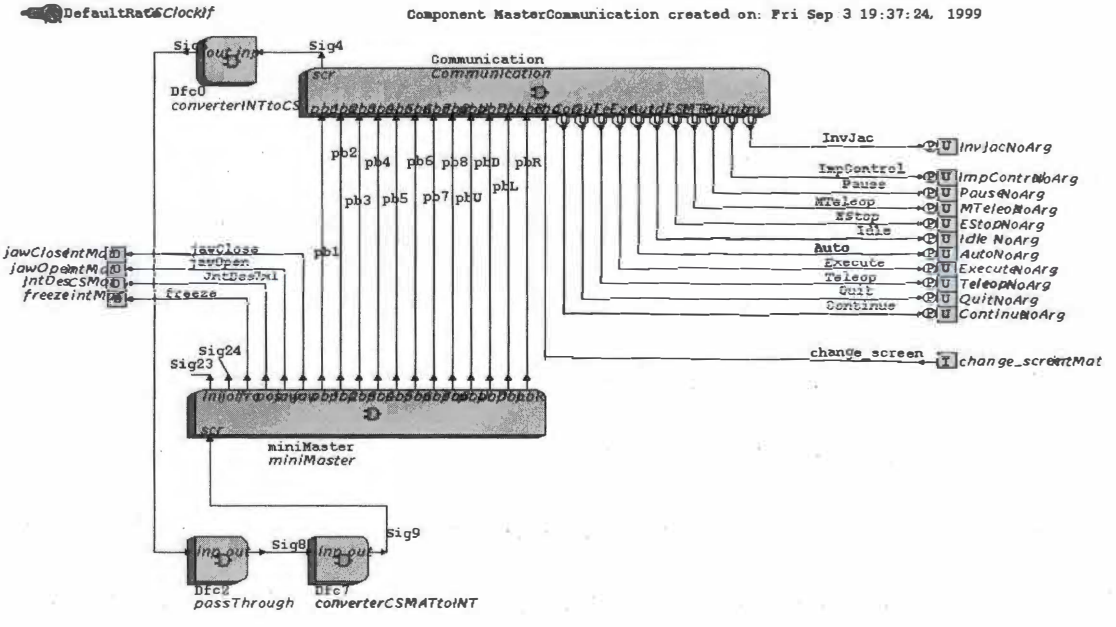


Figure 6.3.6 Communication structure

RobotCC COG

RobotCC COG, shown in Figures 6.3.7 through Figure 6.3.13, is the robot low-level closed loop control structure. In addition to the direct joint to joint control (pure teleoperation) between the master and slave, three different continuous closed loop control schemes are considered: teleoperation with assistance function control, subtask autonomous control, and grey prediction force/position parallel autonomous control which get control signal from a joint space trajectory generator. At the beginning of the task execution, the operator selects one of these four control strategies in which to operate with the master console. That button sends the command to activate one of the following modes of the control execution: "Teleop", "Autonomous", or read "Teleop_assist" and "CutON" from task plan by using a build-in ActivateMode component in the robot closed loop control diagram. Each of these modes enables a different set of the components. For example, if teleoperation control is selected, the blocks in red are activated (Figure 6.3.7, Figure 6.3.8). Figure 6.3.9 and Figure 6.3.10 show the mappings for the autonomous control. Analogous to the teleoperation control, Figure 6.3.7 and Figure 6.3.11 show the mapping for the teleoperation with assistance function control. When Force prediction mode is activated, the components mapping is shown Figure 6.3.9 and Figure 6.3.12. Figure 6.3.13 is the lowest level force loop which calculates the force errors.

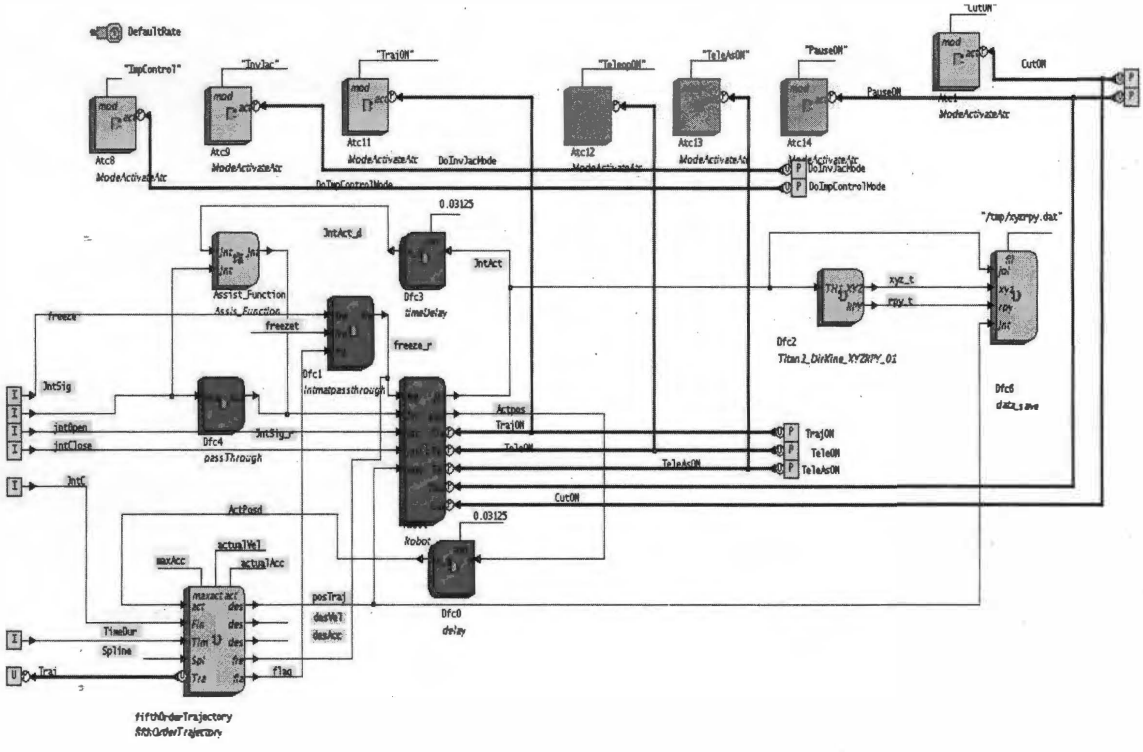


Figure 6.3.7 Closed loop control components with teleoperation mode activated

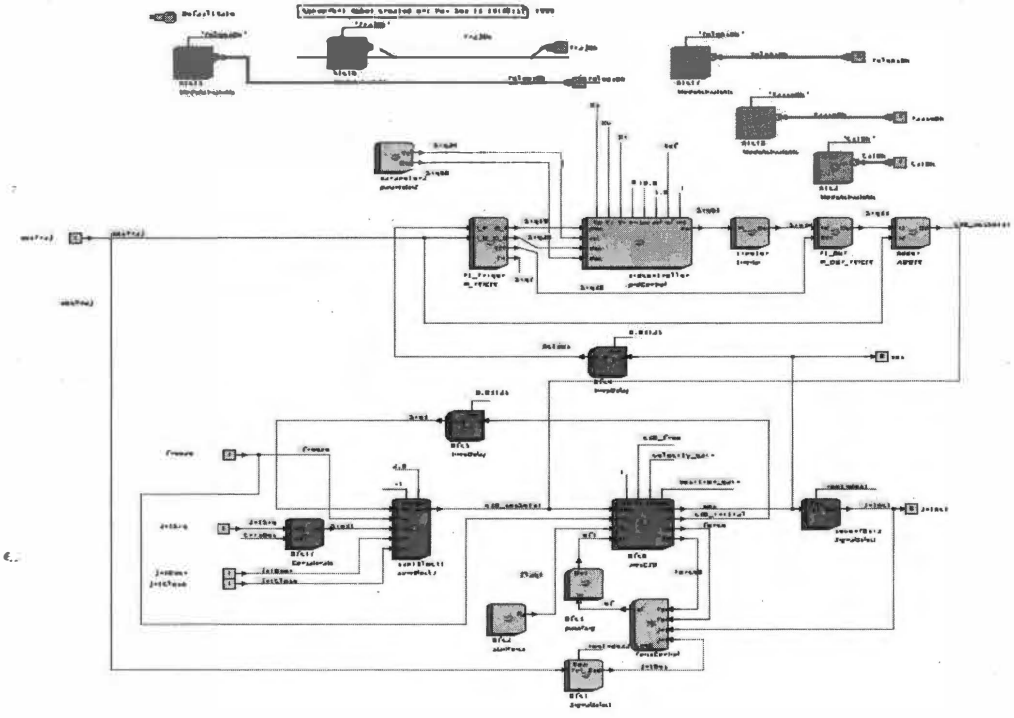


Figure 6.3.8 Lowest level control components with teleoperation w/o assistance

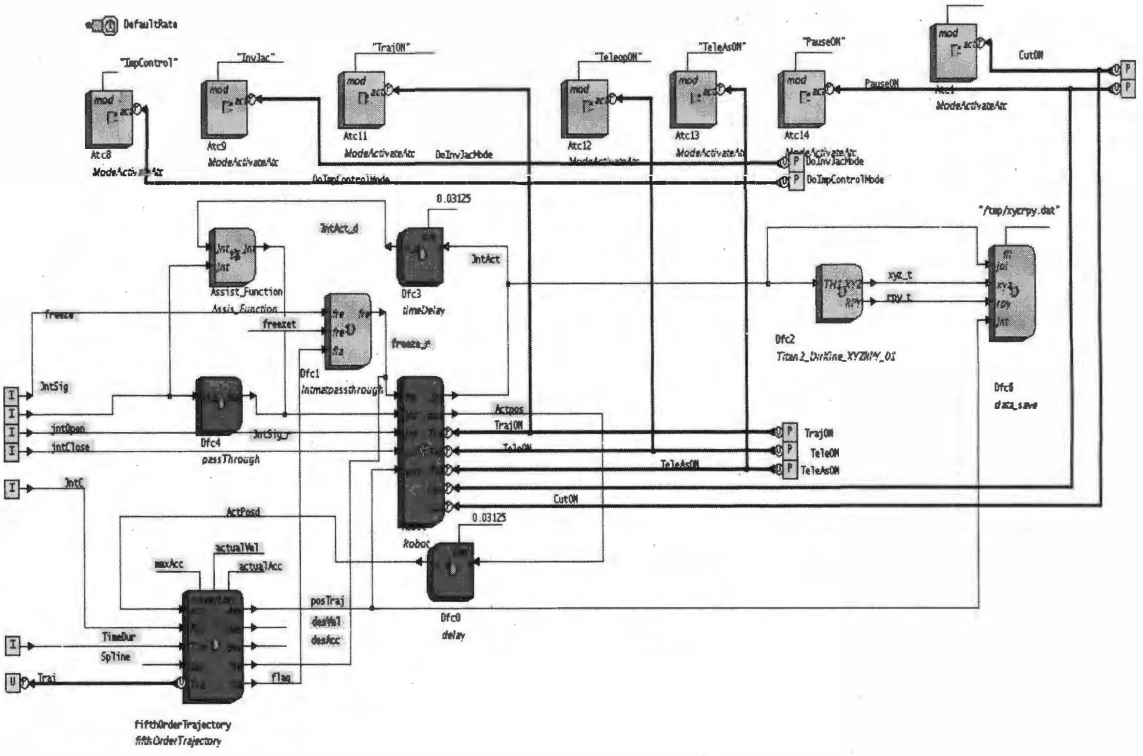


Figure 6.3.9 Closed loop control components with autonomous mode activated.

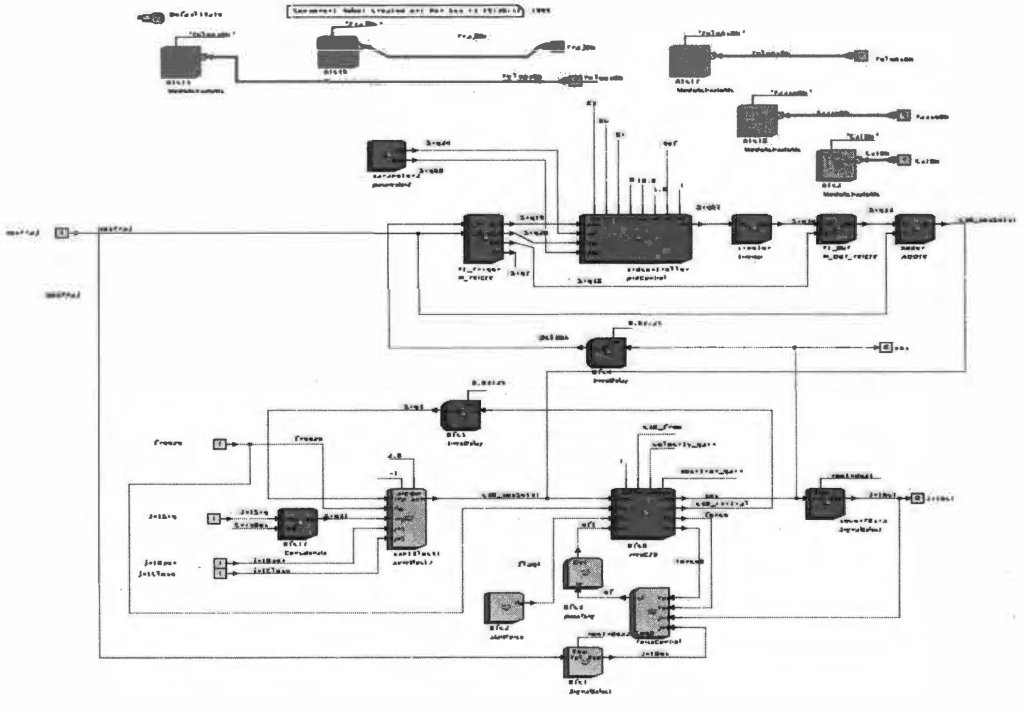


Figure 6.3.10 Lowest level control components with autonomous mode activated.

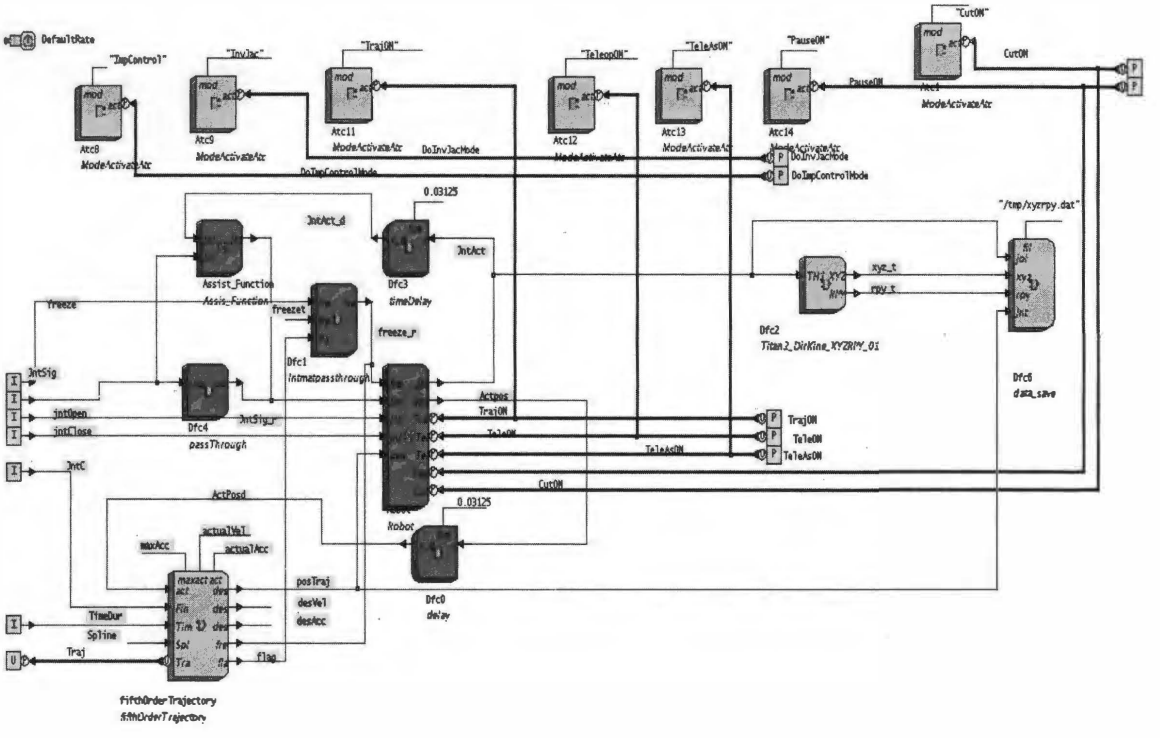


Figure 6.3.11 Components with teleoperation with assistance mode activated.

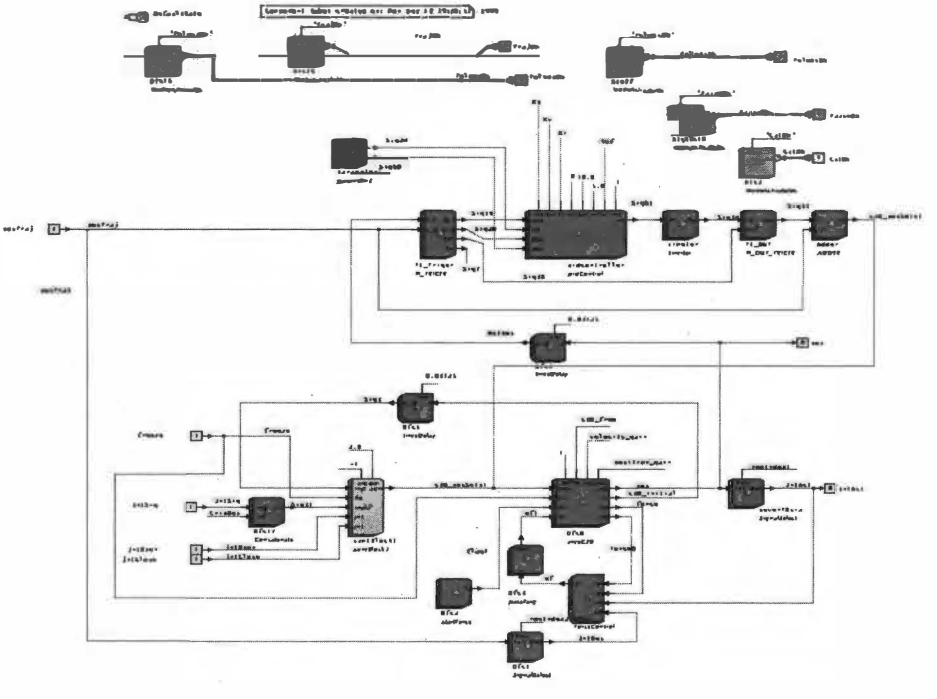


Figure 6.3.12 Components with force prediction control mode activated.

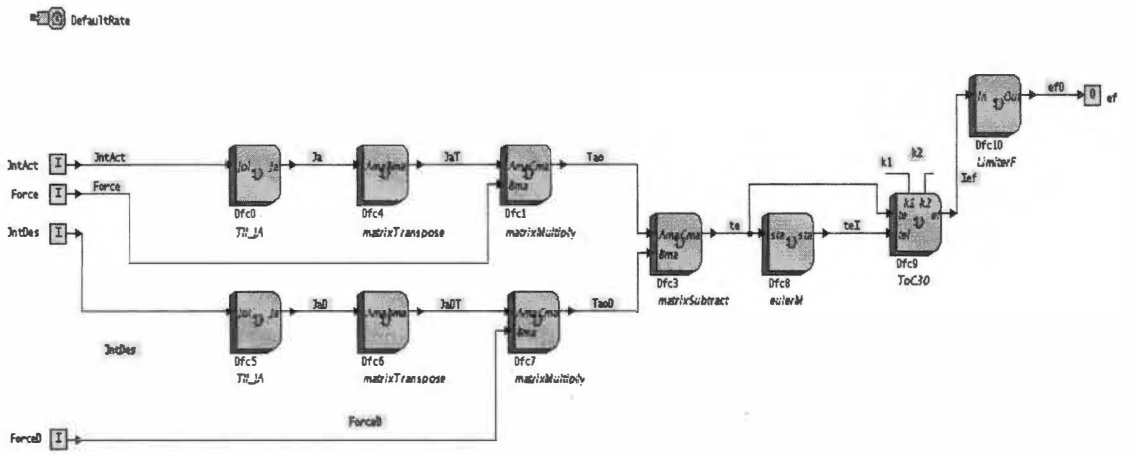
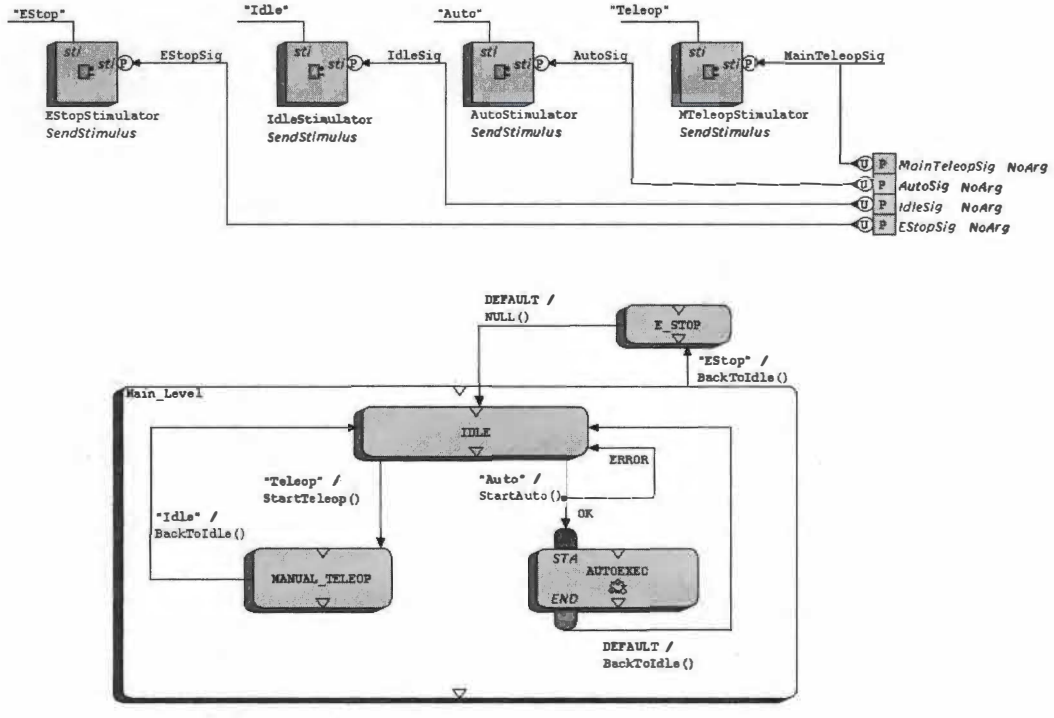


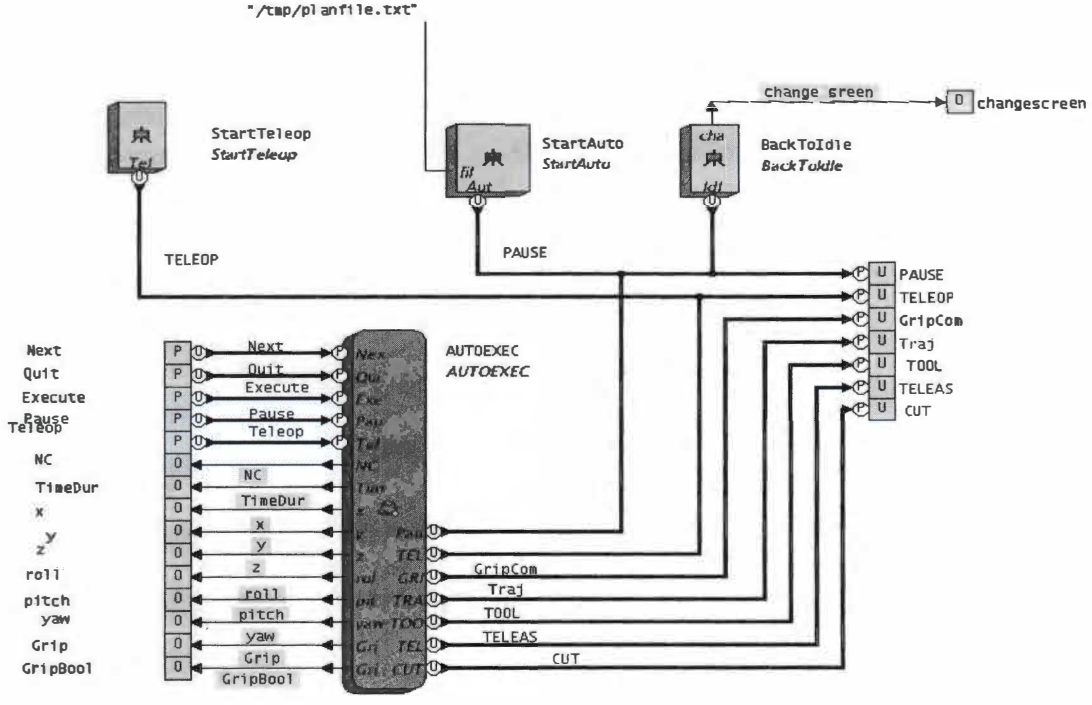
Figure 6.3.13 Force control loop of force prediction mode activated.

DesController (Finite State Machine)

The components which comprise the DesController COG which decide robot work mode are shown in Figure 6.3.14. At this level, there is a main finite state machine that determines whether the robot is being controlled automatically or manually in the absence of a plan or if it is idle. The system may transition from IDLE to either the autonomous execution state AUTOEXEC or the teleoperation state MANUAL_TELEOP given the stimuli "AUTOEXEC" or "MANUAL_TELEOP", respectively. The AUTOEXEC state is actually composed of a finite state machine subchain described in the following section. Upon transition to one of these states, the state transition component StartTeleop ( ) or SartAuto ( ) runs. When the state is transitioned back to idle, the transition component BackToIdle ( ) runs. The purpose of these transition components is to verify that the proper hardware is connected and operational, and configure the system to run in the desired mode. If there is a problem in the transitions out of the IDLE state, the value ERROR is returned, and the system state returns to IDLE.



(a)



(b)

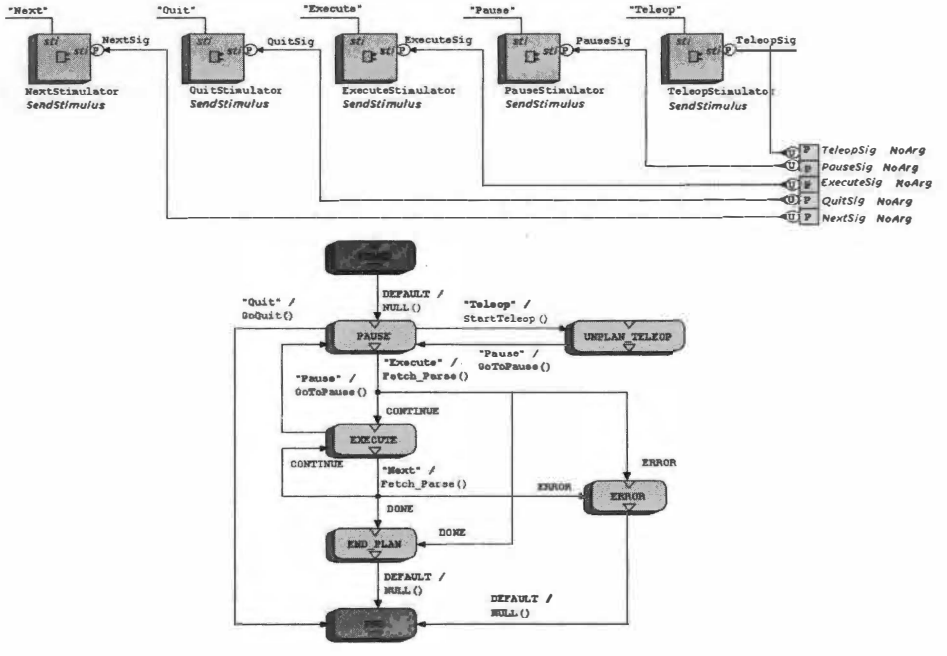
Figure 6.3.14 DES controller COG components

If everything checks out, the return value is OK and the system transitions to the desired state. There is also an ESTOP state where the system can transition to from any other state if there has been an emergency stop. From there, it can only transition back to the IDLE state.

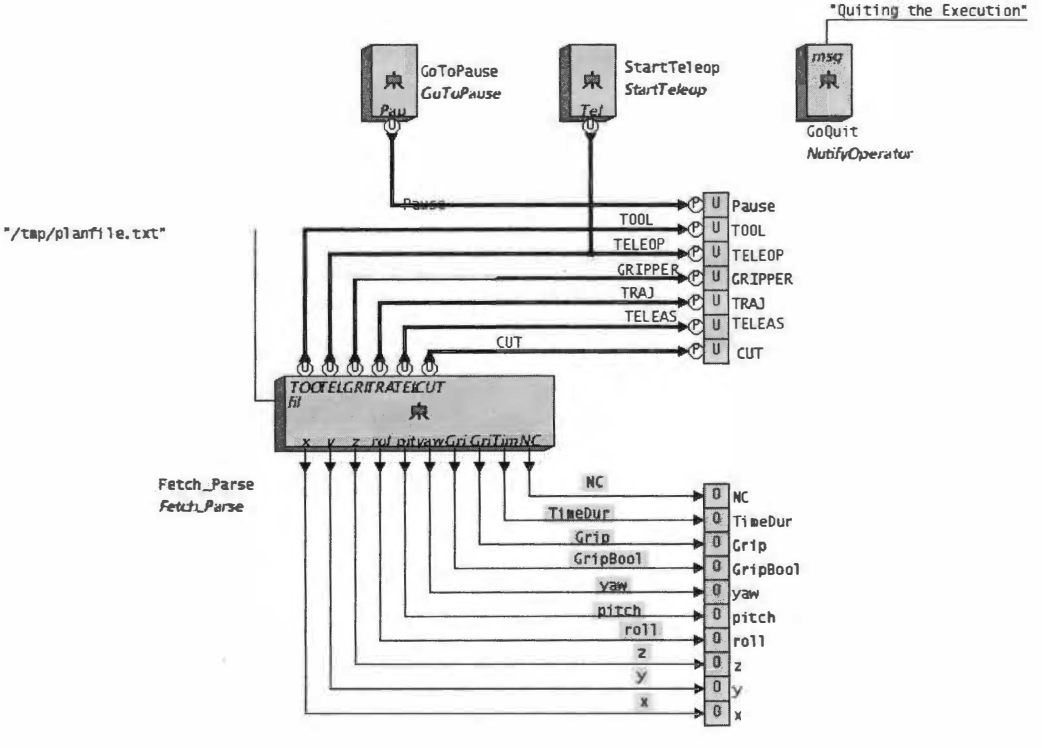
Accompanying the main finite state machine in the same file is a set of components which contain the transition and other code associated with this FSM. At this level, there are three state transition components as described above: StartAuto( ), StartTeleop( ), and BackToIdle( ). Each of these is a user of an ActivateMode component found in the robot continuous control diagram (see Figure 6.3.5) which enables the set of components corresponding to that state. Also found in this diagram are the components which provide code to the master console buttons to send stimuli such as "ESTOP", "IDLE", "AUTOEXEC", and "TELEOP" which cause the main FSM to transition. The primary component in this diagram accompanying the FSM is component corresponding to the automatic plan execution subchain. It is user and provider of several functions which exist in the robot continuous control diagram, as well as providing some data for that controller originating from the execution plan.

Autonomous execution mode control subchain

Since this is an FSM subchain (Figure 6.3.15), there are two main parts to the file, the subchain itself and the associated data flow components. THE FSM subchain is responsible for defining the current behavior of the system as determined by operator and automatic stimuli, and the data flow blocks are responsible for retrieving planned execution steps and enabling the proper set of components for the given action.



(a)



(b)

Figure 6.3.15 Autonomous execution subchain components

The FSM subchain consists of seven states, including START, PAUSE, EXEC, UNPLAN TELEOP, END PLAN, END, and ERROR. The START state is a beginning condition when the subchain is entered from the main FSM, and automatically transitions to the PAUSE state, where the system waits for the operator to signal for execution of a predefined plan or other desired action. From PAUSE, the operator may send stimuli associated with master console buttons corresponding to "QUIT", "EXECUTE", or "TELEOP". The "QUIT" stimulus transitions the diagram to an END state in which the subchain exits to the main FSM and to the IDLE state. This signal is used if a plan is to be abandoned and the associated GotoQuit( ) transition component disables the appropriate components. The "TELEOP" stimulus is sent if the operator wishes to initiate an episode of teleoperation, which has not been incorporated into the downloaded plan. It is used in the case of an unforeseen obstacle or change in approach to a task. The StartTeleop( ) transition component runs at this point to activate the teleoperation mode by using an ActivateMode component in the robot continuous control diagram (see Figure 6.3.5). Transition back to the PAUSE state from the UNPLAN TELEOP state is accomplished by sending the "PAUSE" stimulus and the appropriate ActivateMode component is called by the GotoPause( ) transition component.

The "EXECUTE" stimulus initiates or restarts the execution of the downloaded task plan. The FetchandParse( ) state transition component accompanies it and the "NEXT" stimulus in the transition out of the EXECUTE state. It is responsible for opening the task file on its first call, retrieving the first unexecuted action from this file, calling the appropriate ActivateMode components in the robot continuous control diagram, and setting values for the gripper state for a toggle gripper action or final coordinate of the

manipulator for a move action. Once modes are changed and values are set, it returns one of three values: CONT, DONE, or ERROR. The CONT return value results in a branch, which takes the diagram back to the EXEC state, where the next action can be retrieved upon completion of the previous action. The DONE return value signals that the action retrieved was the last in the file and the diagram should transition to an end state to wait for this last action to be completed before it transitions out of the subchain. The ERROR return value indicates that the file has been corrupted and the system should display an error message and transition out of the autonomous execution subchain. The "NEXT" stimulus may be sent by the SendStimulus component in this diagram, which is a provider to several other components in the continuous control diagram. It may be sent by a component executing an automatic action when that action has finished, or by the operator signaling that a planned episode of teleoperation has been completed.

The END PLAN state was included so that a final action could be completed before transitioning out of the AUTOEXEC subchain. When in this state, the "NEXT" signal sent to indicate a previous action is complete will cause the transition to the END state where the diagram automatically transitions back to the main FSM.

6.3.3 Control Interface

Constellation text interface

Constellation provides access to a run-time menu which can be used to alter operating modes or variable values during execution. Additional custom menu items in the Schilling telerobotic system will allow the operator to inspect, edit, exit, and re-enter the task plan which has been downloaded when desired. When the operator wants to see or to

edit the task plan, this menu option may be invoked by typing the command string 'task plan'.

Other option of the Constellation text interface with the operator is C++ print messages. If the C++ code for the corresponding Constellation function includes any print message commands this message will be printed on the display each time function is executed. For example, if Constellation is unable to open the TaskPlan.txt file during the FetchandParse function execution, a message such as "Task plan file cannot be opened. Returning to 'Idle' State" will be displayed. The message communication between Constellation and the operator allows the operator to always be aware of what is going on during the program execution.

Minimaster Menu Screens

The easiest way for the operator to communicate with the controller is through the minimaster buttons. They are accessed in the software by using the MasterCommunication interface. The master has twelve different buttons which can be used to activate the different stimuli in Constellation. By defining different submenu screens on the minimaster, the same buttons can be used to call several different functions.

Each signal in the miniMaster block corresponds to a different button on the Minimaster. When a button is pressed, a Boolean signal is sent to the Communication block. Depending on the submenu screen number in the Communication block, each signal is used to call the corresponding function through a "bubble" provided by code outside this file. In addition, when the submenu must be changed, the Communication block sends the "screen" Boolean signal to the miniMaster.

Two different submenu screens have been created. Figure 6.3.16 shows screen number 1 corresponding to the beginning of the program or each time program has to be restarted.

After the operator makes a selection and pushes the corresponding button on the Minimaster, the screen will be changed to the next one using the Boolean "screen" signal from the Communication block. The submenu screen shown in Figure 6.3.17 corresponds to the Autoexec state of the system.

6.3.4 Task Plan File

The task plan file is a text document containing the sequence of atomic actions that are to be performed by the telerobotic system. Each action is fully described by a C structure teleoperation state, and final position information for move commands. It is downloaded over the ethernet from the task planner to the real-time control computer when the operator is satisfied it is complete. The format of the file is such that it can be parsed by a state transition component in Constellation called 'fetchandparse'.

```
        SCHILLING DEVELOPMENT
--------------WELCOME---------------
<-Teleop
<-Autoexec
<-Idle
                              Estop->
-------------------------------------------------------------
```

Figure 6.3.16 Welcome screen in which control scheme is selected

```
        SCHILLING DEVELOPMENT

--------CURRENT STATE: AUTO--------

<-Execute                      Quit->

<-Pause

<-Teleop

<-Continue                    Estop->

-------------------------------------------------------------
```

Figure 6.3.17 Minimaster autonomous execution menu

Chapter 7

Experimental Evaluation of the Tool Interaction Controller

7.1 Introduction

Several experiments were performed to evaluate the performance of the HMCTR system with tool interaction force predictor and parallel force/position fuzzy controller introduced in Chapters 4 and 5, respectively and the results are given in the following sections.

7.2 Experiments and Results

To test the performance of the grey prediction force/position parallel fuzzy control algorithm proposed in Chapter 5, several cutting experiments have been performed by using a bandsaw and the advanced telerobotic system test bed installed in REMSL. The Schilling TII manipulator controlled by a newly developed real-time control system was used to pick up the bandsaw and cut a 2" schedule 40 steal pipe mounting on a test mockup. First, in order to determine a suitable force trajectory, five trail cuttings were performed with only position control until a successful cutting was complete. The results of a successful cutting with 40 seconds cutting time are given in Section 7.2.1. After the desired force trajectory for 40 seconds cutting time was determined, the newly developed control system with grey prediction force/position parallel fuzzy control algorithm was tested. Eight continuous cuttings were successfully completed with 40 seconds cutting time. In order to test the robustness of the interaction force control system, the cutting speed was

increased by reducing the command cutting time from 40 to 35, and 30 seconds. Those pipe cutting experiments have also been successfully completed with the same force trajectory for 40 seconds cutting. To simplify the experiment, only cutting force in x direction and contacting force in z direction of sensor coordinate frame were controlled. The results are shown in Section 7.2.2.

7.2.1 Bandsaw Cutting with Pure Position Control

Figures 7.2.1 and 7.2.2 show that the peak cutting force is around 54 lbf and the impact force is around -30 lbf respectively. Figure 7.2.3 is the shoulder joint (joint 2) trajectories and it shows the manipulator begins to oscillate around 25 seconds. Figures 7.2.4 and 7.2.5 are joint trajectories for joint 3 and 4.

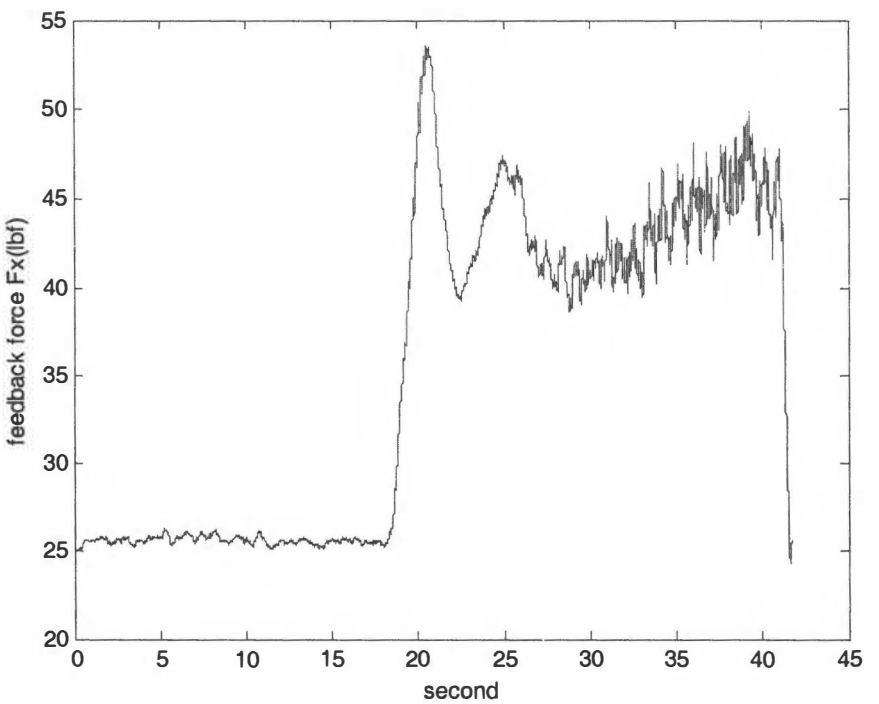


Figure 7.2.1 Cutting force in x direction

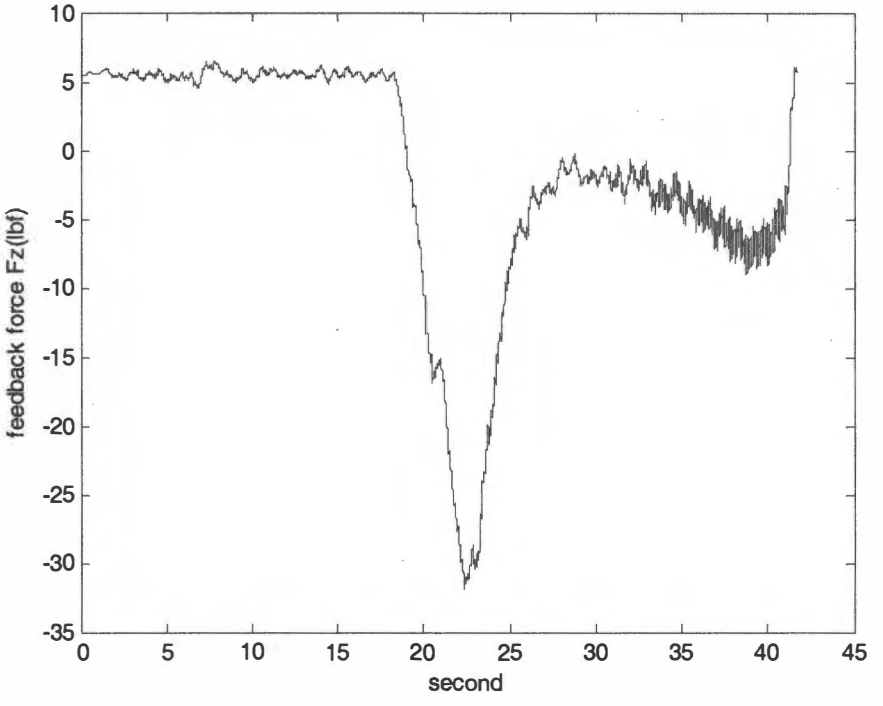


Figure 7.2.2 Interaction force in z direction.

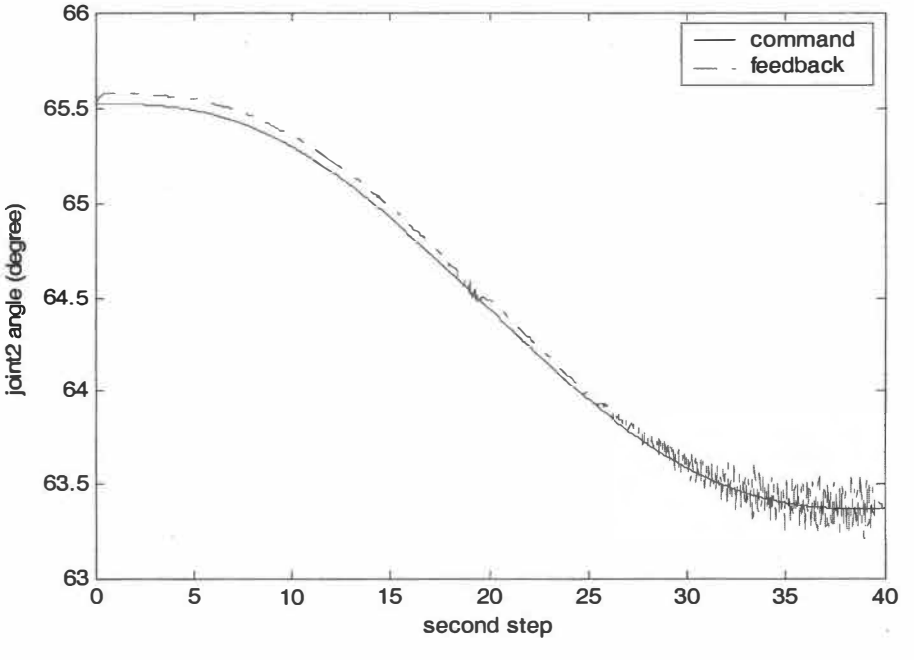


Figure 7.2.3 Joint 2 angles.

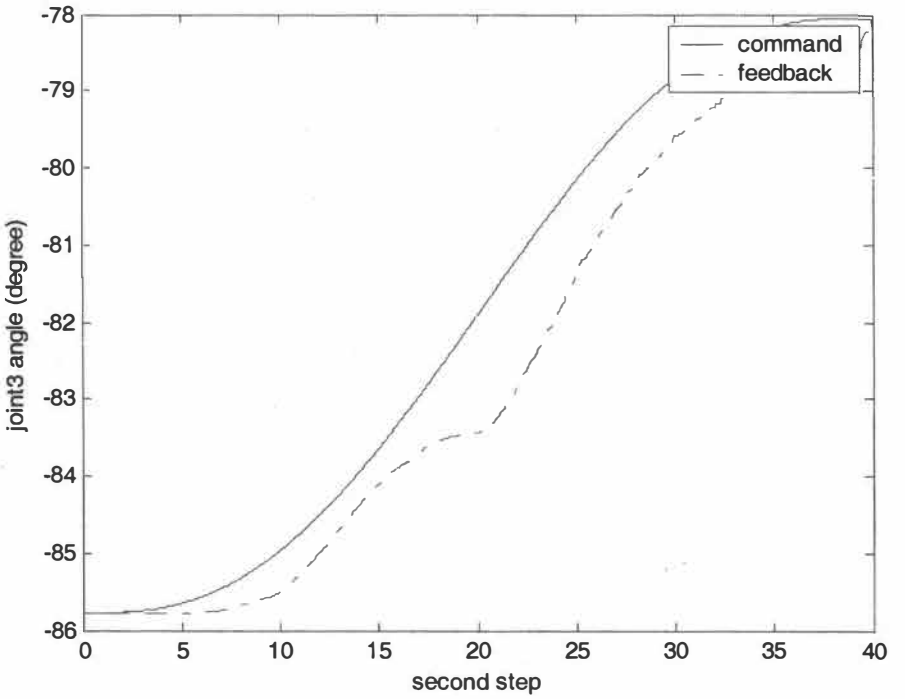


Figure 7.2.4 Joint 3 angles

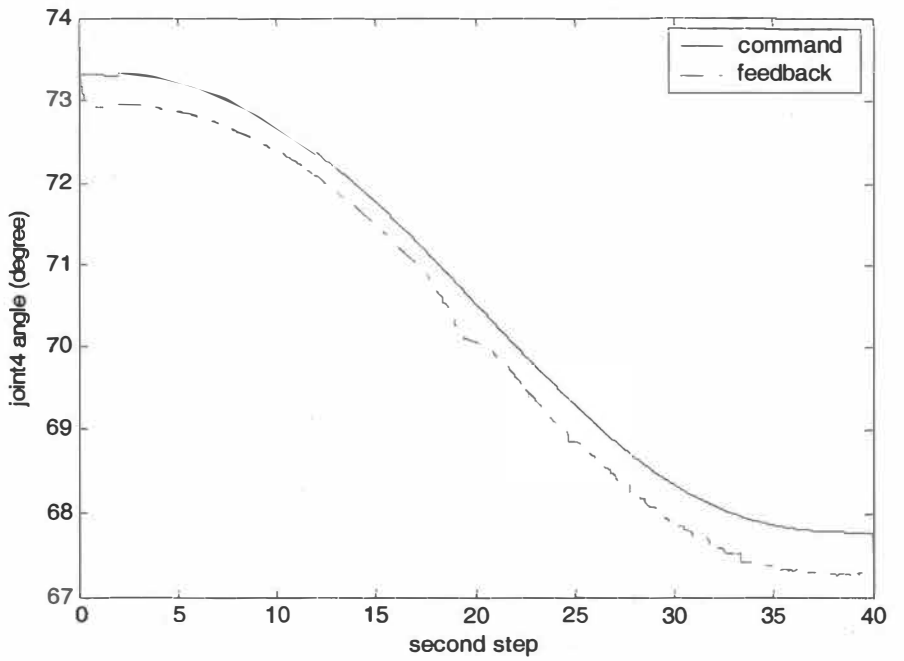


Figure 7.2.5 Joint 4 angles

7.2.2 Bandsaw Cutting with Grey Prediction Control

Figures 7.2.6 – 7.2.10 are the experiment results with 40 seconds cutting time. The peak cutting force in x direction is around 47 lbf as shown in Figure 7.2.6 and the peak impact force in z direction is about -17 lbf as indicated in Figure 7.2.7. There are no vibrations shown in Figures 7.2.8 – 7.2.10. The results of experiment with 35 and 30 seconds command cutting time are shown in Figures 7.2.11 – 7.2.15 and Figures 7.2.16 – 7.2.20, respectively. The peak cutting force and impact force are increased to 54 lbf and -35 lbf for 35 seconds cutting and to 64 lbf and -74lbf for 30 seconds cutting. Even though the interaction forces increased significantly when the cutting speed increased, Figures 7.2.13 – 7.2.15 and Figures 7.2.18 – 7.2.20, show no oscillations in the trajectories of the manipulator joints. The entire HMCTR system is stable during cutting as the cutting speed increases.

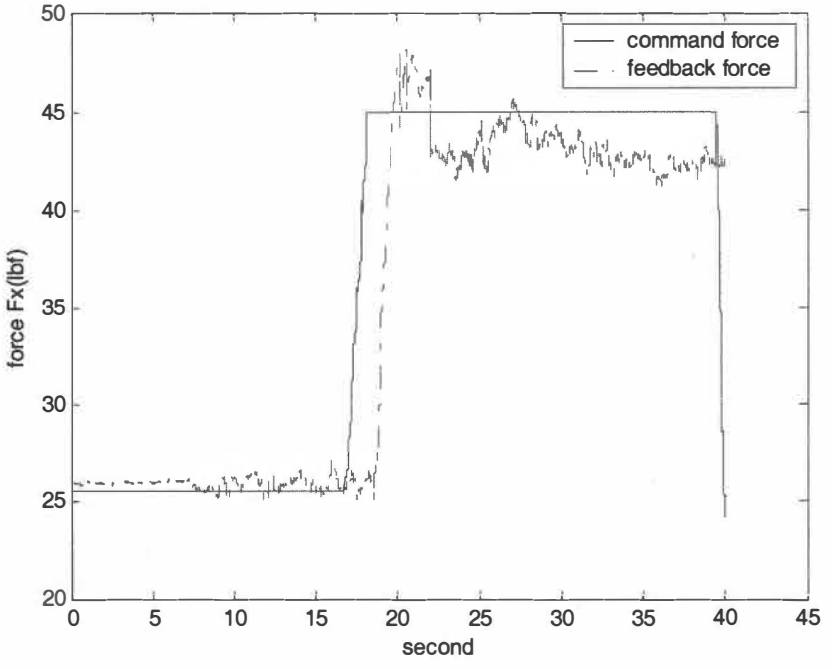


Figure 7.2.6 Cutting force in x direction (40s).

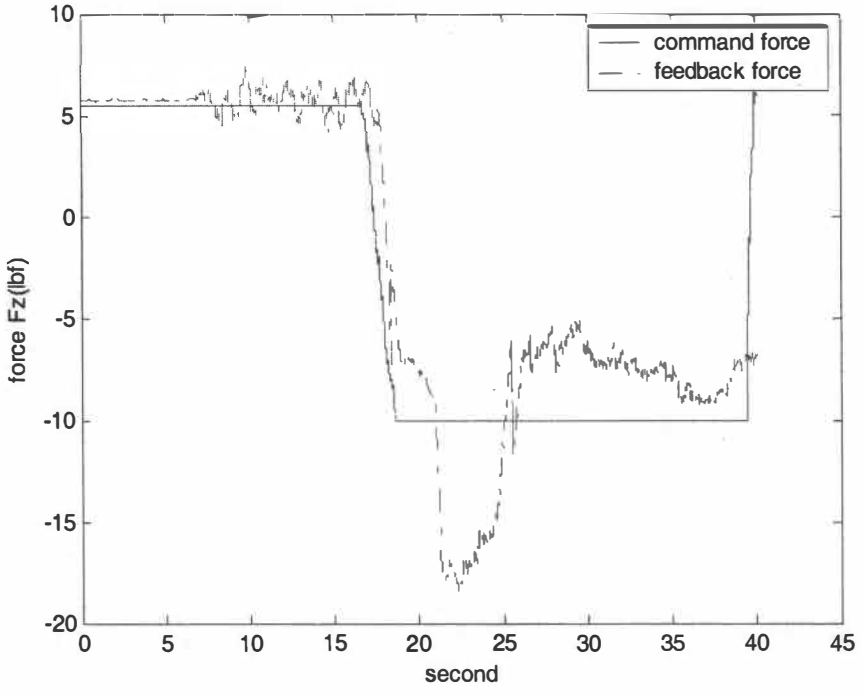


Figure 7.2.7 Interaction force in z direction (40s).

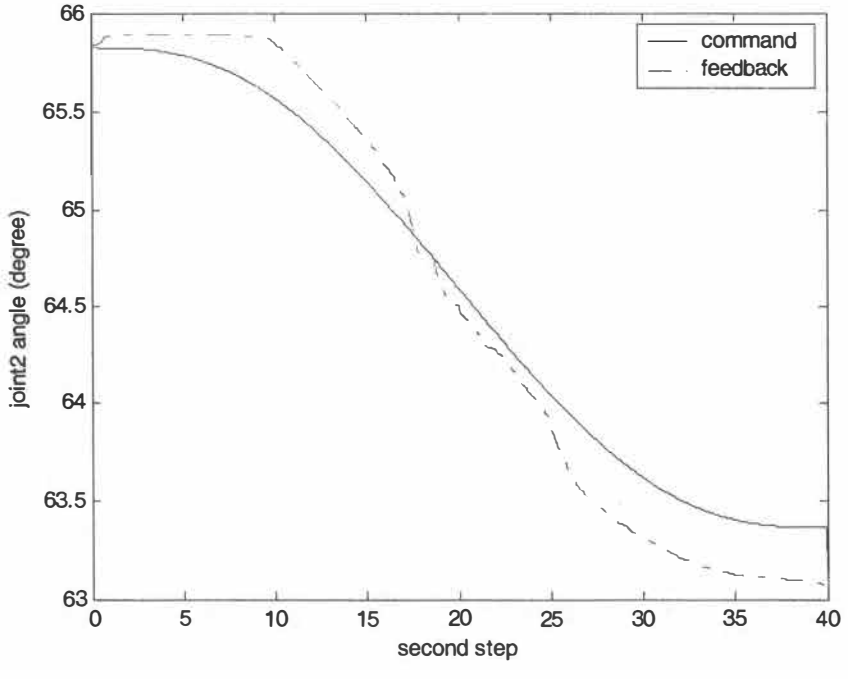


Figure 7.2.8 Joint 2 angles (40s).

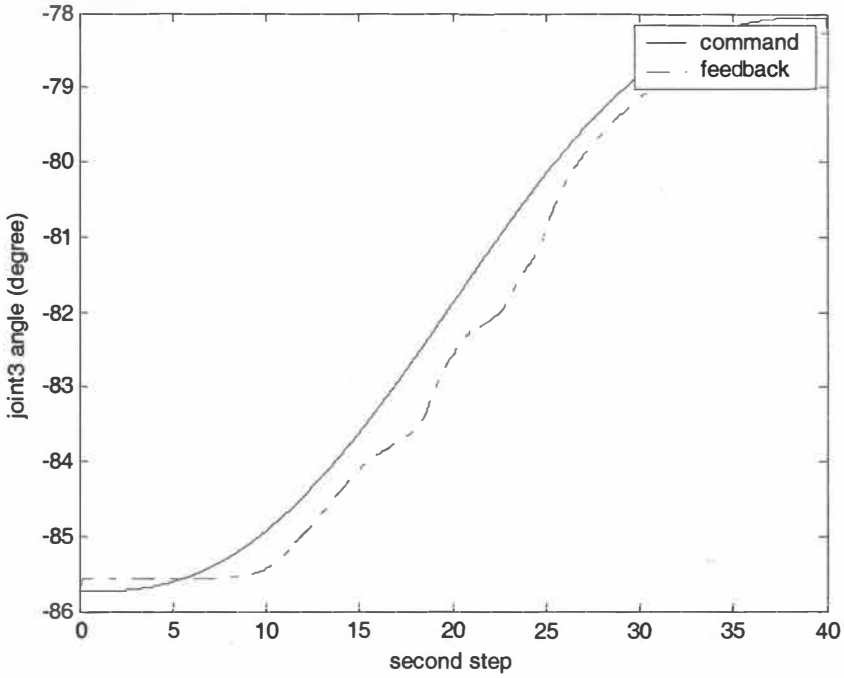


Figure 7.2.9 Joint 3 angles (40s).

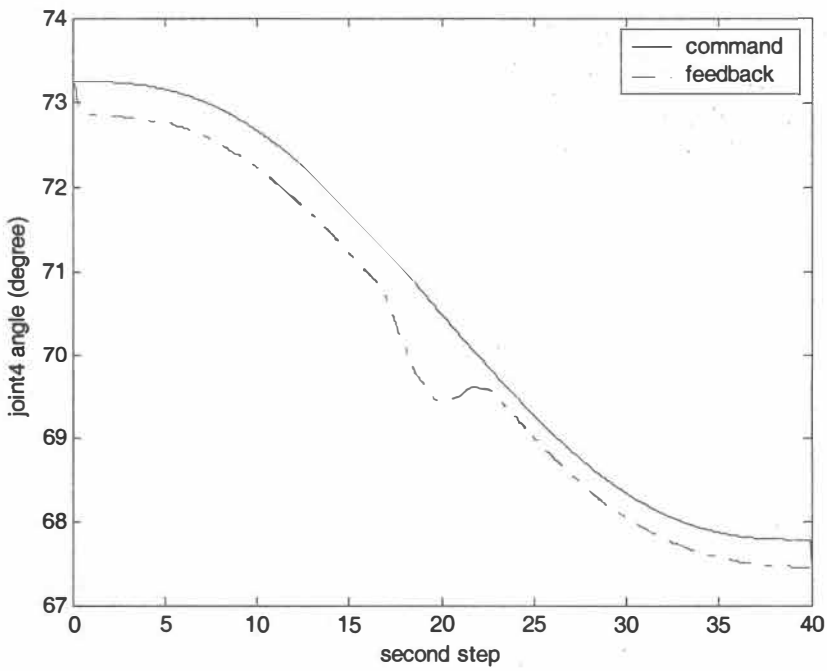


Figure 7.2.10 Joint 4 angles (40s)

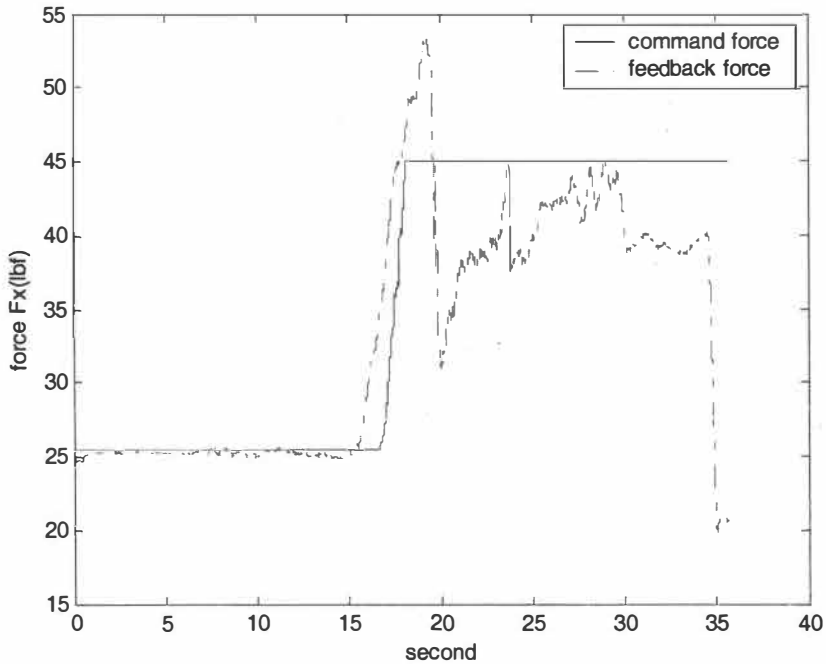


Figure 7.2.11 Cutting force in x direction (35s).

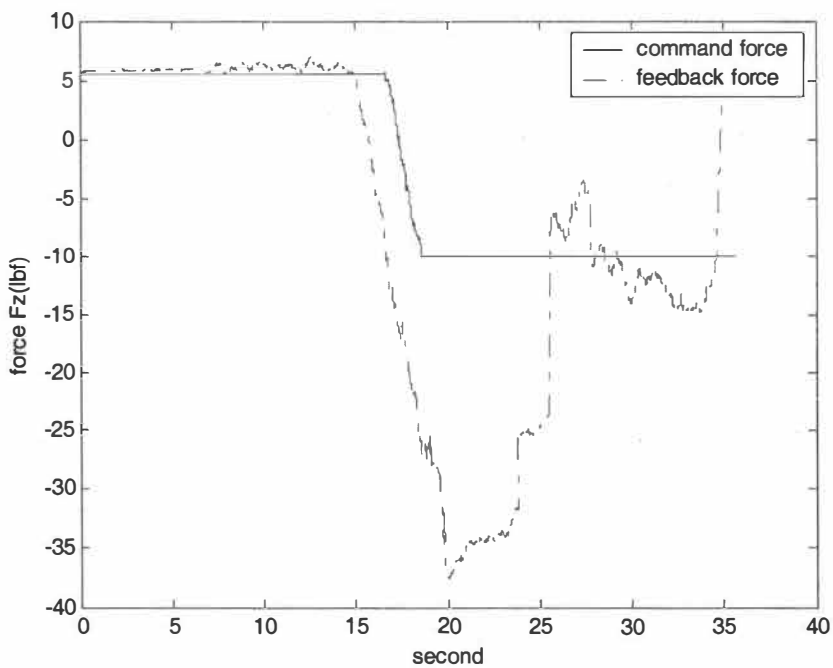


Figure 7.2.12 Interaction force in z direction (35s).

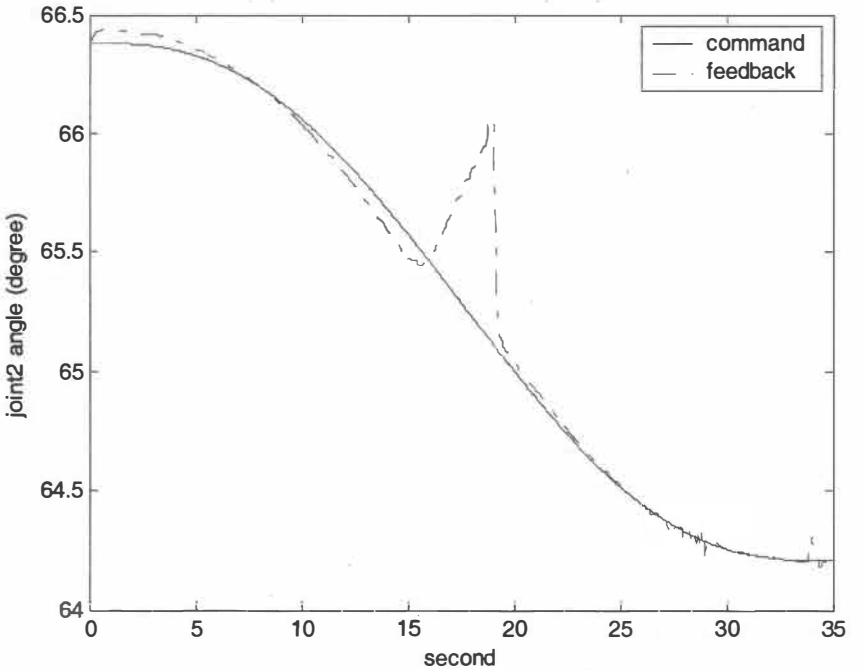


Figure 7.2.13 Joint 2 angles (35s).

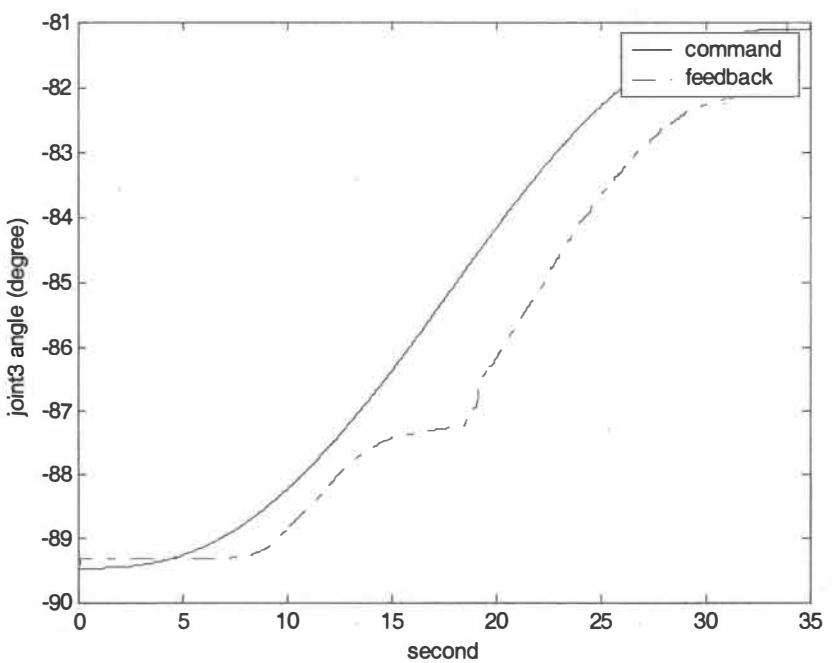


Figure 7.2.14 Joint 3 angles (35s).

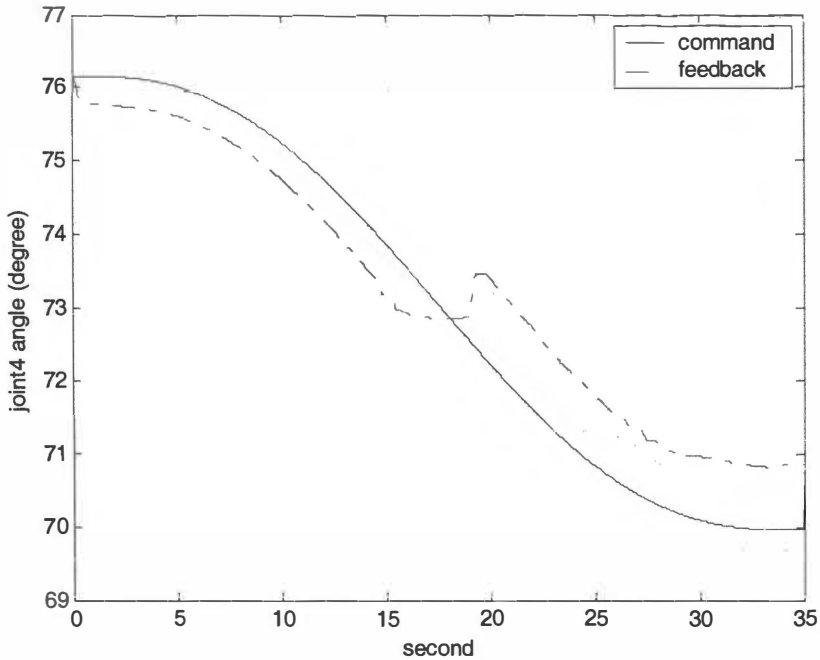


Figure 7.2.15 Joint 4 angles (35s).

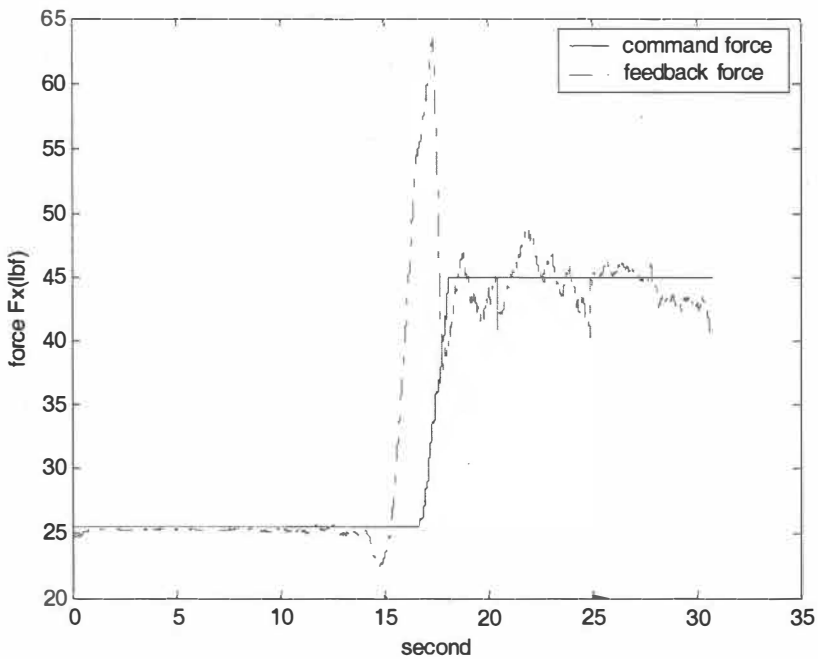


Figure 7.2.16 Cutting force in x direction (30s).

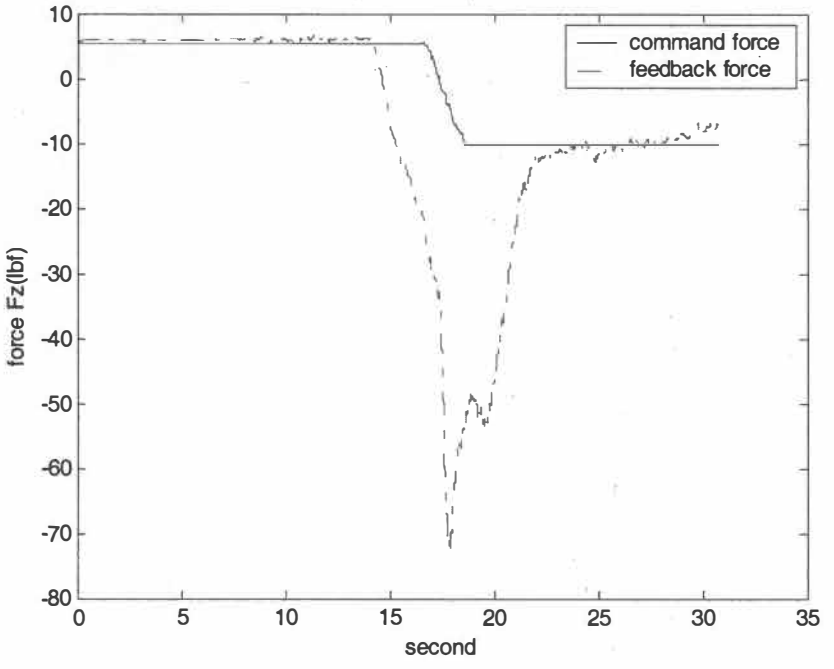


Figure 7.2.17 Interaction force in z direction (30s).

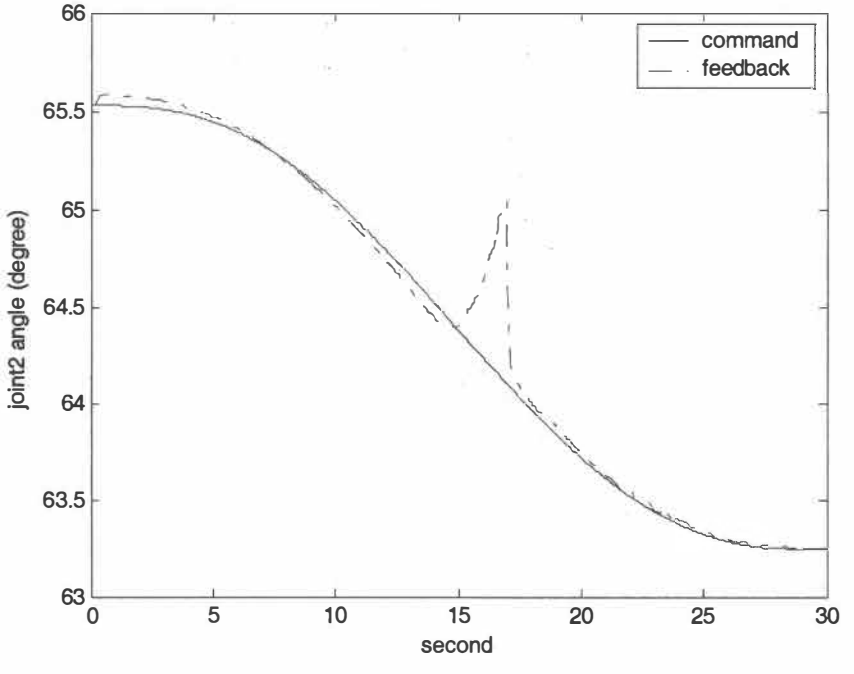


Figure 7.2.18 Joint 2 angles (30s).

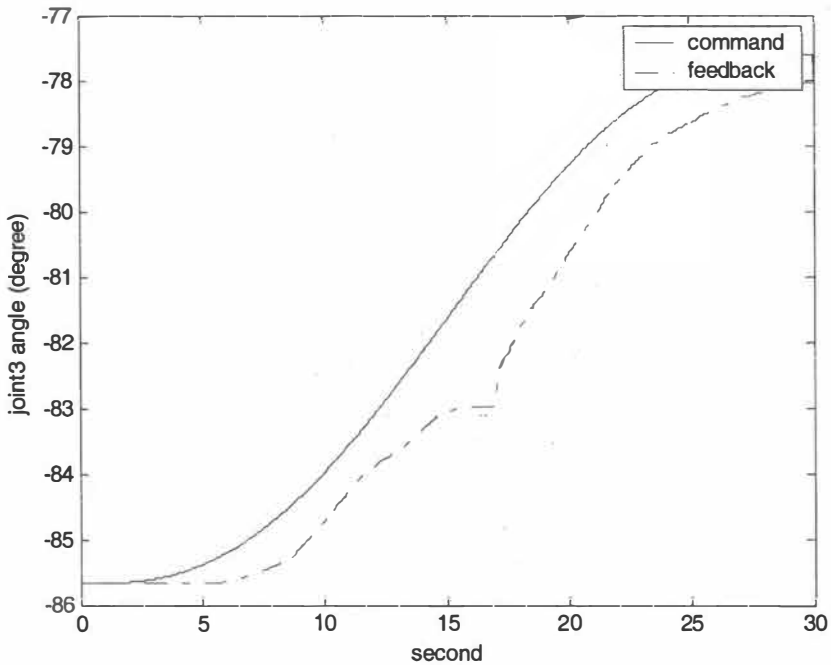


Figure 7.2.19 Joint 3 angles (30s).

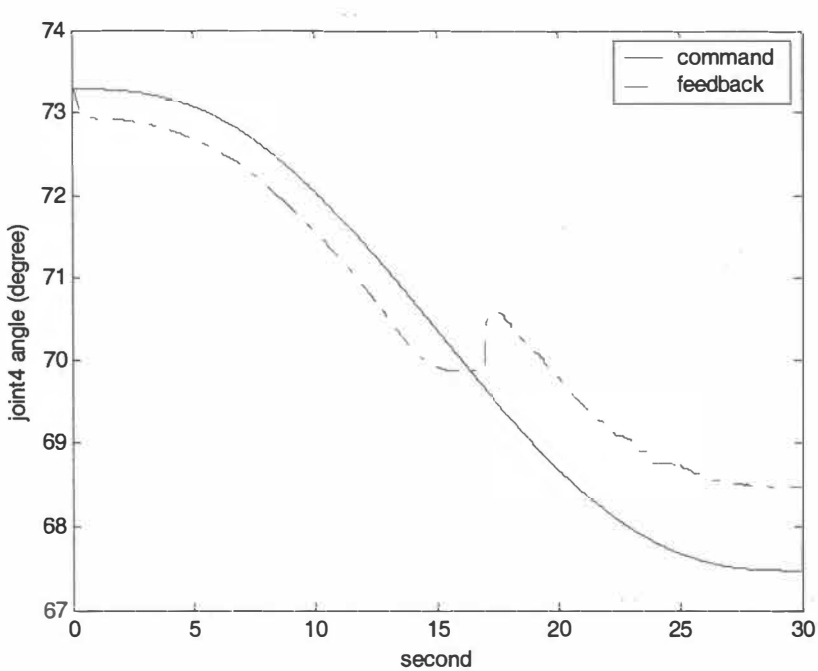


Figure 7.2.20 Joint 4 angles (30s).

7.3 Conclusions

Results in Sections 7.2.1 and 7.2.2 show that the newly developed grey prediction force/position parallel fuzzy control system possesses advantages over pure position control while the HMCTR system doing bandsaw cutting. The impact force in transition from free space to pipe cutting is greatly reduced (See Figures 7.2.1, 7.2.2, 7.2.6, and 7.2.7). The manipulator vibration with pure position control is eliminated by interaction force control (Figures 7.2.3 and 7.2.8). The HMCTR system remains stable even as the cutting speed is increased. The system performance and work quality is improved with the proposed controller.

Chapter 8

Conclusions and Future Work

8.1 Overall Conclusions

Teleoperation in nuclear facility maintenance and clean up, underwater and space operations often interacts with the task environment via tooling. A telerobotic system architecture has been developed considers the tooling interaction effect as an integral part of overall control system. A universal interaction force predictor based on grey system theory is built. The implementation of the grey model predictor is simple and robust. It requires less than 6 data sets to give a good prediction of interaction forces and torques. Its online prediction capability is one of the cores of the grey prediction force/position parallel fuzzy controller. This newly developed telerobotic control algorithm is tested by computer simulation using a two-link manipulator model detailed in Chapter 5. The simulation results show that the controller has good performance characteristics when significant interaction forces are present.

The overall telerobotic system architecture and the grey prediction force/position parallel fuzzy controller have also been implemented in human-machine cooperative telerobotic system at REMSL. Several pipe cutting experiments have been performed with and without interaction forces being considered. The experimental results show that the new controller reduces the impact forces and manipulator oscillations are essentially eliminated. For example, the peak cutting force along the saw blade is reduced from 54 to 47 lbf and more importantly, the peak impact force perpendicular to the saw bladed is reduced from 30 to 17 lbf. The overall system performance and work efficiency are

greatly improved from one successfully cutting out of five trials to eight successfully cuttings out of eight trials under the same conditions with interaction force control.

8.2 Contributions

The most significant contribution of this research to telerobotics is the incorporation tool interaction effects into overall control system. First of all, an overall human-machine cooperative telerobotic system architecture is proposed. The tool interaction component and other computer assistance functions can be easily added into the control system with the proposed modular designed control system architecture. Secondly, the use of the grey system theory to model the tool interaction forces has a degree of originality and an online universal interaction force prediction model is created based on the grey system theory. Thirdly, a grey prediction force/position parallel fuzzy controller is developed and implemented. This is the first attempt to integrate tool interactions into telerobotic system.

8.3 Future Work

It should be mentioned that the limitations imposed in this research, such as (i) offline force control gain tuning, (ii) subtask autonomous control, and (iii) lab scale teleoperation, must be investigated and improved for the truly generalized and applicable HMCTR system. The following sections outline the topics which future researchers are encouraged to investigate for improving and optimizing the performance of the HMCTR system with a grey prediction force/position parallel fuzzy controller.

Manual teleoperation with tooling interaction force control

Most of the remote operations in hazardous environments are completed manually by human operator using HMCTR system since the unstructured nature of the hazardous

task environments makes the total autonomous operation unavailable. Integrating tooling interaction force control associated with computer assistance functions such as velocity, linear, and plane assistance, are intended to reduce mentor stress and physical fatigue of the human operator and thus improve the work efficiency.

Online tuning of force control gain

The dynamic nature of task environment varies from task to task in real remote operations. The force control gain tuned for one task may not be suitable for another one and tuning the control gain by task is also tedious and time consuming. Fortunately, the present of soft computing technologies such as neural networks, fuzzy logic, and genetic method, makes online tuning possible.

Time delay

Teleoperation in space and through internet introduces a significant time delay because of the large distance between two operation sites, relatively low communication speed, and heavy internet traffic. The time delay affects the system performance. The effects of time delay on tooling interaction force control are unavoidable and can be a serious problem as time delay increases.

Other tooling interaction modeling method

In this study, only grey system theory has been investigated in detail for tooling interaction modeling. Other techniques such as neural networks, fuzzy logic, and genetic method etc., should be investigated in more detail.

Bibliography

[1] J. Vertut and P. Cioffet, Teleoperation and Robotics: Evolution and Development. Robot Technology, vol 3A, Hermes Publishing, 1985 pp 17-25.

[2] W. R. Hamel, Sensor-Based Planning and Control in Telerobotics, Control in Robotics and Automation: Sensor-Based Integration, Academic Press, 1999 pp 285-309.

[3] R. V. Dubey, M. A. Abidi, S. E. Everett, etc., Human-Machine Cooperative Telerobotics, Topical Report, Mechanical Engineering Department, University of Tennessee, Knoxville, 1998.

[4] W R. Hamel, G. Zhang, and P. Murray, A Real Time Controller for Human-Machine Cooperative Telerobotics, Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington D.C., May 2002. pp 2107-2114.

[5] S. E. Everett, Human-Machine Cooperative Telerobotics Using Uncertain Sensor and Model Data, Ph. D dissertation, University of Tennessee, Knoxville, 1998.

[6]. C. Guo, T. J. Tarn, N. Xi, and A.K. Bejczy, Fusion of human and machine intelligence for telerobotic systems. Proceedings of the 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan, May 1995, pp 3110-3115.

[7]. Y. Yokokohji, A. Ogawa, H. Hasunuma, and T. Yoshikawa. Operation modes for cooperating with autonomous functions in intelligent teleoperation systems. Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Atlanta, GA, May 1993, Vol 3 pp 510-515.

[8] H. Kazerooni, P. K. Houpt, and T. B. Sheridan, Robust compliant motion for manipulators, IEEE Journal of Robotics Automation, vol. RA-2, 1986, pp 83-105.

[9] J. K. Mills, Manipulator transition to and form contact task: a discontinuous control approach, Proceedings of IEEE International Conference on Robotics and Automation, Cincinnati, OH, May 1990, pp 440-446.

[10] M. C. Cavusoglu, F. Tendick, A Laparoscopic Telesurgical Workstation, IEEE Transactions on Robotics and Automation, Vol. 15, No. 4, August 1999, pp 728-739.

[11] R.H. Taylor, B.D. Mittelstadt, and H.A. Paul, A image-directed robotics system for precise orthopaedic surgery, IEEE Transactions on Robotics and Automation, Vol. 10, June 1994, pp 261-275.

[12] S. Lavallee, J. Troccaz, and L. Gaborit, Image guided operating robot: A clinical application in stereotactic neurosurgery, in Computer Integrated Surgery: Technology and Clinical Applications. Cambridge, MA: MIT Press, 1995.

[13] R. Tombropoulos, A. Schweikard, J.C. Latombe, and J.R. Adler, Treatment planning for image-guided robotic radiosurgery, in Computer Vision, Virtual Reality and Robotics in Medicine. First International Conference, CVRMed'95 Proceedings, Berling, Germany: Springer-Verlag, 1995, pp 131-137.

[14] P. Dario, E. Guglielmelli, B. Allotta, and M.C. Carrozza, Robotics for medical applications, IEEE Robotics and Automation Magazine, Vol. 3, No. 3, 1996, pp 44-56.

[15] http://www.newswire.ca/en/releases/archive/March2003/04/c6665.html.

[16] J. L. Nevins and D. E. Whitely, The force vector assembler concept, in Prepr. 1st CISM-IFToMM Symposium. Theory and Practice of Robots and Manipulators, Udine, Italy, Sept. 1973.

[17] Y. Wang and M. Mason, Modeling impact dynamics for robotic operations, Proceedings of IEEE International Conference on Robotics and Automation, Raleigh, NC, March, 1987, pp 678-685.

[18] D. E. Whitely, Force feedback control of manipulators fine motions, Transactions of ASME Journal of Dynamic Systems, Measurement and Control, vol. 99, June 1977, pp 91-97.

[19] K. Youcef-Toumi and D. A. Gutz, Impact and force control, Proceedings of IEEE International Conference on Robotics and Automation, Scottsdale, AZ, May 1989, pp 410-416.

[20] Y. F. Zheng and H. Hemami, Mathematical modeling of a robot collision with its environment, Journal of Robotic Systems, vol. 2. 1985, pp 289-307.

[21] J. De Schutter and H. Van Brussel, Compliant robot motion, Parts I-II, The International Journal of Robotics Res., vol. 7 no. 4, 1988, pp 3-33.

[22] R. P. Paul and B. Shimano, Compliance and control, Proceeding of 1976 Joint Automation Control Conference, West Lafayette, IN, July 1976, pp 695-699.

[23] J. K. Salisbury, Active stiffness control of a manipulator in Cartesian coordinates, Proceedings of 19th IEEE Conference on Decision Control, Albuquerque, NM, Dec. 1980, pp 95-100.

[24] S. Chiaverini and L. Sciavicco, Force/position control of manipulators in task space with dominance in force, Proceedings of 2nd IFAC Symposium on Robotic Control, Karlsruhe, Germany, Oct. 1988, pp 137-143.

[25] A. De Luca, C. Manes, and F. Nicolo, A task space decoupling approach to hybrid control of manipulators, Proceedings of 2nd IFAC Symposium on Robotic Control, Karlsruhe, Germany, Oct. 1988, pp 157-162.

[26] O. Khatib, A unified approach for motion and force control of robot manipulators: the operational space formulation, IEEE Journal of Robotics and Automation, vol. RA-3, 1987, pp 43-53.

[27] N. H. McClamroch and D. Wang, Feedback stabilization and tracking of constrained robots, IEEE Transactions of Automatic Control, vol. AC-33, 1988, pp 419-426.

[28] J. K. Mills and A. A. Goldenberg, Force and position control of manipulators during constrained motion tasks, IEEE Transactions of Robotics and Automation, vol. RA-5, 1989, pp 30-46.

[29] M. H. Raibert and J. J. Craig, Hybrid position/force control of manipulators, Transactions of ASME, Journal of Dynamic Systems, Measurement and Control, vol. 103, 1981, pp 126-133.

[30] J. T. Wen and S. Murphy, Stability analysis of position and force control for robot arms, IEEE Transactions of Automatic Control, vol. AC-36, 1991, pp36-371.

[31] H. West and H. Asada, A method for the design of hybrid position/force controllers for manipulators constrained by contact with the environment, Proceedings of IEEE International Conference on Robotics and Automation, St. Louis, MO, March, 1985, pp 251-259.

[32] T. Yoshikawa, Dynamic hybrid position/force control of robot manipulators-Description of hand constraints and calculation of joint driving force, IEEE Journal of Robotic Automation, vol. RA-3, 1987, pp 386-396.

[33] J. De Schutter and H. Van Brussel, A methodology for specifying and controlling compliant robot motion, Proceedings of 25th IEEE Conference on Decision and Control, Athens, Greece, December, 1986, pp 1871-1876.

[34] R. P. Paul and B. Shimano, Compliance and Control, Proceedings of JACC, 1976, pp 694-699.

[35] J. K. Salisbury and J. J. Craig, Articulated Hands: Force control and kinematic issues, International Journal of Robotics Research, vol. 1, no. 1 1982, pp 4-17.

[36] P. C. Watson, A multidimensional system analysis of the assembly process as performed by a manipulator, 1st North America Robot Conference, Chicago, 1976

[37] S. K. Drake and S. N. Simunovic, The use of compliance in a robot assembly system, Preprints, IFAC Symposium on Information and Control Problems in Manufacture Tech., Tokyo, 1977.

[38] N. Hogan, Control of mechanical impedance of prosthetic arms, Proceedings of 1980 JACC, San Francisco.

[39] N. Hogan, Impedance control: an approach to manipulation, Parts I-III, Transactions of ASME Journal of Dynamic Systems, Measurement and Control, vol. 107, 1985, pp 1-24.

[40] Anderson, Robert J. and Spong, Mark W. Hybrid Impedance Control of Robotic Manipulators, IEEE Proceedings of Robotics and Automation, 1987, pp 1073-1080

[41] Reza Vossougi and Max Donath, Using Impedance controlled Robots for Simulating Manipulation Tasks Occuring in a Gravityless Environment, Proceeding of the 1990 IEEE International Conference on Robotics & Automation, pp 62-67.

[42] M.Yamakita, M. Negi & K. Ito, Experimental Study of Tele-Bilateral Impedance Control Using Bilinear Model, Proceeding of the 1990 IEEE International Conference on Robotics & Automation, pp 634-640.

[43] L. J. Love and W. J. Book, Environment Estimation for Enhanced Impedance Control, Proceeding of the 1995 IEEE International Conference on Robotics and Automation, pp 1854-1859

[44] S. Arimoto, S. Kawamura and H. Y. Han, Impedance Matching for Evaluation of Dexterity in Execution of Robot Tasks, Proceeding of the 1998 IEEE International Conference on Robotics & Automation, pp 1435-1440.

[45] G. Morel, E. Malis and S. Boudet, Impedance Based Combination of Visual and Force Control, Proceeding of the 1998 IEEE International Conference on Robotics & Automation, pp 1743-1748.

[46] Ciro Natale, Bruno Siciliano & Luigi Villani, Spatial Impedance Control of Redundant Manipulators, Proceeding of the 1999 IEEE International Conference on Robotics & Automation, pp 1788-1793.

[47] A. Rubio, A. Avello, and J. Florez, Adaptive Impedance Modification of a Master–Slave Manipulator, Proceeding of the 1999 IEEE International Conference on Robotics & Automation, pp 1794-1799.

[48] S. Jung, S. B. Yim and T. C. Hsia, Experimental Studies of Neural Network Impedance Force Control for Robot Manipulators, Proceedings of the 2001 IEEE International Conference on Robotics & Automation, Seoul, Korea, May 21-26, 2001, pp 3453-3458.

[49] B. Daachi, A. Benallegue and N. K. M'Sirdi, A Stable Adaptive Force Controller for a Hydaulic Actuator, Proceedings of the 2001 IEEE International Conference on Robotics & Automation, Seoul, Korea, May 21-26, 2001, pp 3465-2470

[50] B. Daachi, A. Benallegue and N. K. M'Sirdi, A Stable Adaptive Force Controller for a Hydaulic Actuator, Proceedings of the 2001 IEEE International Conference on Robotics & Automation, Seoul, Korea, May 21-26, 2001, pp 3465-2470.

[51] M. O. Efe and Okyaynak, Stabilizing and Robustifying the Error Backpropagation Method in Neurocontrol Applications, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000, pp 1882-1887.

[52] Kazuo Kiguchi, Keigo Watanabe, Kiyotaka Izumi and Toshi Fukuda, Application of Multiple Fuzzy-Neuro Force Controllers in an Unknown Environment using Genetic Algorithms, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000, pp 2106-2111.

[53] S. J. Go and M. C. Lee, Fuzzy-sliding Mode Control with the Self Tuning Fuzzy Inference Based on Genetic Algorithm, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000, pp 2124-2129.

[54] K. Koguchi and T. Fukuda, Fuzzy Selection of Fuzzy-Neuro Robot Force Controllers in an Unknown Environment, Proceedings of the 1999 IEEE International Conference on Robotics & Automation, Detroit, Michigan, May 1999, pp 1182-1187.

[55] S. Chiaverini and L. Sciavicco, Edge-following strategies using the parallel control formulation, Proceedings of 1st IEEE Conference on Control Applications, Dayton, OH, Sept. 1992, pp 31-36.

[56] S. Chiaverini and B. Siciliano, On the stability of a force/position control/scheme for robot manipulators, Proceedings of 3rd IFAC/IFIP/IMACS Symposium on Robot Control, Wien, Austria, Sept. 1991, pp 183-188.

[57] S. Chiaverini, B. Siciliano, and L. Villani, Force/position regulation of compliant robot manipulators, IEEE Transactions of Automatic Control, vol. AC-39, 1994.

[58] S. Chiaverini and L. Sciavicco, The parallel approach to force/position control of robotic manipulators, IEEE Transactions on Robotics and Automation, vol. 9, no. 4, Aug. 1993, pp 361-373.

[59] J. J. E. Slotine, Sliding controller design for non-linear systems, Inernational Journal of Control, 1984, pp 421-434.

[60] R. Y. Wu, T. J. Tarn, N. Xi, and A. Isidori, Sensor Referenced Impact Control in Robotics. Control in Robotics and Automation: Sensor-Based Integration, Academic Press, 1999. pp 217-242.

[61] R. M. Brach, Mechanical Impact Dynamics, John Wiley & Sons, 1991.

[62] F. L. Lewis, S. Jagannathan and A. Yesildirek, Neural Network Control of Robot manipulators and Nonlinear Systems, Taylor & Francis 1999, p 175.

[63] W. Chen, J. K. Mills, and D. Sun, A Fuzzy Compensator for Uncertainty of Industrial Robots, Proceedings of the 2001 IEEE International Conference on Robotics & Automation, Seoul, Korea, May 21-26, 2001 pp 2968-2973.

[64] K. Kiguchi, K. Watanabe, K. Izumi and T. Fukuda, Application of Multiple Fuzzy-Neuro Force Controllers in an Unknown Environment using Genetic Algorithms,

Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000, pp 2106-2111.

[65] V. Santibanez, R. Kelly and M. A. Llama, Fuzzy PD+ Control for Robot Manipulators, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000, pp 2112-2117.

[66] E. Gurkan, A.M. Erkmen and I. Erkmen, Intuitionistic, 2-way Adaptive Fuzzy Control, Proceedings of the 1999 IEEE International Conference on Robotics & Automation, Detroit, Michigan, May 1999, pp 2470-2475.

[67] F. Y. Hsu and L. C. Fu, A New Adaptive Fuzzy Hybrid Force/Position Control for Intelligent Robot Deburring, Proceedings of the 1999 IEEE International Conference on Robotics & Automation, Detroit, Michigan, May 1999, pp 2476-2481

[68] S. Jung, S. B. Yim and T. C. Hsia, Experimental Studies of Neural Network Impedance Force Control for Robot Manipulators, Proceedings of the 2001 IEEE International Conference on Robotics & Automation, Seoul, Korea, May 21-26, 2001, pp 3453-3458.

[69] S. Hu, M. H. Ang Jr., and H. Krishnan, Neural Net work Controller for Constrained Robot Manipulators, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000, pp 1906-1911.

[70] F. Sun, Z. Sun, K. Kim, Y. Zhu and W. Lu, Stable Neuro-Adaptive Control for Robot with Unknown Dynamics, Proceedings of the 1999 IEEE International Conference on Robotics & Automation, Detroit, Michigan, May 1999, pp 1806-1811.

[71] W. Yu, A. S. Poznyak and E. N. Sanchez, Neural Adaptive Control of Two-Link Manipulator with Sliding Mode Compensation, Proceedings of the 1999 IEEE International Conference on Robotics & Automation, Detroit, Michigan, May 1999, pp 3122-3127.

[72] S. Commuri and S. Jagannathan, Modular Controls Design for Robot Manipulators Using CMAC Neural Networks, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, April 1997, pp 1725-1730.

[73] K. K. Kumbla and M. Jamshidi, Neural Network Identification of Robot Dynamics Used for Neuro-Fuzzy Controller, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, April 1997, pp 1118-1123.

[74] A. M. S. Zalzala & A. S. Morris, Neural Networks for Robotic Control, Theory and Applications, Ellis Horwood 1996, pp 3.

[75] C. Andersson, M. T. Andersson, J. E. Stahl, Bandsawing. Part I: cutting force model including effects of positional errors, tool dynamics and wear, International Journal of Machine Tools & Manufacture 41, 2001, pp 227-236.

[76] C. Andersson, J. E. Stahl, H. Hellbergh, Bandsawing. Part II: detecting positional errors, tool dynamics and wear by cutting force measurement, International Journal of Machine Tools & Manufacture 41, 2001, pp 237-253.

[77] L. Le-Ngoc and H. McCallion, Self-induced vibration of bandsaw blades during cutting, Proc Instn Mech Engrs, vol. 213 part c, 1999, pp 371-380.

[78] E. Stone and A. Askari, Nonlinear models of chatter in drilling processes, Dynamical Systems, vol. 17, no. 1, 2002, pp 65-85.

[79] J. H., Chin, C.T., Hsieh and L. W. Lee, The shaft behavior of BTA deep hole drilling tool, Int. J. Mech. Sci., vol. 38, no. 5, 1996, pp 461-482.

[80] V. P. Astakhov and M. O. M. Osman, An analytical evaluation of the cutting forces in self-piloting drilling using the model of shear zone with parallel boundaries. Part I: theory, International Journal of Machine Tools & Manufacture vol. 36, no. 11, 1996, pp 1187-1200.

[81] J. A. Yang, V. Jaganatha, and R. Du, A new dynamic model for drilling and reaming processes, International Journal of Machine Tools & Manufacture 42, 2002, pp 299-311.

[82] J. Deng, Introduction to grey system theory, The Journal of Grey System, no. 1, 1989, pp 1-24.

[83] J. Deng, Grey forecasting control, in Grey Systems, Beijing, China Ocean Press, 1988.

[84] D. S. Yi and R. Yang, Grey predictor controller for DC speed control system, The Journal of Grey System, No. 2, 1989, pp 189-215.

[85] Chang Wongching, Chang Wan, and Liang Hsuanming, Grey prediction controller design, The Journal of Grey System, No. 2, 1998, pp 123-131.

[86] Shaoyan Xu, Analysis of GM(1,N) forecasting model and its applications, Grey Systems, Beijing, China Ocean press, 1988, pp 180-194.

[87] S-J Huang and J-S Lee, A robotic motion controller using a self-organizing fuzzy logic algorithm with grey prediction, Proc. Instn. Engrs, vol. 212 part I, 1998, pp 293-304.

[88] E. Freund, The structure of decoupled nonlinear systems, International Journal of Control, vol. 21, 1975, pp 443-450.

[89] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, Resolved acceleration control of mechanical manipulators, IEEE Transactions on Automatic Control, vol. AC-25, 1980, pp 468-474.

[90] L. A. Zadeh, Fuzzy sets, Information and Control, No. 8, 1965, pp 338-353.

[91] L. H. Tsouskalas and R. E. Uhrig, Fuzzy and Neural Approaches in Engineering, John Wiley & Sons, 1997.

[92] T. J. Procyk and E. H. Mamdani, A linguistic self-organizing process controller, Automatica 15, 1979, pp 15-30.

[93] R. Tanscheit and E. M. Scharf, Experiments with the use of a rule-based self-organizing controller for robotics applications, Fuzzy Sets and Systems 26, 1988, pp 195-214.

[94] B. A. M. Wakileh and K. F. Gill, Robot control using self-organizing fuzzy logic, Computer industry. 15(3), 1990, pp 175-186.

[95] S. Daley and K. F. Gill, Attitude control of a space craft using an extended self-organizing fuzzy logic control. Proceedings of IMechE 201, 1987, pp97-106.

[96] J. J. Song and S. Park, A fuzzy dynamics learning controller for chemical process control. Fuzzy sets and systems 54, 1993, pp 121-133.

[97] Zong-Mu Yeh, A performance approach to fuzzy control design for nonlinear systems, Fuzzy sets and systems 64, 1994, pp 339-352.

[98] K. J. Utkin, Variable structure systems with sliding modes: a survey, IEEE Transactions on automatic control, AC-22, 1977, pp 212-222.

[99] W. R. Hamel, S. E. Everett, S. Kim, G. Zhang, Robot Task Scene Analyzer: Technical Overview, REMSL, University of Tennessee, Knoxville, August, 2000.

Appendix

Appendix A:

Jacobian of Schilling TII Manpulator

Equation A-1 is the generation form of Jacobian for a six degree of freedom manipulator. The calculation of Jacobian is based on the direct kinematics relations. Vectors $\vec{z}_0$ - $\vec{z}_5$ are given in equations A-2 – A-7 and vectors $\vec{p}_0$ - $\vec{p}_5$ and $\vec{p}$ are given in equations A-8 – A-14. Figure A-1 shows the frame assignments on the Schilling TII manipulator and Table A-1 is the Denavit-Hartenberg parameters.

$$J=\begin{bmatrix} \vec{z}_0\times(\vec{p}-\vec{p}_0) & \vec{z}_1\times(\vec{p}-\vec{p}_1) & \vec{z}_2\times(\vec{p}-\vec{p}_2) & \vec{z}_3\times(\vec{p}-p_3) & \vec{z}_4\times(\vec{p}-\vec{p}_4) & \vec{z}_5\times(\vec{p}-\vec{p}_5) \\ \vec{z}_0 & \vec{z}_1 & \vec{z}_2 & \vec{z}_3 & \vec{z}_4 & \vec{z}_5 \end{bmatrix} \quad \text{(A-1)}$$

$$\vec{z}_0=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{(A-2)} \quad \vec{z}_1=\begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \quad \text{(A-3)} \quad \vec{z}_2=\begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \quad \text{(A-4)}$$

$$\vec{z}_3=\begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \quad \text{(A-5)} \quad \vec{z}_4=\begin{bmatrix} -\cos(\theta_1)\sin(\theta_2+\theta_3+\theta_4) \\ -\sin(\theta_1)\sin(\theta_2+\theta_3+\theta_4) \\ \cos(\theta_2+\theta_3+\theta_4) \end{bmatrix} \quad \text{(A-6)}$$

$$\vec{z}_5=\begin{bmatrix} \cos(\theta_1)\cos(\theta_2+\theta_3+\theta_4)\cos(\theta_5)-\sin(\theta_1)\sin(\theta_5) \\ \sin(\theta_1)\cos(\theta_2+\theta_3+\theta_4)\cos(\theta_5)+\cos(\theta_1)\sin(\theta_5) \\ \sin(\theta_2+\theta_3+\theta_4)\cos(\theta_5) \end{bmatrix} \quad \text{(A-7)}$$

$$\vec{p}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{(A-8)} \qquad \vec{p}_1 = \begin{bmatrix} a_1 \cos(\theta_1) \\ a_1 \sin(\theta_1) \\ d_1 \end{bmatrix} \quad \text{(A-9)}$$

$$\vec{p}_2 = \begin{bmatrix} a_2 \cos(\theta_1)\cos(\theta_2) + d_2 \cos(\theta_1)\sin(\theta_2) + a_1 \cos(\theta_1) \\ a_2 \sin(\theta_1)\cos(\theta_2) + d_2 \sin(\theta_1)\sin(\theta_2) + a_1 \sin(\theta_1) \\ a_2 \sin(\theta_2) - d_2 \cos(\theta_2) + d_1 \end{bmatrix} \quad \text{(A-10)}$$

$$\vec{p}_3 = \begin{bmatrix} a_3 \cos(\theta_1)\cos(\theta_2 + \theta_3) + a_2 \cos(\theta_1)\cos(\theta_2) + d_2 \cos(\theta_1)\sin(\theta_2) + a_1 \cos(\theta_1) \\ a_3 \sin(\theta_1)\cos(\theta_2 + \theta_3) + a_2 \sin(\theta_1)\cos(\theta_2) + d_2 \sin(\theta_1)\sin(\theta_2) + a_1 \sin(\theta_1) \\ a_3 \sin(\theta_2 + \theta_3) + a_2 \sin(\theta_2) - d_2 \cos(\theta_2) + d_1 \end{bmatrix} \quad \text{(A-11)}$$

$$\vec{p}_4 = \begin{bmatrix} a_4 \cos(\theta_1)\cos(\theta_2 + \theta_3 + \theta_4) + a_3 \cos(\theta_1)\cos(\theta_2 + \theta_3) + a_2 \cos(\theta_1)\cos(\theta_2) \\ \qquad + d_2 \cos(\theta_1)\sin(\theta_2) + a_1 \cos(\theta_1) \\ a_4 \sin(\theta_1)\cos(\theta_2 + \theta_3 + \theta_3) + a_3 \sin(\theta_1)\cos(\theta_2 + \theta_3) + a_2 \sin(\theta_1)\cos(\theta_2) \\ \qquad + d_2 \sin(\theta_1)\sin(\theta_2) + a_1 \sin(\theta_1) \\ a_4 \sin(\theta_2 + \theta_3 + \theta_3) + a_3 \sin(\theta_2 + \theta_3) + a_2 \sin(\theta_2) - d_2 \cos(\theta_2) + d_1 \end{bmatrix} \quad \text{(A-12)}$$

$$\vec{p}_5 = \vec{p}_4 \qquad \text{(A-13)}$$

$$\vec{p} = \begin{bmatrix} d_6(\cos(\theta_1)\cos(\theta_2 + \theta_3 + \theta_4)\cos(\theta_5) - \sin(\theta_1)\sin(\theta_5)) + a_4 \cos(\theta_1)\cos(\theta_2 + \theta_3 + \theta_4) \\ + a_3 \cos(\theta_1)\cos(\theta_2 + \theta_3) + a_2 \cos(\theta_1)\cos(\theta_2) + d_2 \cos(\theta_1)\sin(\theta_2) + a_1 \cos(\theta_1) \\ d_6(\sin(\theta_1)\cos(\theta_2 + \theta_3 + \theta_4)\cos(\theta_5) + \cos(\theta_1)\sin(\theta_5)) + a_4 \sin(\theta_1)\cos(\theta_2 + \theta_3 + \theta_3) \\ + a_3 \sin(\theta_1)\cos(\theta_2 + \theta_3) + a_2 \sin(\theta_1)\cos(\theta_2) + d_2 \sin(\theta_1)\sin(\theta_2) + a_1 \sin(\theta_1) \\ d_6 \sin(\theta_2 + \theta_3 + \theta_4)\cos(\theta_5) + a_4 \sin(\theta_2 + \theta_3 + \theta_3) + a_3 \sin(\theta_2 + \theta_3) + a_2 \sin(\theta_2) \\ - d_2 \cos(\theta_2) + d_1 \end{bmatrix} \quad \text{(A-14)}$$

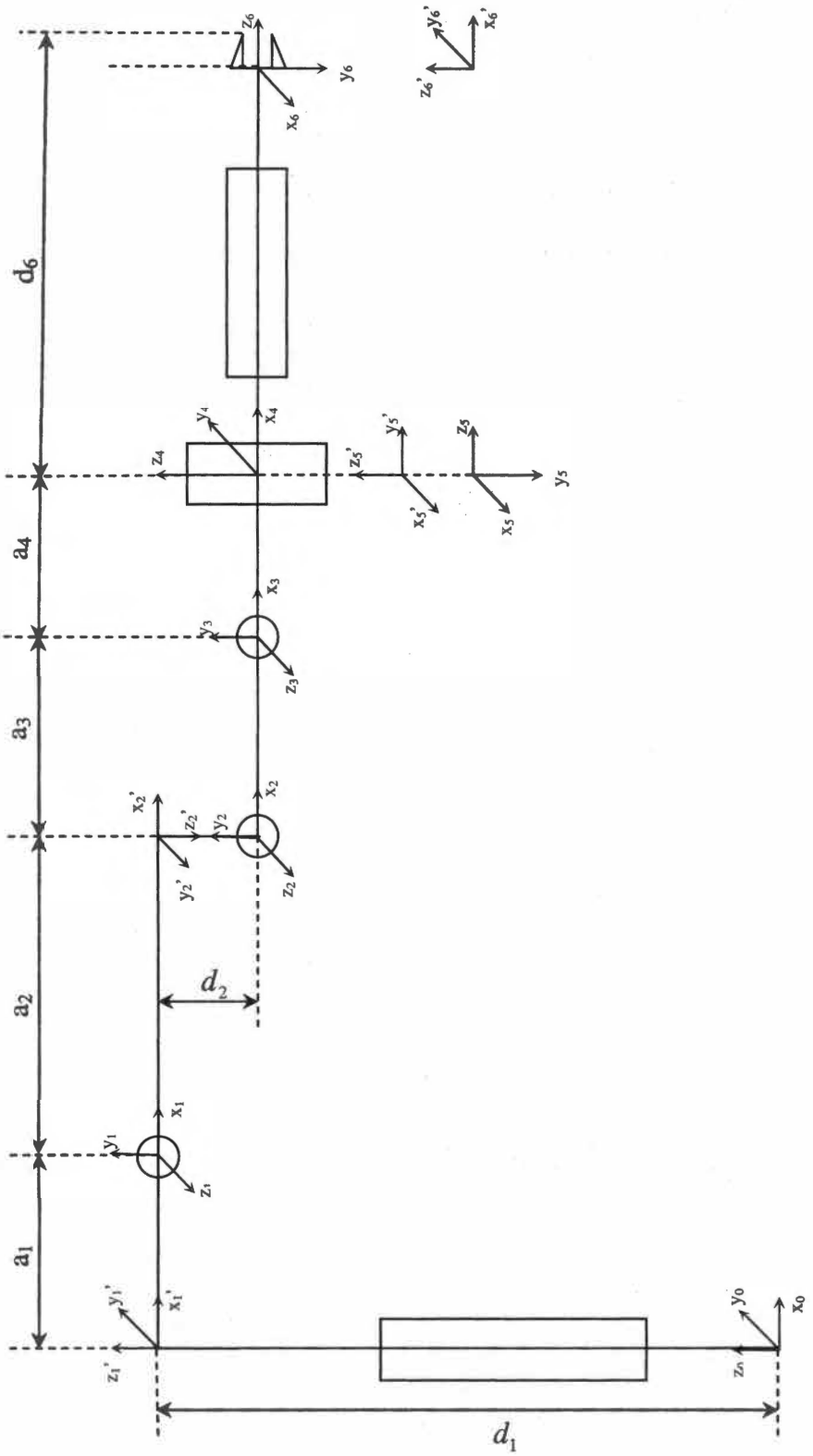


Figure A-1 Coordinate frame assignments for schilling TII manipulator

Table A-1 Denavit-Hartenberg parameters for schilling TII manipulator

	a_i (mm)	α_i (deg)	d_i (mm)	θ_i (deg)
1'	0	0	d_1	θ_1
1	a_1	$\frac{\pi}{2}$	0	0
2'	a_2	$\frac{\pi}{2}$	0	θ_2
2	0	$-\frac{\pi}{2}$	d_2	0
3	a_3	0	0	θ_3
4	a_4	$-\frac{\pi}{2}$	0	θ_4
5'	0	0	0	$-\frac{\pi}{2}$
5	0	$-\frac{\pi}{2}$	0	θ_5
6	0	0	d_6	θ_6

Appendix B

Results of Modeling tooling Interactions

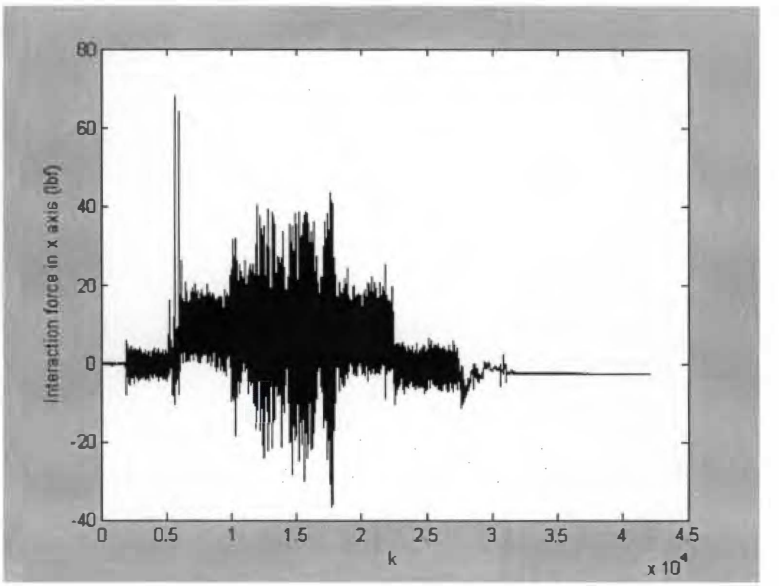


Figure B-1 Raw interaction force in x axis

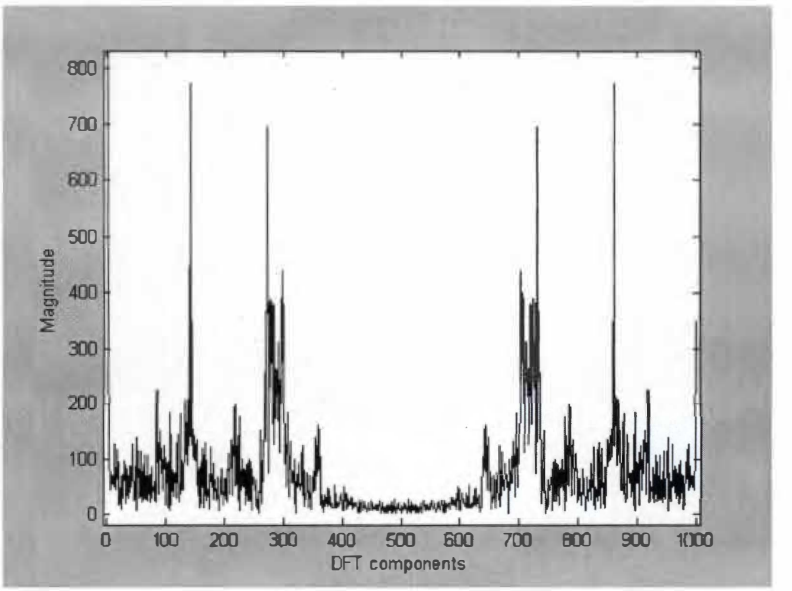


Figure B-2 FFT analysis on raw interaction force in x axis

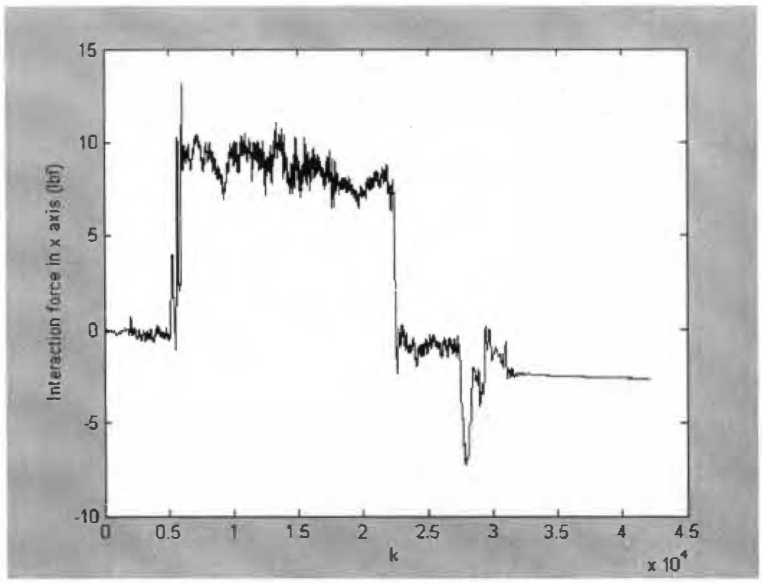


Figure B-3 Filtered interaction force in x axis

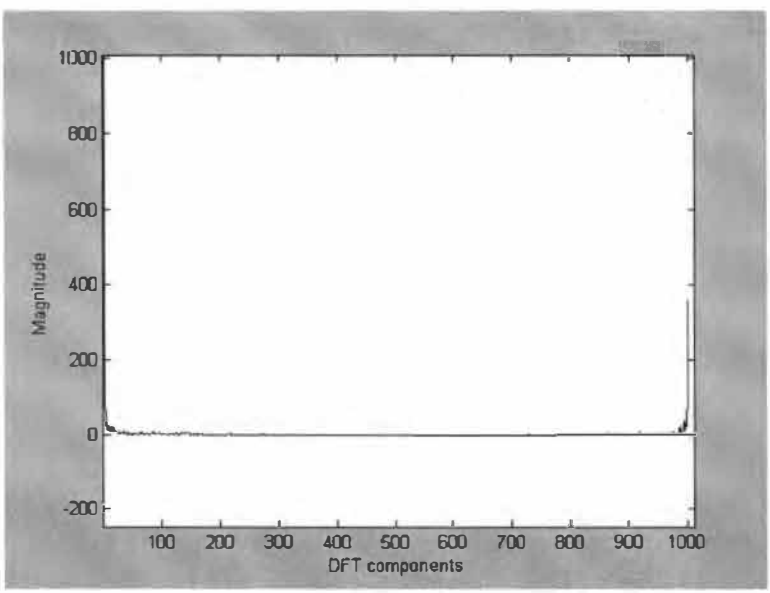


Figure B-4 FFT analysis on filtered interaction force in x axis

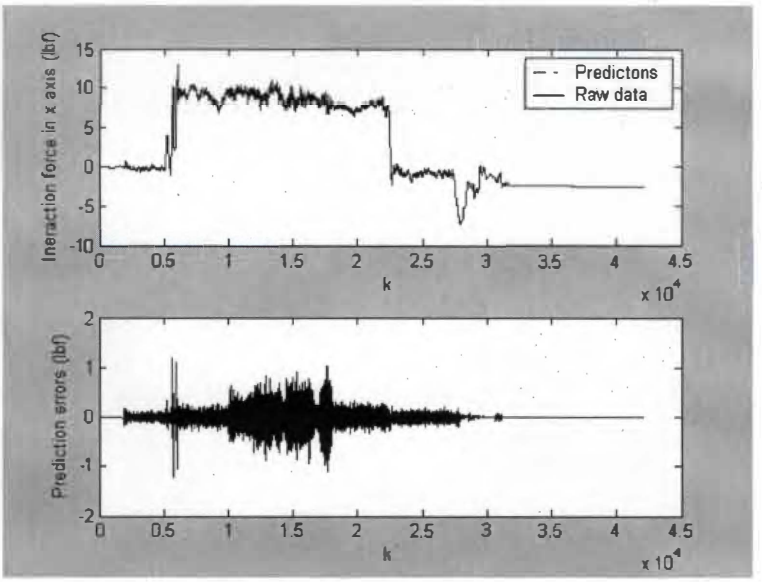


Figure B-5 GM(1,1) prediction results for force in x axis

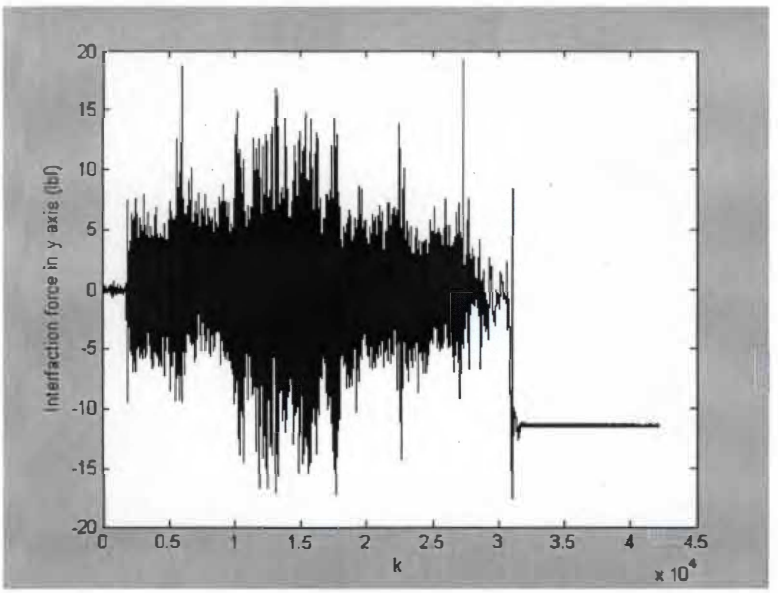


Figure B-6 Raw interaction force in y axis

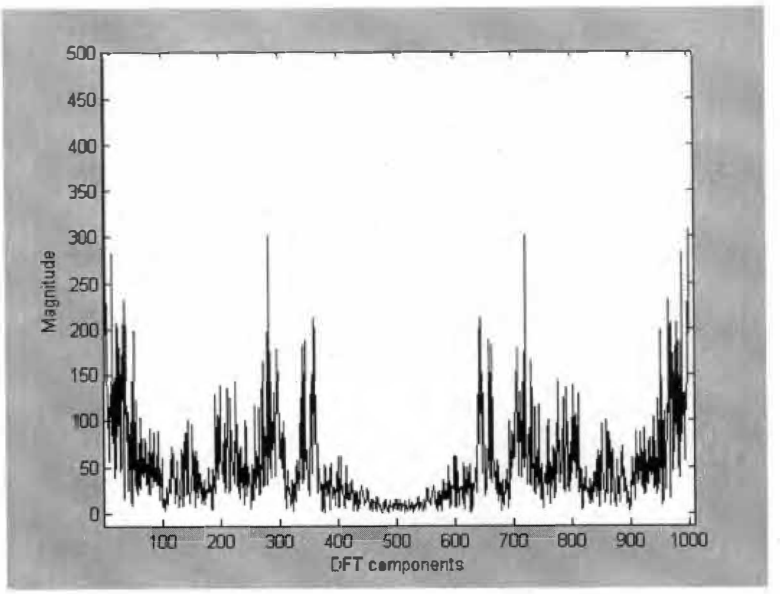


Figure B-7 FFT analysis on raw interaction force in y axis

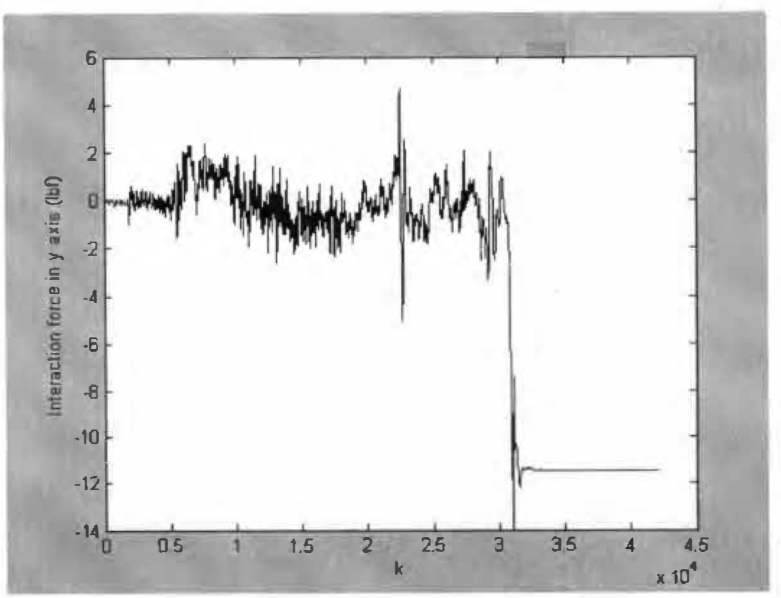


Figure B-8 Filtered interaction force in y axis

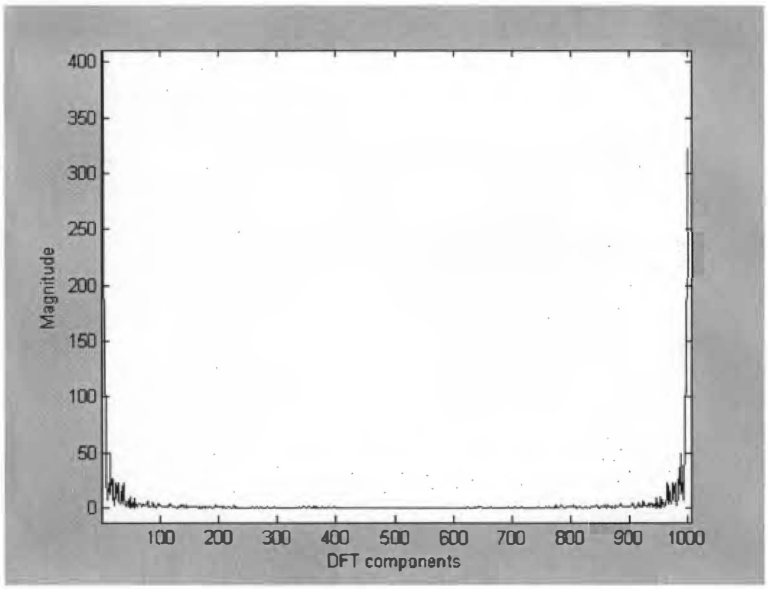


Figure B-9 FFT analysis on filtered interaction force in y axis

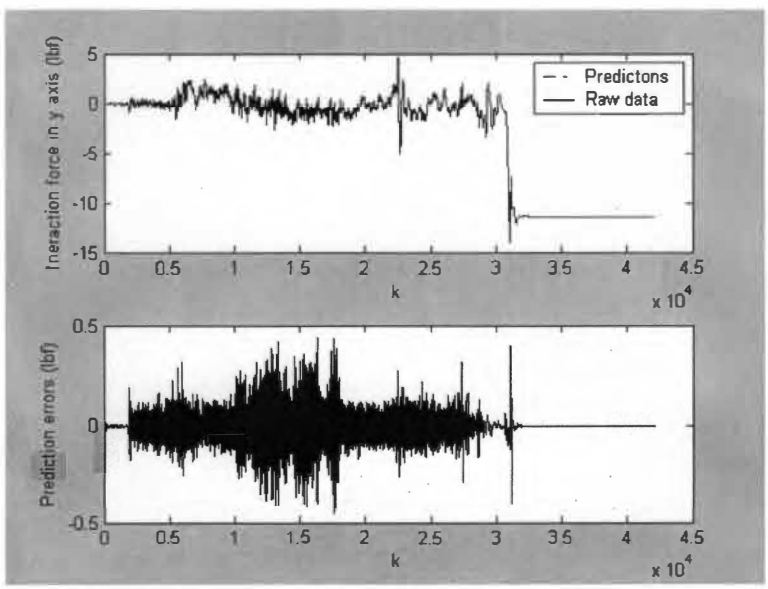


Figure B-10 GM(1,1) prediction results for force in y axis

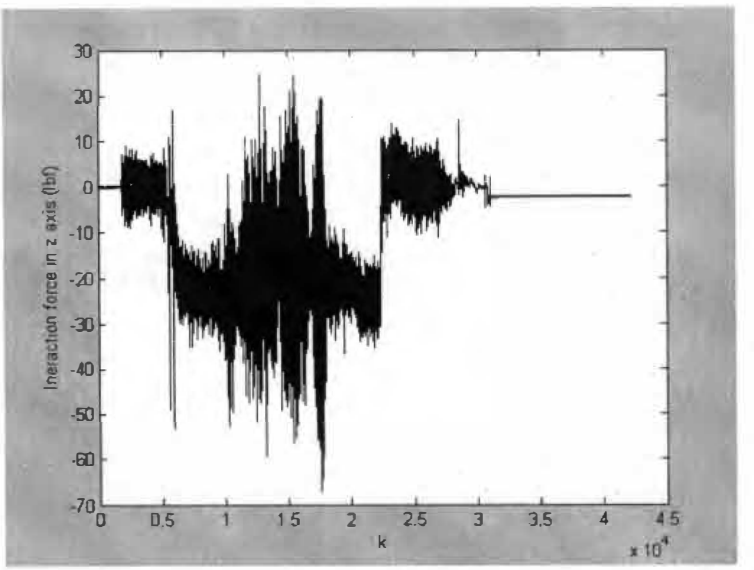


Figure B-11 Raw interaction force in z axis

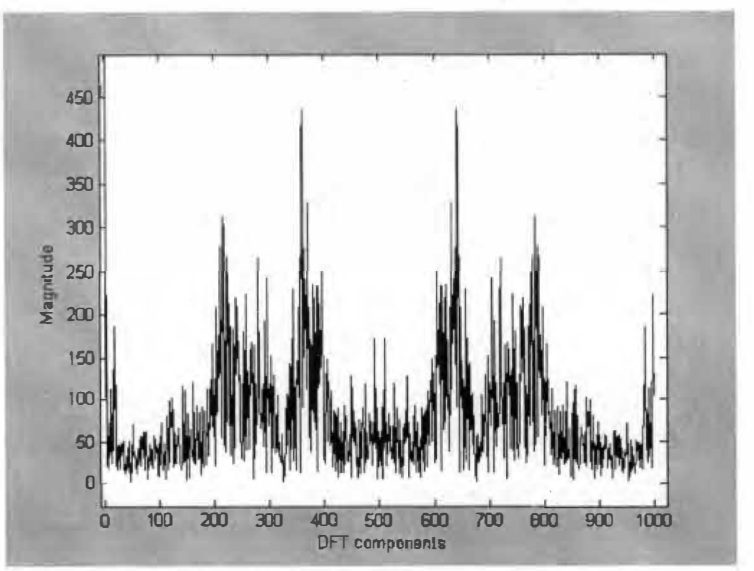


Figure B-12 FFT analysis on raw interaction force in z axis

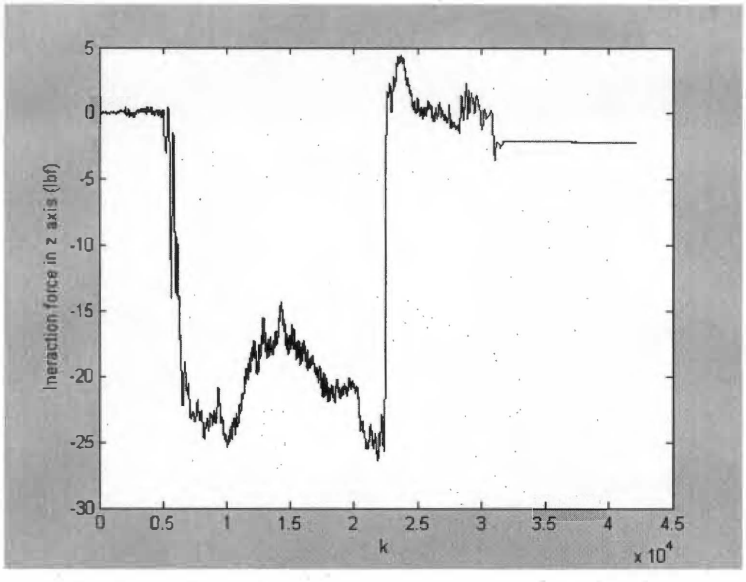


Figure B-13 Filtered interaction force in z axis

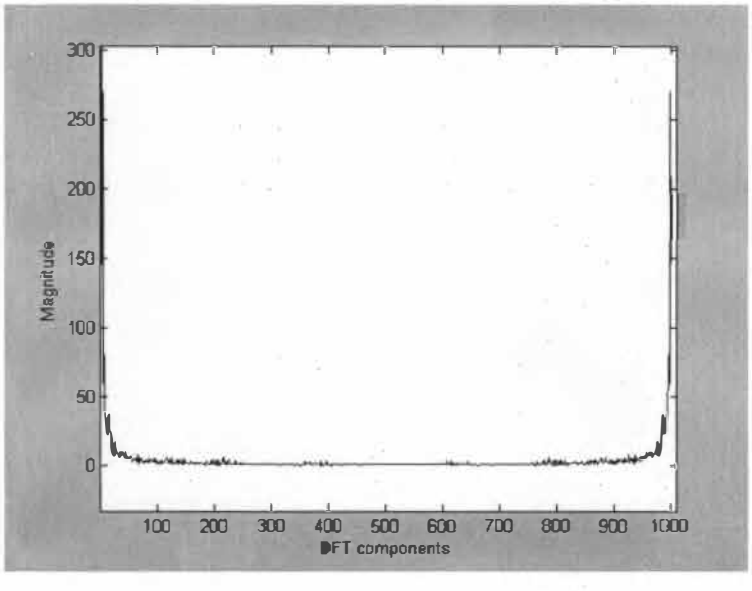


Figure B-14 FFT analysis on filtered interaction force in z axis

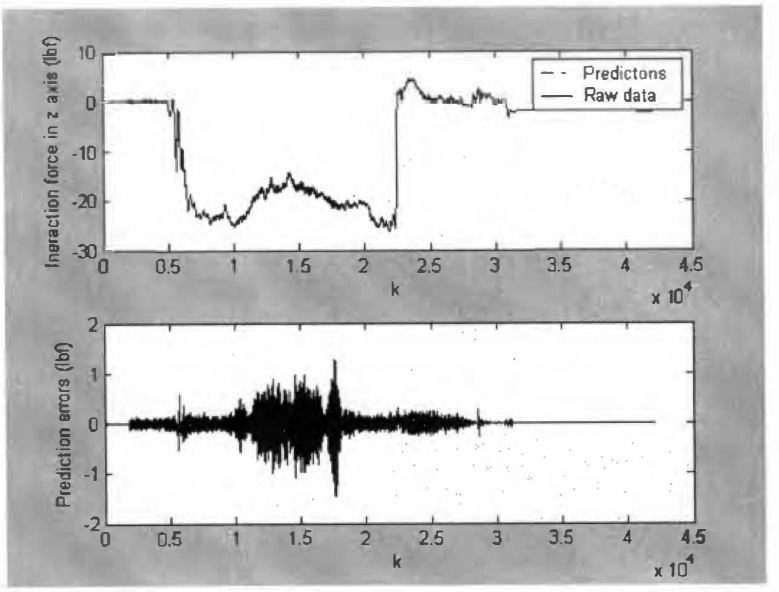


Figure B-15 GM(1,1) prediction results for force in z axis

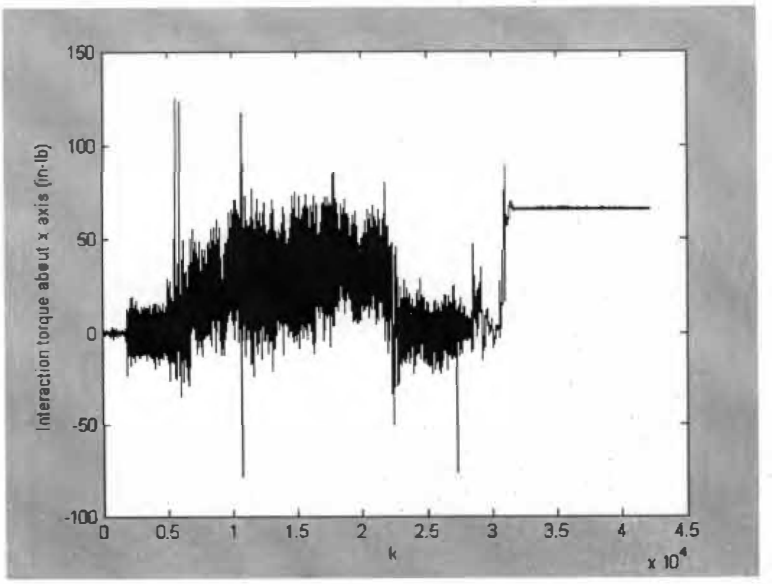


Figure B-16 Raw interaction torque about x axis

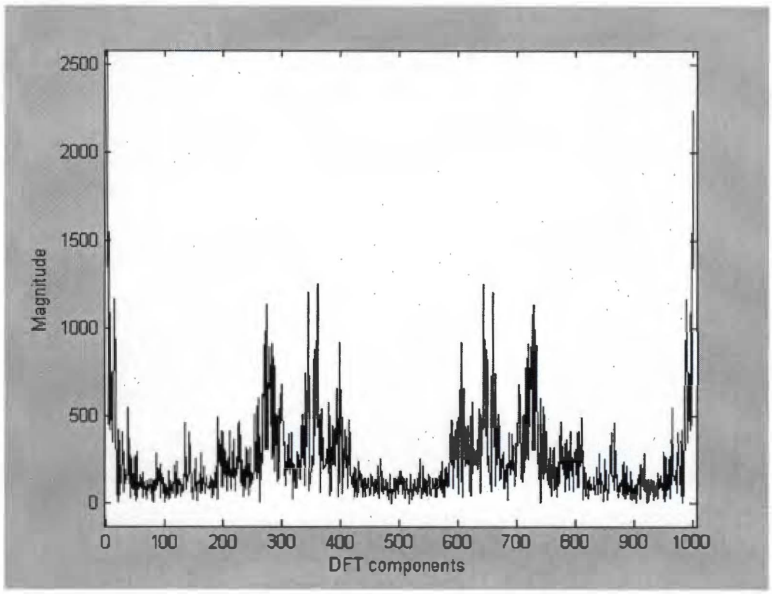


Figure B-17 FFT analysis on raw interaction torque about x axis

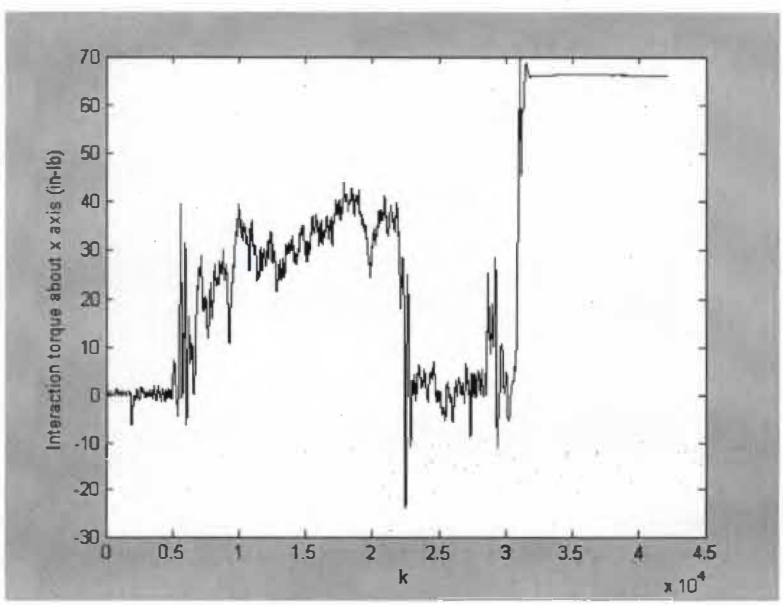


Figure B-18 Filtered interaction torque about x axis

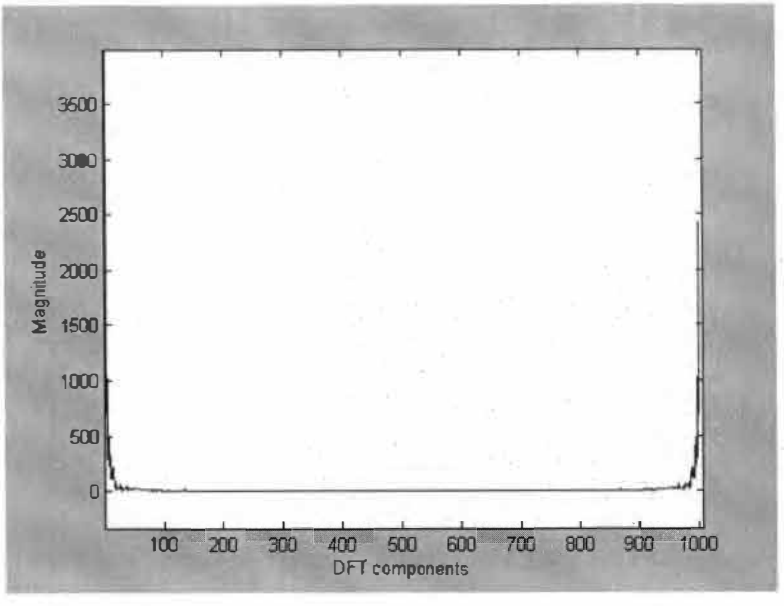


Figure B-19 FFT analysis on filtered interaction torque about x axis

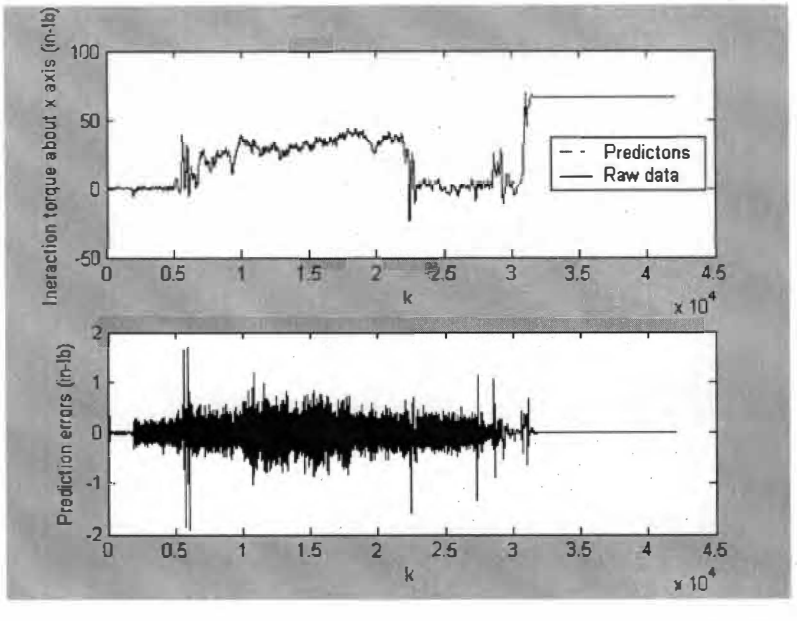


Figure B-20 GM(1,1) prediction results for torque about x axis

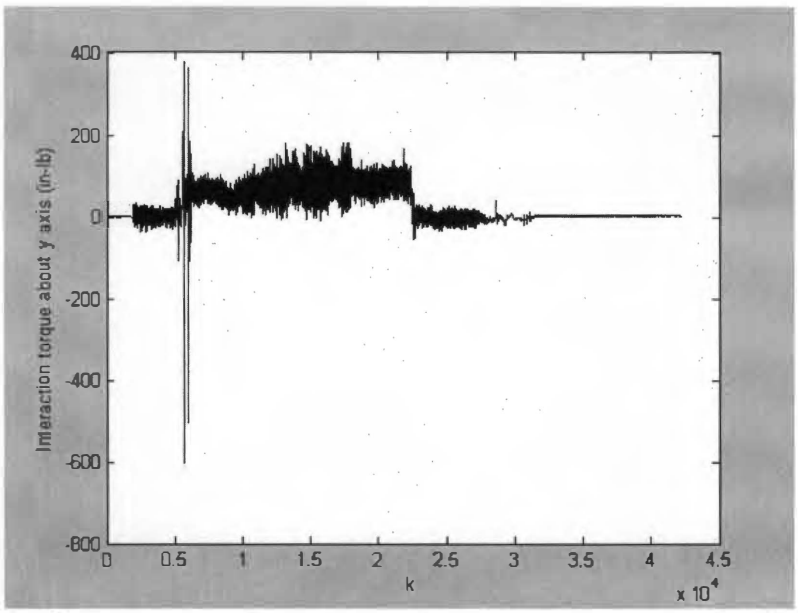


Figure B-21 Raw interaction torque about y axis

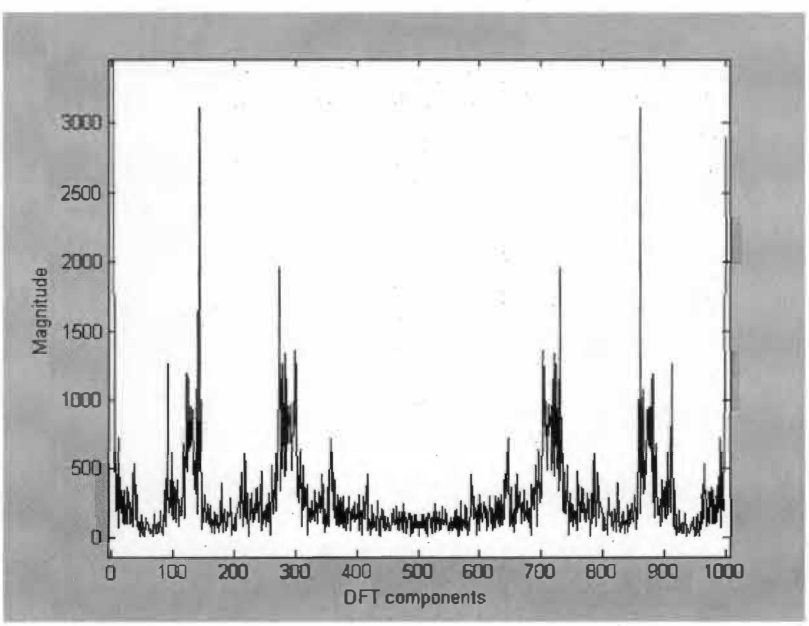


Figure B-22 FFT analysis on raw interaction torque about y axis

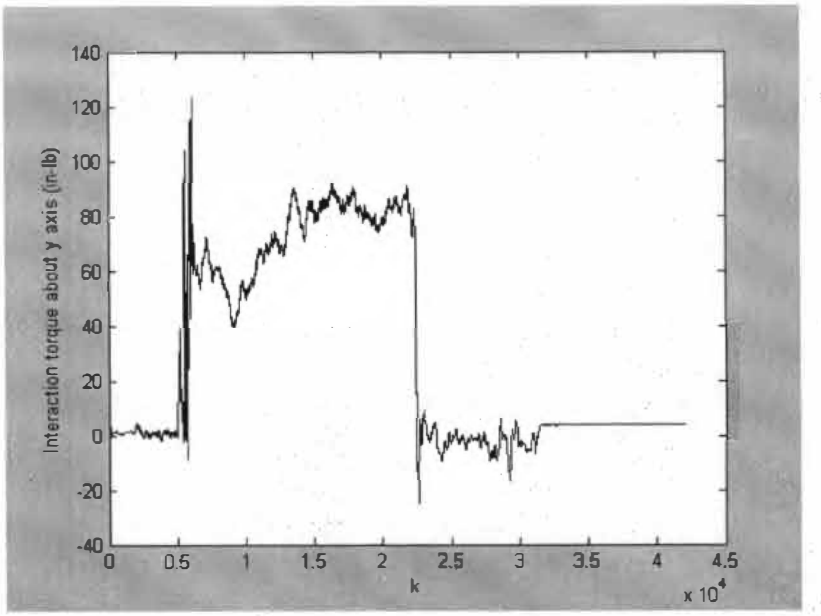


Figure B-23 Filtered interaction torque about y axis

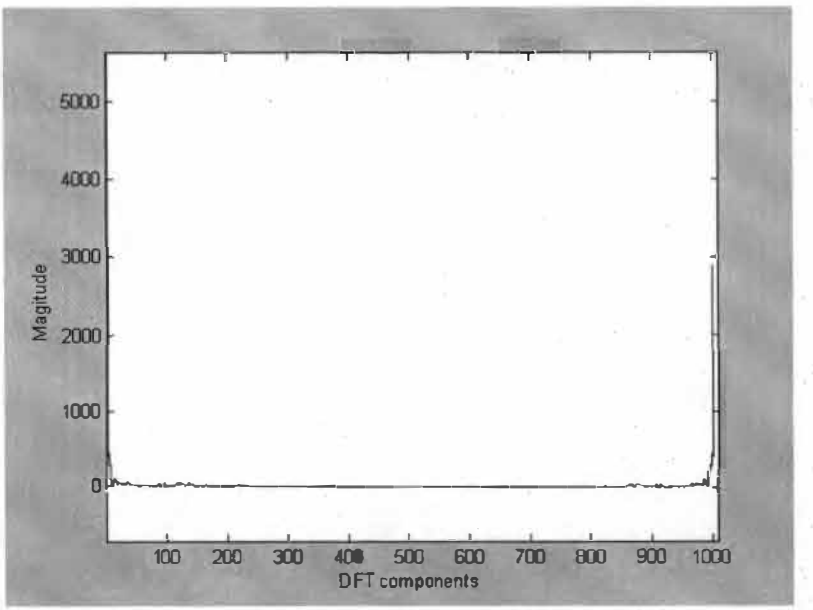


Figure B-24 FFT analysis on filtered interaction torque about y axis

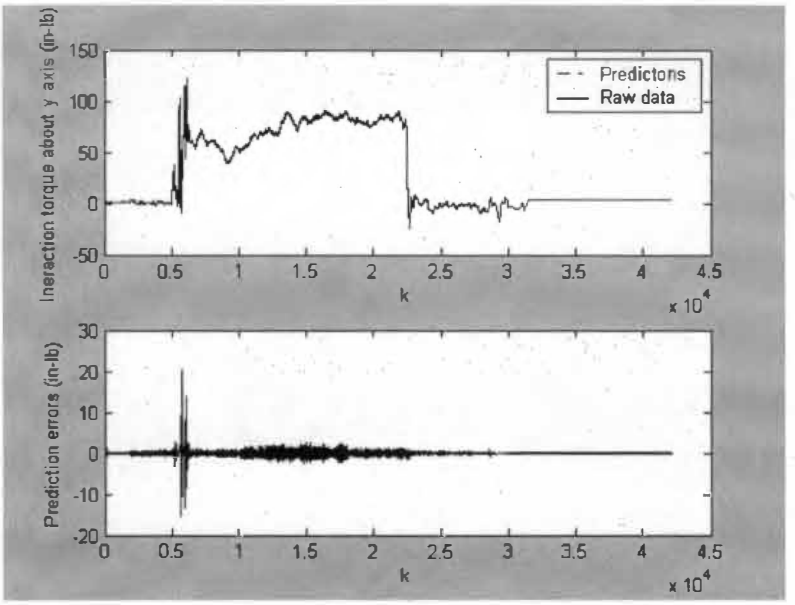


Figure B-25 GM(1,1) prediction results for torque about y axis

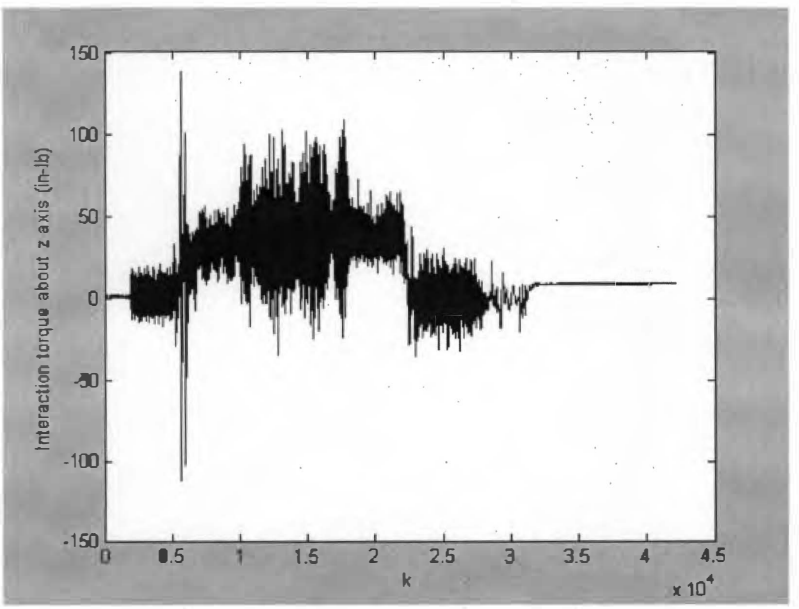


Figure B-26 Raw interaction torque about z axis

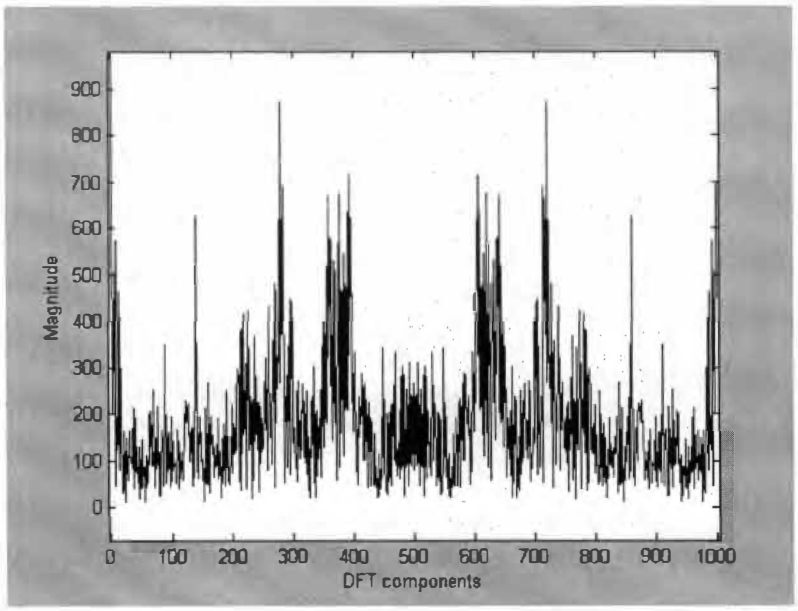


Figure B-27 FFT analysis on raw interaction torque about z axis

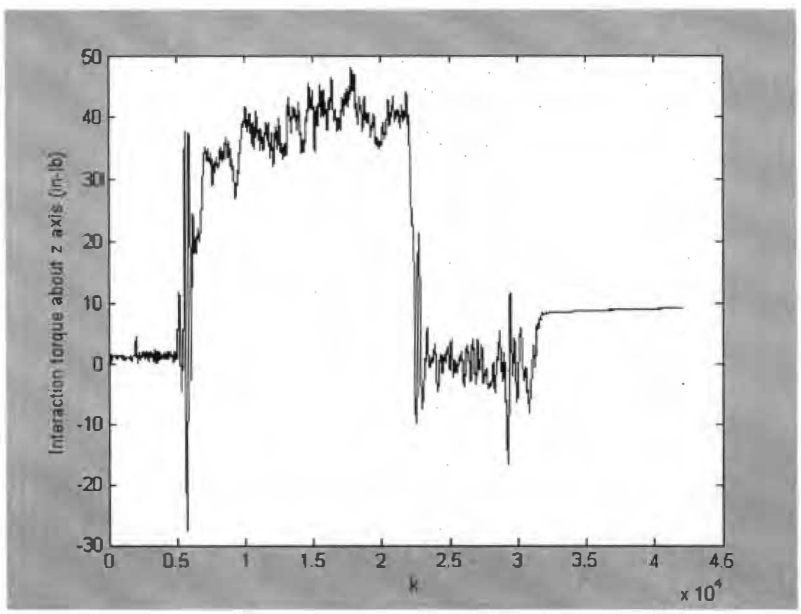


Figure B-28 Filtered interaction torque about z axis

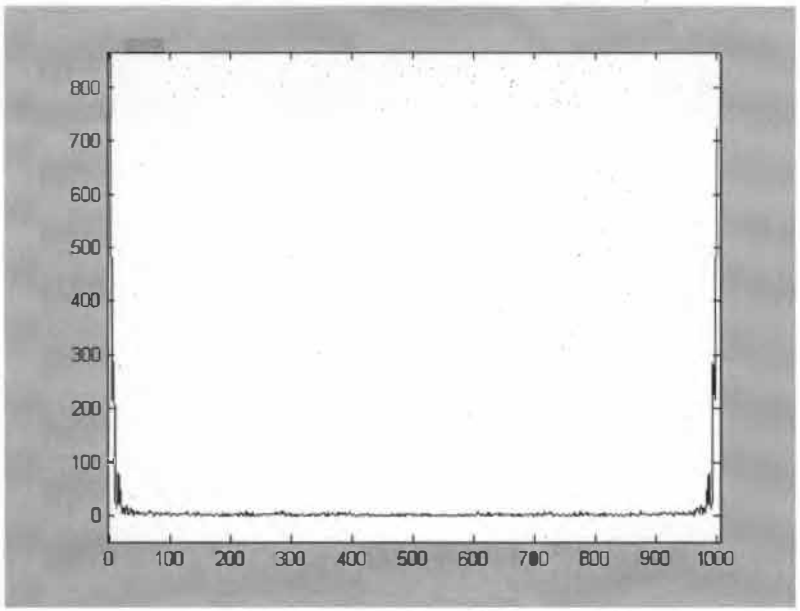


Figure B-29 FFT analysis on filtered interaction torque about z axis

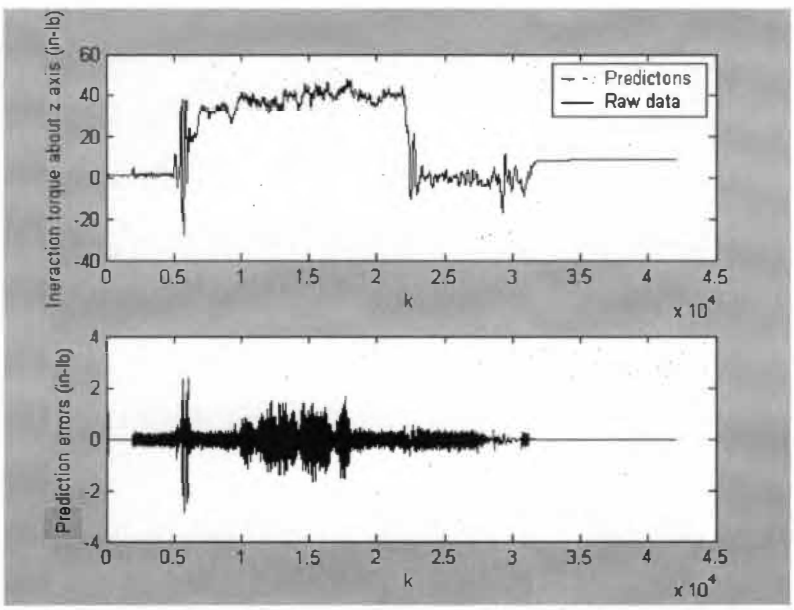


Figure B-30 GM(1,1) prediction results for torque about z axis

Appendix C

Matlab Codes for Two-Link Manipulator Simulation

```
% A two-link Robot Grey Prediction force/position Fuzzy controller
clear
deg = pi/180; % (a degree in radians)

% FUZZY SET DEFINITIONS

% Joint1 angle error
Joint1_e = [-4:0.001:4];
J1_mf_nm =trapmf(Joint1_e,[-4 -4 -1 -0.095]);
J1_mf_ns=trimf(Joint1_e,[-1 -0.095 0]);
J1_mf_ze= trimf(Joint1_e,[-0.095 0 0.095]);
J1_mf_ps=trimf(Joint1_e,[0 0.095 1]);
J1_mf_pm=trapmf(Joint1_e,[0.095 1 4 4]);
joint1_e=[J1_mf_nm;J1_mf_ns;J1_mf_ze;J1_mf_ps;J1_mf_pm];
plot(Joint1_e,joint1_e)
title('Universe of Discourse of Joint1 Angle Errors')
xlabel('Angle Errors in Rad')
ylabel('Grade')
pause

% Joint2 angle error
Joint2_e = [-4:0.001:4];
J2_mf_nm =trapmf(Joint1_e,[-4 -4 -1 -0.095]);
J2_mf_ns=trimf(Joint1_e,[-1 -0.095 0]);
J2_mf_ze= trimf(Joint1_e,[-0.095 0 0.1]);
J2_mf_ps=trimf(Joint1_e,[0 0.095 1]);
J2_mf_pm=trapmf(Joint1_e,[0.095 1 4 4]);
joint2_e=[J2_mf_nm;J2_mf_ns;J2_mf_ze;J2_mf_ps;J2_mf_pm];
plot(Joint2_e,joint2_e)
title('Universe of Discourse of Joint2 Angle Errors')
xlabel('Angle Errors in Rad')
ylabel('Grade')
pause

% Joint1 angle delta error
Joint1_de = [-5:0.001:5];
J1_dmf_ns=trapmf(Joint1_de,[-5 -5 -1 0]);
J1_dmf_ze= trimf(Joint1_de,[-0.2 0 0.2]);
J1_dmf_ps=trapmf(Joint1_de,[0 1 5 5]);
joint1_de=[J1_dmf_ns;J1_dmf_ze;J1_dmf_ps];
```

```
plot(Joint1_de,joint1_de)
title('Universe of Discourse of Joint1 Angle Velocity Errors')
xlabel('Angle Velocity Errors in Rad/Sec')
ylabel('Grade')
pause

% Joint2 angle delta error
Joint2_de = [-5:0.001:5];
J2_dmf_ns=trapmf(Joint1_de,[-5 -5 -1 0]);
J2_dmf_ze= trimf(Joint1_de,[-0.2 0 0.2]);
J2_dmf_ps=trapmf(Joint1_de,[0 1 5 5]);
joint2_de=[J2_dmf_ns;J2_dmf_ze;J2_dmf_ps];
plot(Joint2_de,joint2_de)
title('Universe of Discourse of Joint2 Angle Velocity Errors')
xlabel('Angle Velocity Errors in Rad/Sec')
ylabel('Grade')
pause

% Control torque for Joint 1
Joint1_t = [-2000:1:2000];
J1t_mf_nl =trapmf(Joint1_t,[-2000 -2000 -1000 -600]);
J1t_mf_nm=trimf(Joint1_t,[-1000 -600 -20]);
J1t_mf_ns=trimf(Joint1_t,[-600 -20 0]);
J1t_mf_ze= trimf(Joint1_t,[-20 0 20]);
J1t_mf_ps=trimf(Joint1_t,[0 20 600]);
J1t_mf_pm=trimf(Joint1_t,[20 600 1000]);
J1t_mf_pl=trapmf(Joint1_t,[600 1000 2000 2000]);
joint1_t=[J1t_mf_nl;J1t_mf_nm;J1t_mf_ns;J1t_mf_ze;J1t_mf_ps;J1t_mf_pm;J1t_mf_pl];

plot(Joint1_t,joint1_t)
title('Universe of Discourse of Joint1 Control Torques')
xlabel('Control Torques')
ylabel('Grade')
pause

% Control torque for Joint 2
Joint2_t = [-2000:1:2000];
J2t_mf_nl =trapmf(Joint2_t,[-2000 -2000 -1000 -600]);
J2t_mf_nm=trimf(Joint2_t,[-1000 -600 -20]);
J2t_mf_ns=trimf(Joint2_t,[-600 -20 0]);
J2t_mf_ze=trimf(Joint2_t,[-20 0 20]);
J2t_mf_ps=trimf(Joint2_t,[0 20 600]);
J2t_mf_pm=trimf(Joint2_t,[20 600 1000]);
J2t_mf_pl=trapmf(Joint2_t,[600 1000 2000 2000]);
joint2_t=[J2t_mf_nl;J2t_mf_nm;J2t_mf_ns;J2t_mf_ze;J2t_mf_ps;J2t_mf_pm;J2t_mf_pl];
```

```
plot(Joint2_t,joint2_t)
title('Universe of Discourse of Joint2 Control Torques')
xlabel('Control Torques')
ylabel('Grade')
pause

% Interaction force difference
Fxe = [-10:0.01:10];
Fxe_mf_nl =trapmf(Fxe,[-10 -10 -0.1 -0.05]);
Fxe_mf_nm=trimf(Fxe,[-0.1 -0.05 -0.01]);
Fxe_mf_ns=trimf(Fxe,[-0.05 -0.01 0]);
Fxe_mf_ze=trimf(Fxe,[-0.01 0 0.01]);
Fxe_mf_ps=trimf(Fxe,[0 0.01 0.05]);
Fxe_mf_pm=trimf(Fxe,[0.01 0.05 0.1]);
Fxe_mf_pl=trapmf(Fxe,[0.05 0.1 10 10]);
fxe=[Fxe_mf_nl;Fxe_mf_nm;Fxe_mf_ns;Fxe_mf_ze;Fxe_mf_ps;Fxe_mf_pm;Fxe_mf_pl
];
plot(Fxe,fxe)
title('Universe of Discourse of Difference of Interaction forces')
xlabel('Differece')
ylabel('Grade')
pause

% Coefficient of velocity
Vm = [-1:0.001:1];
Vm_mf_dl =trapmf(Vm,[-1 -1 -0.30 -0.15]);
Vm_mf_dm=trimf(Vm,[-0.30 -0.15 -0.10]);
Vm_mf_ds=trimf(Vm,[-0.15 -0.10 0]);
Vm_mf_ze=trimf(Vm,[-0.10 0 0.10]);
Vm_mf_is=trimf(Vm,[0 0.10 0.15]);
Vm_mf_im=trimf(Vm,[0.10 0.15 0.3]);
Vm_mf_il=trapmf(Vm,[0.15 0.30 1 1]);
VM=[Vm_mf_dl;Vm_mf_dm;Vm_mf_ds;Vm_mf_ze;Vm_mf_is;Vm_mf_im;Vm_mf_il];
plot(Vm,VM)
title('Universe of Discourse of Coefficient of velocity')
xlabel('Coefficient')
ylabel('Grade')
pause

% Rule Consequent Definition

consequent = [
joint1_t(1,:)
joint1_t(2,:)
joint1_t(3,:)
```

```
joint1_t(5,:)
joint1_t(6,:)
joint1_t(1,:)
joint1_t(2,:)
joint1_t(4,:)
joint1_t(6,:)
joint1_t(7,:)
joint1_t(2,:)
joint1_t(3,:)
joint1_t(5,:)
joint1_t(6,:)
joint1_t(7,:)
];

%rule for VM
consequent1 = [
VM(1,:)
VM(2,:)
VM(3,:)
VM(4,:)
VM(5,:)
VM(6,:)
VM(7,:)
];

disp('Computing...,Please wait!')

global t1 t2 te1 te2
ng=5;           %number of points to create grey model
t1=0;
t2=0;
te1=0;
te2=0;
namda = 30;
roll = 0.255;
eta = 0.01;     %learning rate
param;          %Setting all the parameters
t0=0;           %Initial Simulation time (SEC)
tf=9;           %Final Simulation Time (SEC)
kp(1)=2200;
kp(2)=2250;
kv(1)=0;
kv(2)=0;
ki(1)=200;
ki(2)=560;
```

```
kf1=0.005;
kf2=0.0064
x0=[0 0 0 0]';          %Initial Conditions
fd=zeros(6,1);
tdep=[0 0]';
tep=[0 0]';
[qd0]=invkin(0.5*a1,0,a1,a2);
x0(1)=qd0(1);
x0(2)=qd0(2);
xx=x0;
xxx(1,1)=0;
xxd(1,1)=0;
tt=0:T:tf;              %generating the discrete steps between t0 and tf
ll=length(tt-1);
qdp=[0 0]';
q=[x0(1) x0(2)]';
  t1=0;
  t2=0;
e=zeros(2,1)
IJ=zeros(2,1);
fx0=zeros(ng,1);
fy0=zeros(ng,1);
tz0=zeros(ng,1);
fxm0=zeros(ng,1);
fym0=zeros(ng,1);
tzm0=zeros(ng,1);
fp0=zeros(6,1);
vm=1;
Jf1=0;
Jf2=0;
for i=1:length(tt)-1
  [xd,fd]=trajxf(0,tt(i),a1,a2);
  [qd]=invkin(xd(1),xd(2),a1,a2);
  xxd(i,1)=xd(1);
  xxd(i,2)=xd(2);
  %desired interaction forces in joint spacce
  [Jd]=Ja(qd,a1/1000,a2/1000);
  tde=Jd'*fd;
  tde1=tde(1);
  tde2=tde(2);
  Tde1(i,1)=tde1;
  Tde1(i,2)=tde2;

  %feedback interaction forces in joint space
  if(i>1)
```

```
   [f_fdbk] = fdbk((xxx(i-1,1)),tt(i-1),kf1);
   fp0=f_fdbk;
   Fp(i,1)=f_fdbk(1);
   Fp(i,2)=f_fdbk(2);
   Fp(i,6)=f_fdbk(6);
else
   [f_fdbk] = fdbk((xxx(i,1)),tt(i),kf1);
   fp0=f_fdbk;
   Fp(i,1)=f_fdbk(1);
   Fp(i,2)=f_fdbk(2);
   Fp(i,6)=f_fdbk(6);
End

%gery model
 if (i>ng)

    %select previous ng data for prediction
    for j = 1:ng
       fx0(j)=F(i+j-ng-1,1);
       fy0(j)=F(i+j-ng-1,2);
       tz0(j)=F(i+j-ng-1,6);
    end
    %end of select

    %create biased data
    ma1=max(abs(fx0));
    ma2=max(abs(fy0));
    ma3=max(abs(tz0));
    for k= 1:ng
       fxm0(k) = fx0(k)+ma1+100;
       fym0(k) = fy0(k)+ma2+100;
       tzm0(k) = tz0(k)+ma3+100;
    end
    %end of creating biased data

    %grey predict
    [fx]=GM11(fxm0);
    [fy]=GM11(fym0);
    [tz]=GM11(fym0);
    %end of prediction

    %unbiased data
    fxx=fx-ma1-100;
    fyy=fy-ma2-100;
    tzz=tz-ma3-100;
```

```
    Fp(i,1)=fxx;
    Fp(i,2)=fyy;
    Fp(i,6)=tzz;
    fp0(1)=fxx;
    fp0(2)=fyy;
    fp0(6)=tzz;
 end
 %grey model end

  if (i>=400&&i<405)
    fp0(1)=300;
    fp0(2)=-fp0(1)/2;
 end
 if (abs(fd(1))<=0.001)
    FXE=0.0166*(fp0(1)-fd(1))/(abs(fp0(1))+0.0001);
 else
    FXE=0.0166*(fp0(1)-fd(1))/abs(fd(1));
 end

[vm]=rbeval3(Fxe,fxe,consequent1,FXE,Vm);

vmm(i)=vm;
[J]=Ja(q,a1/1000,a2/1000);
te=J'*f_fdbk;
tep=J'*fp0;
te1=te(1);
te2=te(2);
Te1(i,1)=te1;
Te1(i,2)=te2;

%calculte force errors and rate of change of force errors
Jf_error(1)=-tde1+tep(1);
Jf_error(2)=-tde2+tep(2);
Jf1=Jf1+Jf_error(1);
Jf2=Jf2+Jf_error(2);
ef1=0.00000218*Jf_error(1)+0.0000025*Jf1;
ef2=0.00000218*Jf_error(2)+0.0000025*Jf2;

%calculate position and velocity errors
qdd(i,1)=qd(1);
qdd(i,2)=qd(2);
if (i>1)
    qdp(1)=(qdd(i,1)-qdd(i-1,1))/T;
    qdp(2)=(qdd(i,2)-qdd(i-1,2))/T;
end
```

```
  J1_error(1) = (-qd(1)+xx(1))+ef1;
  J2_error(1) = (-qd(2)+xx(2))+ef2;

  J1_error(2) = (-qdp(1)+xx(3));
  J2_error(2) = (-qdp(2)+xx(4));
  %end of calculating position and velocity errors

  %calculate control torque based on fuzzy motion control
  [t1]=rbeval2(Joint1_e,Joint1_de,joint1_e,joint1_de,consequent,J1_error,Joint1_t);
  [t2]=rbeval2(Joint2_e,Joint2_de,joint2_e,joint2_de,consequent,J2_error,Joint2_t);

  clear t;
  clear y;
  TSPAN=[tt(i) tt(i+1)];
  options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-4 1e-4]);
  [t,y]=ode45('staspace',TSPAN,x0,options);

  x0=y(length(t),:)';
  xx=x0;

  k111(i)= t1;
  k112(i)= t2;
  q=[x0(1) x0(2)]';
  [xxx(i,1),xxx(i,2)]=kinem(q,a1,a2);
  yyy(i,:)=x0';
  Qd(i,:)=qd';
  Fd(i,:)=fd';
  F(i,:)=f_fdbk';
  if (i<=ng)
     Fp(i,:)=f_fdbk';
  end
  Tde(i,:)=tde';
  Te(i,:)=te';
end

for i=1:length(tt)-1
  ttt(i)=tt(i);
end
figure(1)
stairs(ttt,Qd(:,1),'b');
hold on
stairs(ttt,Qd(:,2),'r');
hold on
stairs(ttt,yyy(:,1),'b-.');
```

```
hold on
stairs(ttt,yyy(:,2),'r-.');
hold on
title('Joint Angles theta1(t) and theta2(t) in rad ');
xlabel('Time (Sec)');
ylabel('Angle (Rad)');
legend('qd1','qd2','Joint1','Joint2');
%axis([0 15 0 3]);
hold off
figure(2)
stairs(ttt,yyy(:,3),'b-');
hold on
stairs(ttt,yyy(:,4),'r-.');
title('Joint Angle Velocities in Rad/Sec ');
xlabel('Time (Sec)');
ylabel('Velocity (Rad/Sec)')
legend('Joint1','Joint2');
hold off

figure(3)
stairs(ttt,xxd(:,1),'b');
hold on
stairs(ttt,xxd(:,2),'r');
hold on
stairs(ttt,xxx(:,1),'b-.');
hold on
stairs(ttt,xxx(:,2),'r-.');
title('Trajectory in base frame');
xlabel('Time (Sec)');
ylabel('x/y m')
legend('x0','y0','x','y');
hold off


figure(4)
plot(ttt,Fd(:,1),'b');
hold on
plot(ttt,Fd(:,2),'r');
hold on
plot(ttt,F(:,1),'b-.');
hold on
title('Interaction forces in base frame');
xlabel('Time (Sec)');
ylabel('fd  N')
legend('fdx','fdy','fx','fy');
```

```
hold off
figure(5)
plot(ttt,Tde(:,1),'b');
hold on
plot(ttt,Tde(:,2),'r');
hold on
plot(ttt,Te(:,1),'b-.');
hold on
plot(ttt,Te(:,2),'r-.');
title('Interaction forces in base frame');
xlabel('Time (Sec)');
ylabel('fd  Nm')
legend('tde1','tde2','te1','te2');
hold off
figure(6)
plot(ttt,vmm,'r-');
title('Velocity coefficient');
xlabel('Time (Sec)');
ylabel('vm');
disp('--end of program--')

%invkin.m
%inverse kinematics for 2-link manipulator
function [q]=invkin(x,y,l1,l2)
alpha=atan2(y,x);
beta=acos((x^2+y^2+l1^2-l2^2)/(2*l1*sqrt(x^2+y^2)));
q1=alpha+beta;%for elbow-up
c2=(x^2+y^2-l1^2-l2^2)/(2*l1*l2);
q2=-acos(c2);%for q2=-pi:0
q=[q1 q2]';

%trajxf.m
%generate the position and force trajectories
function [xd,fd]=trajxf(vm,t,l1,l2)
xd=zeros(2,1);
xd(1)=50*t+0.5*l1+vm;%0.05*vm*t+0.5*l1;
xd(2)=0;
xdd=0.1;
ydd=0;
if (t<=2)
   fd=zeros(6,1);
elseif (t>2&&t<7)
   fd=zeros(6,1);
   fd(1)=100;
   fd(2)=-0.5*fd(1);
```

```
   fd(6)=0;%fd(2)*xd(1)/1000;
else
   fd=zeros(6,1);
end


%Ja.m
%Jacobian of 2-link manipulator

function [J]=Ja(q,l1,l2)
J=zeros(6,2);
J(1,1)=-l1*sin(q(1))-l2*sin(q(1)+q(2));
J(1,2)=-l2*sin(q(1)+q(2));

J(2,1)=l1*cos(q(1))+l2*cos(q(1)+q(2));
J(2,2)=l2*cos(q(1)+q(2));

J(6,1)=1;
J(6,2)=1;


%fdbk.m
%simulate the interaction forces
function [f_fdbk] = fdbk(xt,t,kf1)


f_fdbk=zeros(6,1);
if(t<=2)
   xe=600;
elseif (t>2&&t<=2.1)
   xe=50*t+500-5*(t-2);%0.05*t+0.5-0.005*(t-2);
else
   xe=50*t+500-0.5;%0.05*t+0.5-0.0005;
end
e=(xt-xe);
[t e]
if(e>0)
      f_fdbk(1)=(e)/kf1;
      f_fdbk(2)=-f_fdbk(1)/2;
      f_fdbk(6)=0;
   end
if (t>=7)
    f_fdbk=zeros(6,1);
end
```

```
%File GM11.m
%Grey Predition Model
%Function [y]=GM11(y0)
%y0---is raw data matrix nby1
%  --- n is the number of raw used to construct y1 n=4 or 5
%y--- output, prediction of k+1 step
%CopyRight Ge Zhang
%March 31,2003

function [y] = GM11(y0)
[n,m]=size(y0);
B=zeros(n-1,2);
yn=zeros(n-1,1);
y1=zeros(n,1);
phi=zeros(2,1);

%time step k=0:N but the index of vector in matlab beginning form 1.
% n the size of the input vector and N=n-1

for k=1:n
   for i = 1:k
      y1(k)=y1(k)+y0(i);
   end
end
B(:,2) = 1.0;
for i = 1:n-1
   B(i,1)= -0.5*(y1(i)+y1(i+1));
end

yn=y0(2:n);              % for time step k=1:N
phi=pinv(B)*yn;

c=phi(1);
u=phi(2);

%next step y0(k+1) = y

if (abs(c)<=0.000000001)
   if (c==0)
      c=0.000000001;
   end
   c = sign(c)*0.000000001;
end
y22=(y0(1)-u/c)*exp(-c*n)+u/c;
```

```
y11=(y0(1)-u/c)*exp(-c*(n-1))+u/c;
y=y22-y11;


%File rbebal2.m
function [torque]=rbeval2(Joint_e,Joint_de,joint_e,joint_de,consequent,J_error,Joint_t)
%
%       RBEVAL2 receives a Joint errors and delta errors and returns the control torque
%
%       J_error=[j_err j_derr]
%
%       torque = control torque
%

j_err=J_error(1);
j_derr=J_error(2);

% Fuzzification
DOF1=interp1(Joint_e',joint_e',j_err')';
DOF2=interp1(Joint_de',joint_de',j_derr')';

% Fuzzy Operator AND
antecedent_DOF = [
min(DOF1(5), DOF2(3))
min(DOF1(4), DOF2(3))
min(DOF1(3), DOF2(3))
min(DOF1(2), DOF2(3))
min(DOF1(1), DOF2(3))
min(DOF1(5), DOF2(2))
min(DOF1(4), DOF2(2))
min(DOF1(3), DOF2(2))
min(DOF1(2), DOF2(2))
min(DOF1(1), DOF2(2))
min(DOF1(5), DOF2(1))
min(DOF1(4), DOF2(1))
min(DOF1(3), DOF2(1))
min(DOF1(2), DOF2(1))
min(DOF1(1), DOF2(1))];

% Fuzzy Rule Consequent Definitions
% Defined in robot

% Implication Operation

Consequent = product(consequent,antecedent_DOF);
```

```
% Aggregate
aggregation = max(Consequent);

% Defuzzify
torque=centroid(Joint_t,aggregation);


%File rbeval3.m
function [vm]=rbeval3(Fxe,fxe,consequent,FXE,Vm)
%
%       RBEVAL3 receives interaction force errors and returns the velocity modification
%       coefficients
%       vm = velocity modification coefficients

% Fuzzification
DOF=interp1(Fxe',fxe',FXE')';

% Fuzzy Operator AND
antecedent_DOF = [
DOF(7)
DOF(6)
DOF(5)
DOF(4)
DOF(3)
DOF(2)
DOF(1)];

% Fuzzy Rule Consequent Definitions
% Defined in robot
% Implication Operation
Consequent = product(consequent,antecedent_DOF);

% Aggregate
aggregation = max(Consequent);


% Defuzzify
vm=centroid(Vm,aggregation);
```

Vita

Ge Zhang was born in Da Xian, Si Chuan Province, P. R. China, on May 25, 1968. He was raised in Chengdu, Si Chuan. He graduated from Chengdu No. 7 High School in 1986. From there he went to Northwestern Polytechnical University, Xi'an, P. R. China. He received a Bachelor of Science degree and a Master of Science degree in Aerospace Engineering from Northwestern Polytechnical University in 1990 and 1993, respectively. Following graduation, he worked as an aerospace engineer in Southwest Institute of Technical Physics for 5 years. He began his Ph.D. program in Mechanical Engineering at University of Tennessee, Knoxville in 1999 and he completed his Ph.D. degree in December 2004.

5897 7108 2
04/06/05 MAB