

11-12-2003

Development and Testing of a Haptic Interface to Assist and Improve the Manipulation Functions in Virtual Environments for Persons with Disabilities

Rohit Tammana
University of South Florida

Follow this and additional works at: <https://scholarcommons.usf.edu/etd>

 Part of the [American Studies Commons](#)

Scholar Commons Citation

Tammana, Rohit, "Development and Testing of a Haptic Interface to Assist and Improve the Manipulation Functions in Virtual Environments for Persons with Disabilities" (2003). *Graduate Theses and Dissertations*.

<https://scholarcommons.usf.edu/etd/1489>

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.

Development and Testing of a Haptic Interface to Assist and Improve the Manipulation
Functions in Virtual Environments for Persons with Disabilities

by

Rohit Tammana

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Mechanical Engineering
Department of Mechanical Engineering
College of Engineering
University of South Florida

Major Professor: Rajiv V. Dubey, Ph.D.
Autar K. Kaw, Ph.D.
Glen H. Besterfield, Ph.D.

Date of Approval:
November 12, 2003

Keywords: rehabilitation robotics, assistance functions, skill learning, phantom, hidden
markov model

© Copyright 2003 , Rohit Tammana

Acknowledgements

As I complete the transformation of my work into the thesis, I would like to take this opportunity to express my appreciation to some people who have not only helped me during the course of this research but during my last two years here.

My foremost thanks to Dr. Rajiv Dubey who was the major professor of this thesis, for giving me an opportunity to work with him, the freedom and the flexibility in my work, and most importantly for his immense patience during the course of my research. I would also like to acknowledge the financial support provided by him for the research with which I have been involved during the last year. I would also like to thank Dr. Autar K. Kaw and Dr. Glen H. Besterfield for being on my committee.

I would like to thank Wentao Yu for spending long hours with me, for his help since the earliest days of the lab and guiding me in my research work.

I would like to express my gratitude to Jim Farrell of Robotics Research Corporation, OH and to Billy Chan and Ryan Toohil from Sensable Technologies for their initial help with SolidWorks Simulation.

I would like to thank all the members at the Rehabilitation Engineering and Technology Program (RETP), USF for helping me out in conducting tests by persons with disabilities.

I would also like to thank all my friends for the enthusiasm and inspiration, which was always there when I needed it. I owe great thanks to my colleagues, at the rehabilitation robotics lab. My special thanks go to my friends Kiran, Deepa, Vivek and Santosh for their friendship was more motivating in my work than any technical assistance they have contributed. I would like to thank many others whose paths I crossed in the past few years I spent in Tampa. I refer to the friends Govardhan, Sridhar, Srinivas Meka and Ramesh Goutham for their ever-congenial nature.

Last but not the least, I am indebted to my parents, my brother-in-law and my sister who have been my inspirations and support for a lifetime, and without whom anything would have been impossible. They have been and still anxious to listen to my victories and defeats in life and constantly encouraging me with their pride in who I am and what I do.

And of course, Thank God. Any success I have had is a gift of God.

Dedication

I would like to dedicate this work to my family. Thank you for your prayers.

Table of Contents

List of Tables	iv
List of Figures	v
Abstract	viii
Chapter 1: Introduction	1
1.1 Research Motivation	1
1.2 Thesis Objectives	4
1.3 Thesis Outline	5
Chapter 2: Background	7
2.1 Technical Background	7
2.2 Previous Work	11
2.3 Teleoperation Overview	12
2.3.1 Teleoperation Research for Nuclear Clean-up	12
2.3.2 Rehabilitation Robotics Applications	14
Chapter 3: PHANTOM® Haptic Interface and GHOST® SDK	19
3.1 Introduction	19
3.2 Phantom Feature Overview	20
3.3 PHANTOM® Capabilities	24
3.3.1 Conceptual 3D Modeling and Design	24
3.3.2 Product Development and Manufacturing	25
3.3.3 Computer Animation	26
3.3.4 Telerobotics	27
3.3.5 Biomedical and Prosthetic Applications	28
3.3.6 Molecular and Nano-Manipulation	29
3.4 GHOST® SDK Overview	29
3.4.1 Adding Graphics with GHOST® GL	33
Chapter 4: SolidWorks Simulation of RRC Manipulator	36
4.1 Introduction	36
4.2 Robotics Research Corporation Seven DOF Manipulator	37
4.2.1 Controller Architecture	40
4.3 SolidWorks and Application Programming Interface	41
4.4 SolidWorks Simulation of RRC Manipulator	43

4.4.1 Dialog Simulation	44
4.4.2 Feedback Simulation	47
4.5 NetGhost Interface	48
4.6 RRG Kinematix	52
Chapter 5: Assistance Concept and Hidden Markov Model Development	55
5.1 Introduction	55
5.2 Assist Function Development for Office Environment	56
5.2.1 Position Assistance Function	59
5.2.1.1 Linear Assistance Function	59
5.2.1.2 Planar Assistance Function	60
5.2.1.3 Curve Trajectory Assistance	61
5.2.2 Velocity Assistance Function	62
5.2.3 Force Assistance Function	66
5.3 Hidden Markov Model Based Skill Learning and Motion Recognition	68
5.3.1 Data Preprocessing	68
5.3.2 Hidden Markov Model Construction	70
5.3.3 Skill Learning Approach and Tremor Suppression Filter	72
5.3.4 Methods of Evaluation	73
5.3.5 Tremor Filter Design and Matlab/Simulink Code	75
5.3.6 Motion Recognition and Design of Assistances	82
Chapter 6: Virtual Environments and Simulation Experiments	83
6.1 Introduction	83
6.2 Development of Virtual Office Environment	85
6.2.1 Task Definitions	89
6.2.2 Test Bed for Experiments	93
6.2.3 Haptic Interaction Simulation	94
6.2.4 Recording Techniques and Data Analysis	96
6.2.5 Therapy and Treatment	96
6.3 Hidden Markov Model Based Skill Learning Approach	97
6.3.1 Experimental Test Bed	100
6.3.2 Haptic Interaction Simulation	101
6.4 Hidden Markov Model Based Human Motion Recognition	103
6.4.1 Experimental Test Bed	104
6.4.2 Haptic Interaction Simulation	104
Chapter 7: Results and Discussion	106
7.1 SolidWorks Simulation of RRC Manipulator Using the Phantom	106
7.2 Tests Executed in Office Environment	106
7.2.1 Tests Executed by People With Disabilities	107
7.3 Results for the HMM Based Skill Learning Approach	120
7.3.1 Tests by Persons without Disabilities	121
7.3.2 Tests by Persons with Disabilities Before HMM Training	122
7.3.3 Tests by Persons with Disabilities After HMM Skill Learning	130

7.3.4 Discussion	138
7.4 Box and Blocks Test	139
Chapter 8: Conclusions and Recommendations	144
References	147

List of Tables

Table 7.1	Results Using Planar Assist Function for Object Position A	108
Table 7.2	Results Using Velocity Assist Function for Object Position B	111
Table 7.3	Results Using Velocity Assist Function for Object Position C	114
Table 7.4	Results Using Variable Scaling Function for Object Position D	116
Table 7.5	Results Using Curve Assist Function for Object Position E	118
Table 7.6	Numerical Results for the Labyrinth Before HMM Skill Learning	127
Table 7.7	Numerical Results for the Labyrinth After HMM Skill Learning	137
Table 7.8	Results for the Tremor Analysis before and after HMM Skill Learning	138

List of Figures

Figure 1.1	Percentages of Persons with Work Disability	2
Figure 1.2	Graph Showing People with Disabilities who would like to Work	2
Figure 2.1	RAID Work Station	15
Figure 2.2	MIME Work Station	16
Figure 3.1	Phantom Hand Controller Designs	23
Figure 3.2	Representation of the 3D Touch Technology System®	23
Figure 3.3	Sample program Structure for the Development of a Scene	24
Figure 3.4	Haptic Simulation Process	34
Figure 4.1	RRC Seven Degree-of-Freedom Manipulator	39
Figure 4.2	RRC R2 Control Architecture™	41
Figure 4.3	SolidWorks Model of RRC Manipulator	43
Figure 4.4	SolidWorks Simulation Screen Shot	44
Figure 4.5	Edit Boxes for Position Data	45
Figure 4.6	Edit Boxes for Holding Joint Angle Data	46
Figure 4.7	SolidWorks Screen Shot of the Phantom Simulation	47
Figure 4.8	NetGhost Interface for the SolidWorks Simulation	49
Figure 4.9	Block Diagram for the Code	51
Figure 5.1	Slave's Desired Direction of Motion	63
Figure 6.1	Virtual Office Environment	86

Figure 6.2	Virtual Office Environment with Teach Pendent	87
Figure 6.3	Interface for Virtual Environment	88
Figure 6.4	Maze Simulation Environment	101
Figure 6.5	Box and Blocks Simulation Environment	105
Figure 7.1	Graphical Analysis of Execution Times for Object Position A	108
Figure 7.2	Approach and Move Trajectories for Object Position A	109
Figure 7.3	Graphical Analysis of Execution Times for Object Position B	111
Figure 7.4	Approach and Move Trajectories for Object Position B	113
Figure 7.5	Graphical Analysis of Execution Times for Object Position C	114
Figure 7.6	Approach and Move Trajectories for Object Position C	115
Figure 7.7	Graphical Analysis of Execution Times for Object Position D	116
Figure 7.8	Approach and Move Trajectories for Object Position D	117
Figure 7.9	Graphical Analysis of Execution Times for Object Position E	118
Figure 7.10	Trajectories for Master and Slave for Object Position E	119
Figure 7.11	Trajectories for Master and Slave for Object Position B	120
Figure 7.12	Trajectories of Normal Persons for Skill Learning	121
Figure 7.13	Maze Test – Trajectories of Disabled Persons Before Training	122
Figure 7.14	Trajectories of Disabled Person With Tremor	123
Figure 7.15	Collision/Force Plots of a Disabled Person	125
Figure 7.16	Collision/Force Plots of a Disabled Person-2	126
Figure 7.17	Trajectories After Tremor Elimination	127
Figure 7.18	Tremor Plots Before Training	129
Figure 7.19	Trajectories of Disabled Persons After Training	131

Figure 7.20	Trajectories After HMM Training	132
Figure 7.21	Collision/Force Plots of a Disabled Person After Training	133
Figure 7.22	Trajectories of a Disabled Person After Training Without Tremor	135
Figure 7.23	Tremor Plots After HMM Training	136
Figure 7.24	Box and Blocks Test - No Assistance Provided	139
Figure 7.25	Box and Blocks Test – Velocity Components Without Assistance	140
Figure 7.26	Box and Blocks Test - Assistance Provided	141
Figure 7.27	Box and Blocks Test – Velocity Components With Assistance	141
Figure 7.28	Box and Blocks Test - Execution Times	143

**Development and Testing of a Haptic Interface to Assist and Improve the
Manipulation Functions in Virtual Environments for Persons with Disabilities**

Rohit Tammana

ABSTRACT

Robotics in rehabilitation provides considerable opportunities to improve the quality of life for persons with disabilities. Computerized and Virtual Environment (VE) training systems for persons with disabilities, many of which utilize the haptic feedback, have gained increasing acceptance in the recent years. Our methodology here is based on creating virtual environments connected to a haptic interface as an input device. This robotic setup introduces the advantages of the haptic rendering features in the environment and also provides tactile feedback to the patients.

This thesis aims to demonstrate the efficacy of assistance function algorithms in rehabilitation robotics in virtual environments. Assist functions are used to map limited human input to motions required to perform complex tasks. The purpose is to train individuals in task-oriented applications to insure that they can be incorporated into the workplace. Further, Hidden Markov Model (HMM) based motion recognition and skill learning are used for improving the skill levels of the users. For the Hidden Markov Model based motion recognition, the user's motion intention is combined with

environment information to apply an appropriate assistance function. We used this algorithm to perform a commonly used vocational therapy test referred to as the box and the blocks test. The Hidden Markov Model based skill approach can be used for learning human skill and transferring the skill to persons with disabilities. A relatively complex task of moving along a labyrinth is chosen as the task to be modeled by HMM. This kind of training allows a person with disability to learn the skill and improve it through practice. Its application to motion therapy system using a haptic interface helps in improving their motion control capabilities, tremor reduction and upper limb coordination. The results obtained from all the tests demonstrated that various forms of assistance provided reduced the execution times and increased the motion performance in chosen tasks. Two persons with disabilities volunteered to perform the above tasks and both of the disabled subjects expressed an interest and satisfaction with the philosophy behind these concepts.

Chapter 1: Introduction

1.1 Research Motivation

Physical disabilities make it difficult or impossible for individuals to perform several simple job related tasks such as pressing a button to operate a machine, opening a door, or moving a light object. While considering employment, the true potential can be enhanced by technology that augments the human performance. According to the Survey of Income and Program Participation (SIPP), 32.1 million working-age people (or 18.7% of the population age 15 to 64) have a disability. Also the data provided by the National Center for the Dissemination of Disability Research indicate that an ongoing problem among working-age Americans with disabilities is unemployment and/or underemployed. On December 17, 1999, Ticket to Work and Work Incentives Improvement Act (TWWIIA) was passed which focuses on increasing access to health care, as well as employment, training and rehabilitation services for individuals with disabilities.

In a large number of cases the mobility and manipulation functions of individuals with disabilities have been severely limited. Enhancing these functions would significantly increase the opportunities for millions of Americans with disabilities to be employed, educated, and participating citizens living in the community.

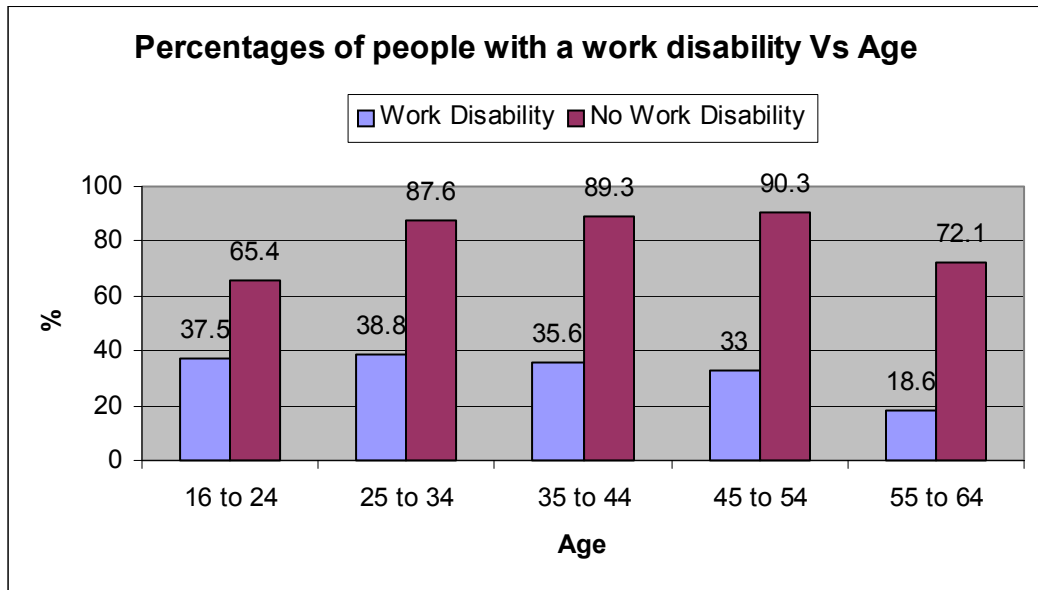


Figure 1.1 Percentages of Persons with Work Disability

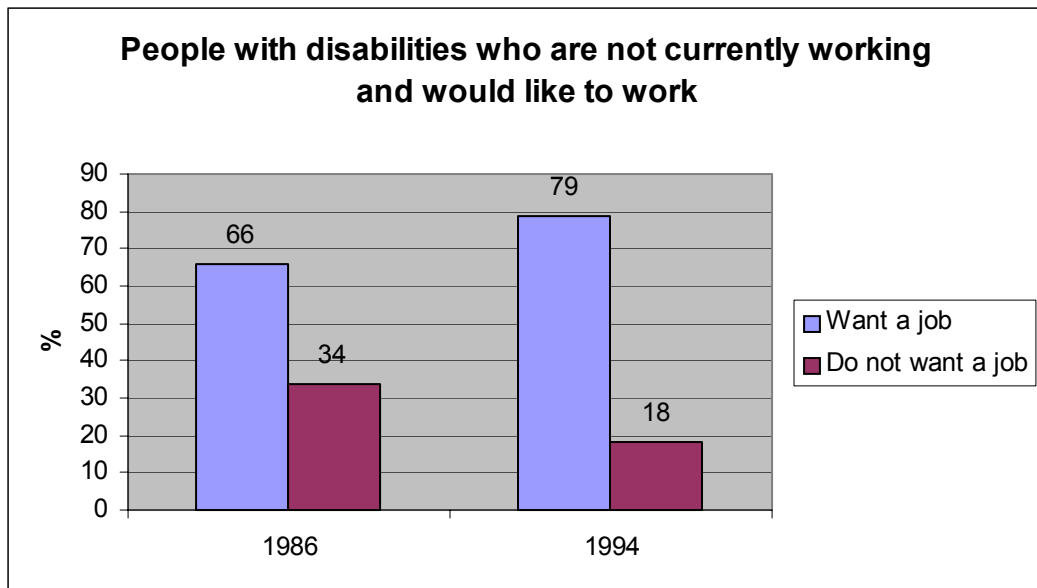


Figure 1.2 Graph Showing Persons with Disabilities who would like to Work

Robotics in rehabilitation deals with advancing robotics technology to provide persons with disabilities with tools to improve their quality of life and productivity for work. Thus Rehabilitation Technology is concerned with harnessing technology to benefit the everyday life of elderly and disabled persons. Research performed by R Erlandson [1] and others show the application of Robotics to Vocational Rehabilitation tasks. A tele-manipulator system enables interaction between a human operator and an environment without requiring direct physical contact between them. Such a system can provide human "presence" and decision-making capability in an environment that might otherwise be inaccessible to humans. Since tele-operators are separated from the human body, help can be provided more effectively for persons whose strength; freedom of motion and dexterity is limited in some fashion.

It is anticipated that with the developments in intelligent haptic interfaces, a greater number of severely disabled individuals can be employed and can enter the workforce. Persons with different kinds of disabilities can gain better access to graphical user interfaces. Haptic feedback, which is essentially force feedback in a man-machine interface, allows computer simulations of various tasks to relay realistic, tangible sensations to a user. With the incorporation of haptic feedback into virtual environments, users have the ability to push, pull, feel, and manipulate objects in virtual space rather than just seeing a representation on a video screen and the ability of individuals with disabilities to work in more challenging situations will increase significantly.

1.2 Thesis Objectives

The objective of this thesis is to incorporate various forms of assistance to the execution of complex job related tasks by individuals with disabilities in different environments. The object is to enhance inadequate human input, due to a limited range of motions and forces that an individual with disabilities might present, through the use of an intelligent haptic device to translate impaired motions and forces into more complex and extended ones. This system interfaces with a virtual environment to simulate the execution of certain job-related tasks. The purpose is to train individuals in different environments to insure they can be incorporated into the workplace. Based on the objective, following tasks are defined for the thesis:

- Develop different virtual environments to simulate the execution of complex job-related tasks.
- Test the assistance function concept previously used in teleoperation tasks off-line in virtual environment using GHOST SDK for executing a desired end-effector trajectory. The tasks chosen were based on a perceived need to increase the manipulation capability of the individual by reaching and handling objects via teleoperation in different environments.
- Use the Hidden Markov Model to recognize motions performed in virtual environment with a haptic device. This kind of assistance is useful not only for

path following but also for positioning the object and avoiding obstacles. This module is expected to relieve the operator's mental burden from doing the repeated tasks.

- Evaluate the manipulation functions of persons with disabilities. We use a sequence of motions to evaluate the skill by analyzing the performance of individual users over multiple repetitions of the dynamic task. This approach can be used for validating the skill level of specific users as well as the efficacy of various training methods. Virtual Environment has been created and used as a test-bed for a variety of skill learning applications.

1.3 Thesis Outline

The work starts with a history and background illustrating key terminology in the areas of telerobotics and various studies that were performed in the particular area of robotics applied to vocational applications, along with a brief overview of the project under which this work was performed, in Chapter 2. The details of the PHANTOM haptic device and the GHOST SDK software tool kit are discussed in Chapter 3. Chapter 4 describes the SolidWorks simulation of the RRC manipulator, which is used as an experimental test bed. Chapter 5 introduces the algorithms for the assist functions developed for the simulation and the Hidden Markov Model algorithm for skill learning in simulation environments. The development of the virtual environment scenario is presented in Chapter 6. Chapter 7 describes the simulation experiments along with the results. Finally,

conclusions are drawn and ideas for future work are presented in Chapter 8. Program listing, graphics model and source code for the computer programs are provided in the Appendices.

Chapter 2: Background

2.1 Technical Background

Intelligence in a rehabilitation robotic system is not usually stressed because of the presence of person in the control loop to make all the decisions. This expectation has resulted in the development of many rehabilitation robots that provide an interface to the user to issue robot motion commands by means of traditional teleoperation. Although teleoperation may reduce the physical burden of a manipulative task this is replaced with the mental burden of controlling the robot. Reducing the mental load on the user during the performance of manipulative tasks will help in increasing the acceptance of robotic aids for rehabilitation [2]. Just as early Industrial robots were seen as material handling devices, so too were the applications of robotics in rehabilitation. Rehabilitation is an activity, which aims to enable a disabled person to reach an optimum mental, physical and social functional level. Thus rehabilitation robotics deals with advancing robotics technology to provide physically disabled people with tools to improve their quality of life and productivity of work [3].

Examples within this application area include vocational tasks, such as manipulative tasks in an structured environment (paper handling in office based work etc.) and daily living activities in structured and non structured environments such as educational tasks, eating and personal hygiene [4]. This implies the use of robots in a way that is quite different from industrial applications where robots normally operate in a structured environment with predefined tasks, separated from human operators. This may not be the case in rehabilitation robotics. Thus rehabilitation robotics have more in common with service robots which integrate human and robots in same task, requiring certain safety aspects and special attention to man-machine interfaces for people with physical problems operating a specific programming device. Thus, more attention must be paid to the user requirements, as the user is a part of the process in the execution of various tasks.

A basic goal in rehabilitation robotics is to be able to use the robot for a task that is done only once. This is in contrast to with most industrial uses of robots, where robots are used in preprogrammed repetitive tasks. Thus many tasks in rehabilitation robotics can be said to be unique in the sense that a motion for a certain task e.g. picking up an object from a shelf or opening a door, cannot be preprogrammed. This indicated that there is a need for manual control of the robot in a way similar to telemanipulator.

In the late 1940's and 1950's Ray Goertz began developing the first rudimentary manipulators at the Argonne National Laboratory near Chicago [5]. Although prosthetic limbs had been developed by the 1940's which could be actuated by leather straps tied to the wearer's body Goertz's research ushered in the first modern remote manipulators. His

mechanisms were mechanical pantograph devices that allowed radioactive materials to be handled by operators outside of the hot area. Later electrical servos replaced mechanical linkages and cameras replaced direct viewing, so that the operator could be arbitrarily far away. The increase in the computer power in the 1960's and 1970's made possible a much wider variety of computer controlled motion and force feedback. Some of the early applications included the retrieval of a nuclear bomb lost on the ocean bottom near Spain in 1960 with the US Navy's CURV vehicle and the construction of a force feedback powered artificial elbow for amputees. From that period people have been remotely controlling machines, devices have been built to give us a sense of feel when controlling remote actions.

As remote manipulation technology progressed, several types of systems and concepts were defined. Though many definitions are available today, the definitions proposed by Sheridan [6] seem to be the most precise in the robotics literature. Goertz's concept, in which a person's sensing and manipulation capability is extended to a remote location, is referred to as a teleoperator. According to Sheridan, teleoperation extends a person's sensing and/or manipulating capability to a location remote from that person. [It] includes artificial sensors of the environment, and communication channels to and from the human operator. In addition, a teleoperator may include arms and hands or other devices to apply forces and perform mechanical work on the environment. The term teleoperation refers most commonly to direct and continuous human control of teleoperator, but can be used to encompass *telexotics* as well. Teleoperation therefore involves augmenting,

supervising, or substituting artificial intelligence and control functions of the robot with the intelligence and pattern recognition abilities of the human operator.

In these systems, the kinematic chain, which is manipulated, by the operator and may provide force feedback, is referred as the *master*, while the remote manipulator is referred to as the *slave*. *Telepresence* encompasses the ideal situation in which ‘operator receives sufficient information about the teleoperator and the task environment, displayed in a sufficiently natural way, that the operator feels physically present at the remote site’.

Telerobot, according to Sheridan is ‘an advanced form of teleoperator the behavior of which a human supervises through a computer intermediary. That is, the operator intermittently communicates to the computer information about goals, constraints, plans, contingencies, assumptions, suggestions, and orders relative to a remote task, getting back integrated information about accomplishments, difficulties, and concerns, and (as requested) raw sensory data. The subordinate telerobot executes the task on the basis of information received from the human operator plus its own artificial sensing and intelligence’.

Other definitions of telerobotics have included systems in which the human operator remains in continuous control as in teleoperation, but various forms of machine intelligence are incorporated in the form of variation of control parameters or automatic motions. The approach here will be to retain the direct control of the human operator, and as such is more akin to “teleoperation” as defined by Sheridan.

2.2 Previous Work

Several strategies in which human decisions are merged with computer assistance have been investigated in recent years. One explicit attempt to combine human and machine control was made by Hayati and Venkataraman [7]. In this strategy, force and velocity commands from a master input device were combined with those from an automatic controller along each direction to be controlled, using similar methods to those used in shared control. Backes [8] presented a controller that superimposed various preprogrammed motions onto the command from the master, which could be initiated when desired. Computer generated force inputs were added to those from the master in the impedance controlled formulation to assist in controlling the motion of the manipulator, such as ensuring joint limit and obstacle avoidance. The idea of having a variety of preprogrammed control modes available was presented by Yokokohji et al. [9], allowing a better match between controller and task. This paper proposed that each degree of freedom of the manipulator should have available several control modes, such as autonomy, teleoperation with force feedback, teleoperation without force feedback, etc., which could be selected with a switch by the operator when desired. Marth et al. [10] used an event-based controller to improve impact performance, which provided more automatic selection of controller parameters. The same philosophy was used later by Guo et al. to allow semiautonomous obstacle avoidance in a teleoperated system. In this implementation, velocity commands from an input device are superimposed onto an autonomous motion, allowing human controlled obstacle avoidance. After an obstacle is passed, the manipulator is autonomously moved back to its original preprogrammed path.

Elaborate virtual constraints have also been used to assist an operator in maneuvering a slave manipulator, including those by Joly and Andriot and Kosuge et al.. In their work, virtual mechanisms have been used to constrain the manually controlled manipulator's motion on a desired surface or to be pulled into alignment with a task. Aigner integrated potential field effects and remote control of a manipulator providing teleoperation assistance. Potential fields were used to produce velocity commands, which, when added to those generated by the input device, maneuvered the manipulator away from walls or around obstacles automatically.

The current work demonstrates the utility of a robotic setup to perform a set assessment tasks. The robotic setup introduces the advantages of the haptic rendering features in the interface and the assistance function concept previously used in teleoperation tasks. Results suggest that the overall performance and time execution considerably improve when these forms of assistance are used.

2.3 Teleoperation Overview

2.3.1 Teleoperation Research for Nuclear Clean-up

Teleoperation was first developed for use in the nuclear industry and has since found applications in undersea explorations, waste management and space exploration. A part of the U.S. nuclear legacy is 282 underground tanks that contain millions of gallons of radioactive waste and 7 calcine vaults. The tanks are located at Hanford Site, Idaho

National Engineering and Environmental Laboratory, Oak Ridge Reservation, Savannah River Site, and West Valley Demonstration Project. The United States Department of Energy (DOE) faces several significant challenges in remediating the transuranic and high-level waste stored in the underground tanks.

Environmental restoration and waste management challenges in the DOE, and around the world, involve radiation or other hazards, which will necessitate the use of remote operations to protect human workers from dangerous exposures. Remote operations carry the implication of greater costs since remote work systems are inherently less productive than contact human work due to the inefficiencies/complexities of teleoperation. To reduce costs and improve quality, much attention has been focused on methods to improve the productivity of combined human operator/remote equipment systems; the achievements to date are modest at best. The most promising avenue in the near term is to supplement conventional remote work systems with robotic planning and control techniques borrowed from manufacturing and other domains where robotic automation has been used. Practical combinations of teleoperation and robotic control will yield telerobotic work systems that outperform currently available remote equipment. It is important to recognize that the basic hardware and software features of most modern remote manipulation systems can readily accommodate the functionality required for telerobotics.

Current methods for modifying, operating, cleaning and decontaminating these pits are labor intensive and costly and result in a high dose to workers. The assistance

strategies developed and implemented in this work were used to enhance clean-up operations in this environment by reducing personal exposure and increasing the efficiency of task execution.

2.3.2 Rehabilitation Robotics Applications

In the past three decades, many rehabilitation robots for people with manipulative and locative disabilities have been designed and developed. These devices have attempted to fully or partially substitute or restore the lost functions and enable people with disabilities perform many activities of daily life affecting their employment and quality of life.

The earliest research in this area began in the 1970s [11]. The Rancho “Golden” arm, developed at Rancho Los Amigos Hospital in Downey, California in 1969 was the first success for rehabilitation robotics [12]. Since 1980’s, a considerable progress has been made in Europe [13]. MANUS, built on the conclusions of the Spartacus project by specifying and developing a wheelchair-mounted manipulator, is the most well known of those successors. Another project that has enjoyed relative success is the Handy 1 [14], which is primarily used as a feeding device for children with cerebral palsy in eating independently. More recently, besides improving eating skill, the aid has been considered for other activities including drinking, shaving, making up and leisure activities [14]. The RAID workstation can even access books, paper documents, diskettes, and CD_ROM’s placed on an integral shelving system [15].

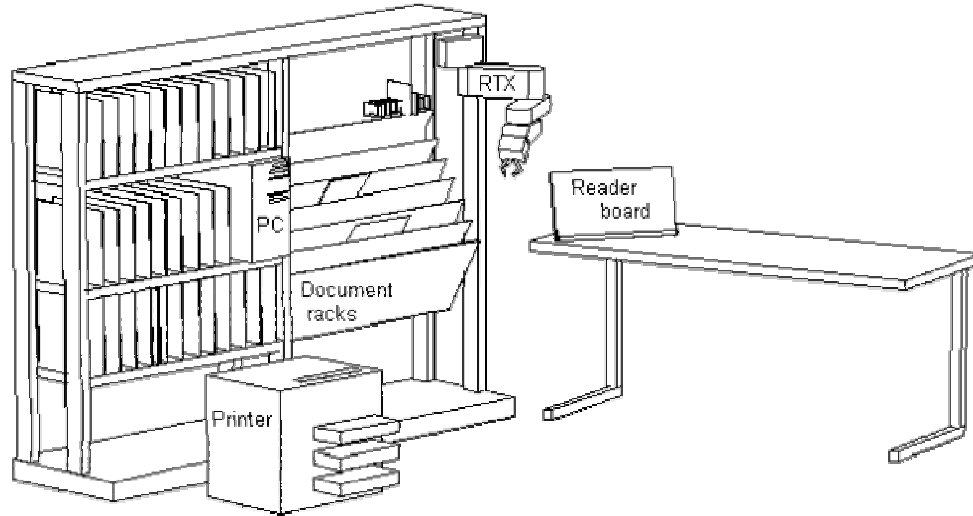


Figure 2.1 RAID Work Station

During this period, several successful projects also emerged in North America. DEVAR (desktop assistant robot for vocational support in office settings) uses a PUMA-260 robot mounted on an overhead rack that performs pre-programmed tasks in highly structured environment. [16]. It can be used to handle paper, floppy disks, pick up and use the telephone, and retrieve medication. RAA (Robotic Assistive Appliance), developed at the Neil Squire Foundation in Vancouver, Canada, offers a human size manipulator at a workstation with 6 degrees of freedom with either programmed or direct control [17]. MIT-MANUS is the most successful therapy platform to undergo intensive clinical testing [18]. This device is a planar, two-revolute-joint, back drivable robotic device that attaches to the patient's hand and forearm through a brace. The patient can move the robot, or the robot can move the patient, in the horizontal plane. The patient receives feedback of the hand trajectory on the computer screen.

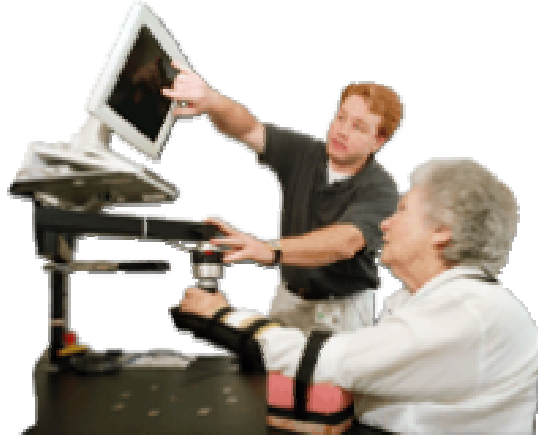


Figure 2.2 MIME Work Station

MIME is powerful enough to move a patient's arm throughout the three-dimensional workspace against gravity [19]. When the patient moves her unimpaired arm, a mechanical digitizing stylus senses the movement. The PUMA 560 robot arm then moves the patient's impaired arm along a mirror-symmetric trajectory. The result of clinical tests with MIME showed integration of robot-aided therapy into clinical exercise programs would allow repetitive, time-intensive exercises to be performed without one-to-one attentions from a therapist [20]. In 1997, Toshiro Noritsugu and Toshihiro Tanaka used pneumatic rubber artificial muscle actuators to operate a 2-DOF manipulator driven for functional recovery therapy. In this system, the mechanical impedance of the human arm under therapy is used as an objective evaluation index for therapy, so the robot can adjust its own impedance parameters according to the physical condition of the patient [21]. M.J. Johnson etc. used Driver's SEAT to motivate stroke survivors to use their upper limb in ADLs(activities of daily living)[22]. F.A. Mussa-Ivaldi and J.L. Patton even argued that robots could teach people how to move their arm through designing force fields [23]. These force fields are capable of inducing the desired movements as

after-effects of the adaption triggered by prolonged exposure to the fields. D. Reinkensmeyer designed a Java therapy, which is an inexpensive robotic tele-rehabilitation system for arm and hand therapy [24].

The Palo Alto Rehabilitation research and Development division of the Department of Veterans Affairs Medical Center has an extensive history in Rehabilitation Robotics Research [25]. Their philosophy on human-machine interface has been to emphasize autonomy of the robot during operation. In Vancouver, Canada the Robotic Assistive Appliance (RAA) was developed at the Neil Squire Foundation [26]. The RAA resulted from eight years of research in rehabilitation robotics. The use of Robotics in vocational applications to aid in the job placement of individuals with manipulation disabilities have been explored in recent studies at the Applied Science and Engineering Laboratory (ASEL) at the University of Delaware and the duPont Hospital for Children.

One perennial problem in rehabilitation robotics is identifying which of the users need should be addressed. Surveys were conducted in England and North America and included over 200 potential users of rehabilitation technology. The surveys include predevelopment questionnaires that focused on user task ability and anticipated use of an orthosis or rehabilitation robot. Determining the user task priorities was mandatory for any further training and technology assessment. The results of the surveys indicated that reaching, gripping, and picking up objects from a shelf are all very important tasks. Specific vocational tasks did not appear on the priorities list. However, many of the manipulation tasks that were rated highly in the studies could, in fact, facilitate

employment. Other tasks, such as writing, operating a computer, or using a telephone were also important and indeed employment related. Also, the study concluded with a very important design issue for future research, which was that the users expressed the need of a wide range of object manipulation tasks in a variety of unstructured environments.

In 1997, Toshiro Noritsugu and Toshihiro Tanaka used pneumatic rubber artificial muscle actuators to operate a 2-DOF manipulator driven for functional recovery therapy. In this system, the mechanical impedance of the human arm under therapy is used as an objective evaluation index for therapy, so the robot can adjust its own impedance parameters according to the physical condition of the patient [27]. M.J. Johnson etc. used Driver's SEAT to motivate stroke survivors to use their upper limb in ADLs [28]. F.A. Mussa-Ivaldi and J.L. Patton even argued that robots could teach people how to move their arm through designing force fields. These force fields are capable of inducing the desired movements as after-effects of the adaption triggered by prolonged exposure to the fields. D. Reinkensmeyer designed a Java therapy, which is an inexpensive robotic telerehabilitation system for arm and hand therapy [29].

These therapy platforms are mostly for upper-limb disabilities. For mobility impairment, usually wheelchair is used for its movement compensation. There are not many therapy platforms available to date. The Mechanized Gait Trainer (MGT) is a singly actuated mechanism that drives the feet through a gait-like trajectory. The Lokomat is a motorized exoskeleton worn by the patients during treadmill walking.

Chapter 3: PHANTOM® Haptic Interface and GHOST® SDK

3.1 Introduction

In order to operate a robotic hardware, an input device is required. There are three main categories of input devices. The teach pendent is the most simple form of input, and is not much more than a specialized keyboard to control the manipulator. Most manipulators come standard with this type of control and they are usually intended for occasional use, mainly to perform calibration movements and the like. The second category is a three-dimensional controller without force feedback. This undoubtedly offers more control to the user. However, this type of control is not necessarily preferred, as it does not provide the user with any feedback, and thus makes the control of the robot dependent on the sensory feedback of the user, which can be limited depending on the type of disability. However, most devices come with this type of control along with a teach pendent. Joysticks, in a way, fall into this category, usually providing two-dimensional control, oftentimes a simple third dimension, by allowing the user to depress the joystick slightly. The most interesting type of input is a three-dimensional controller with force feedback, as this provides the user with the most options. This type of control can be found in a variety of different forms.

A widely used three-dimensional controller is the PHANToM [3.7], marketed by SensAble Technologies. It provides the user with 6 degrees of freedom and also provides force feedback. This haptic interface is a particular kind of window through which a user can experience or manipulate a device that senses body movement. Haptic feedback implies computer control over the tactile or kinesthetic properties of a physical interface, permitting real time representation of a virtual or remote environment rather than a specific, constant handle. A manual interface without actuation and computer control (for instance, a mouse) can also provide benefits of physicality and continuous control, either actively or passively actuated haptic interfaces can change their feedback in response to the environment they display and control.

3.2 Phantom Feature Overview

The PHANToM force reflecting haptic interface [44] was originally designed by Massie and Salisbury and subsequently commercialized by SensAble Technologies, Inc., (Woburn, MA). Developed at MIT, the PHANTOM elegant design provides high-fidelity 3D force-feedback, the ability to operate in an office/desktop environment, compatibility with standard PCs and a universal design for a broad range of applications. The system has been used successfully in a multitude of applications, in a typical virtual environment and telerobotics applications. Early haptic interfaces were often termed “master manipulators”. This term was used because the haptic interface was used in a master / slave configuration for teleoperation. They have also been called “kinesthetic interfaces” because of their direct correlation with the sensations derived from bodily movements.

Recently, however, the term “haptic interface” has grown increasingly popular because “haptic” refers to the entire perceptual system associated with touch and manipulation, not just one of the many contributing sensory systems (e.g., kinesthetic, tactile, etc.).

The PHANToM haptic device can be used as a means for achieving interactive simulation. Haptic interaction occurs at the point of contact between the PHANToM and the scene object. The PHANToM haptic interface provides 3D force-feedback using the General Haptic Open Software Toolkit GHOST SDK. This haptic device connects to the computer via an extended parallel port (EPP) and is powered by a compact, universal (110/230 VAC) power supply. An Encoder Stylus, which provides six-degree-of-freedom positional sensing is connected to the end- effector.

The PHANToM allows users to actually feel virtual objects. Unlike buzzing tactile stimulators, the PHANToM actively exerts an external force on the user's fingertip creating the illusion of interactions with solid virtual objects. The PHANToM contains 3 motors, which control the x, y, and z forces exerted on the user's fingertip. Mounted on each motor is an optical encoder to determine the x, y, and z position of the user's fingertip. The torque from the motors is transmitted through a proprietary transmission cable to a stiff, lightweight linkage. Incorporated at the end of this linkage is a passive 3 degree-of-freedom set of gimbals attached to a thimble.

The passive gimbals allow the thimble to rotate so that a user's fingertip may assume any comfortable orientation. The user's fingertip can be modeled as a point or frictionless

sphere within the virtual world. The device has low friction, low inertia, and no unbalanced weight. Thus movements through free virtual space are unimpeded. For higher precision the stylus, in the laboratory at the University of South Florida, was used in place of the thimble.

The PHANTOM 1.5/6DOF used for the present research is a desktop tool that allows for the exploration of application areas requiring force feedback in six degrees of freedom (6DOF). Examples of such applications include virtual assembly, virtual prototyping, maintenance path planning, teleoperation and molecular modeling. This model provides full force and torque feedback to the user. Simulating torque force feedback makes it possible to feel the collision and reaction forces and torques of a part in a virtual assembly path, or the rotational torques supported by a remote "slave" robot in a teleoperation environment.

An operator can manipulate the stylus to control the tip of a virtual pencil or paintbrush, to touch the virtual surfaces. A photograph of PHANToM is shown in Figure 4.1. The second component is the middleware GHOST, Physics Engines, software. It handles the complex computations associated with touch and allows developers to deal with high-level objects and physical properties such as location, mass, friction and stiffness using Visual C++ code. The last component includes end user applications such as simulation and modeling software for CAD. These applications enhance the possibilities of 3D Touch Technology. A representation of the system is shown in figure 4.2.



Figure 3.1. Phantom Hand Controller Designs

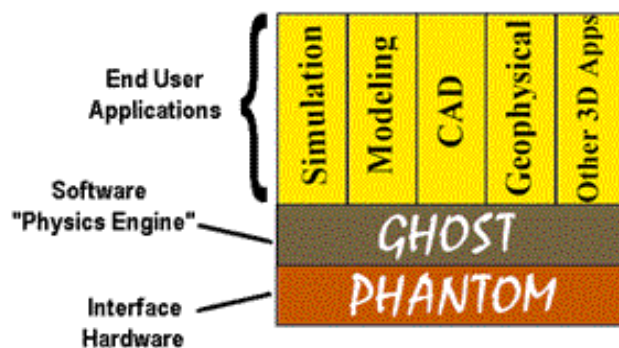


Figure 3.2. Representation of the 3D Touch Technology System®

The interface of PHANToM with C++ allows for the generation of force feedback effects such as collisions, force fields, torque, and inertia. The PHANToM handles

graphics through OpenGL, which provides a standard Graphics Library. A graphical scene is represented by a number of transformations that are accumulated vertically. The scene starts with a root or parent and moves down to branches or children. A representation of a sample structure is shown in figure4.3.

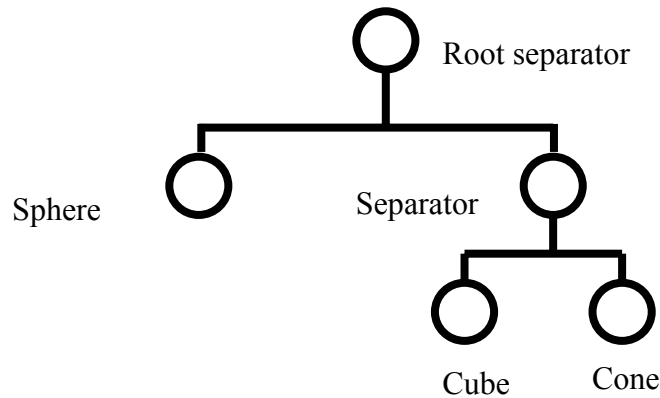


Figure 3.3. Sample Program Structure for the Development of a Scene

3.3 PHANTOM® Capabilities

3.3.1 Conceptual 3D Modeling and Design

The PHANTOM offers a user-friendly computer interface for designers. With its 3D touch technology, it provides a natural, intuitive interface that enables creative digital expression for 3D modeling and design. Using 3D touch technology benefits digital automation, reduces development cycles, and increases design options.

3.3.2 Product Development and Manufacturing

With the mouse or keyboard it is difficult to perform critical engineering or manufacturing analysis of CAD/CAM models for issues such as manufacturability, assembly and maintenance planning, or ergonomics. By adding 3D Touch capabilities to digital designs, engineers are able to perform digital product design and manufacturing engineering activities on the computer in the same way they would in the physical world: rapidly, with their sense of touch.

Using the GHOST® Software Development Kit, CAD/CAM software vendors can augment their applications with 3D Touch capabilities. This will enable end-users to interact with digital models in the most natural way. It will be possible to "feel" as well as see collision detection or the intersection of complex surfaces in 3D models, to provide hands-on assembly training, and to use one's sense of touch to test human factors. Using the PHANTOM system, engineers can interact with model assemblies to assess fit, ease of assembly or manufacturability, without creating a mock-up. The benefits of this approach include reduction of product development cycles, earlier detection of design problems, and more effective training.

The following are some of the applications made possible by integrating 3D Touch technology into CAD/CAM:

- Virtual mock-up: realistically simulated human factors, such as switches, knobs, or snap-fits
- Virtual assembly: form and fit evaluation, maintenance path planning, manufacturability analysis and assembly training
- Design review: form and fit evaluation, dynamic modeling of kinematic assemblies.

3.3.3 Computer Animation

Using a mouse, keyboard, or stylus pad to interact with 3D environments in computer animation is difficult and time consuming. There is no easy way for an animator to simply reach into a scene to adjust a character's position, apply digital paint, or mold a model's shape. These procedures have been tedious and difficult to perform in the past. Using the GHOST software development kit, computer animation software vendors can enhance their applications with 3D Touch technology, providing a more natural and intuitive interface for their end-users that improves their productivity and streamlines the animation design process.

Adding a tactile interface to computer animation will permit end-users to interact with digital models in the most natural way, with their hands. This will enable applications such as computer-aided sculpting of 3D models, touch-guided collision detection while animating kinematic sequences, and an artistic and intuitive interface for 3D painting and texture mapping. The benefits of a 3D Touch interface include greatly

reduced learning curves for animation software packages, shortened development cycles for computer animated models, increased realism, and unrestricted creativity. The applications of the PHANToM haptic interface in animation include:

- Inverse kinematics: touch-enhanced collision detection
- 3D Paint: simulated realistic painting, touch-enhanced texture maps
- Hands-on modeling: touch-enhanced morphing and sculpting

3.3.4 Telerobotics

Teleoperation has been used in applications in which the environment endangers the human due to high temperatures, radiation or the high pressures of the deep ocean. Additionally, teleoperation is employed whenever the objects to be manipulated are too large or heavy and the human body is limited in strength or dexterity. Adding the sense of touch to the guidance of a robot arm allows the user greater accuracy and fidelity in remote operations and manipulations, enabling greater effectiveness in applications such as Hazardous materials handling and Remote vehicle control.

Since teleoperators are separated from the human body, help can be provided more effectively for people whose strength; freedom of motion and dexterity is limited in some fashion. Applications already exist in which robots can manipulate table utensils under direct control of the person being fed. The Handy 1 is a commercially available

robotic aid that enables people with disabilities to be more independent in daily activities such as eating, drinking and washing (M. Topping, 2001).

3.3.5 Biomedical and Prosthetic Applications

These include augmentation, filtering and supporting manual activities by persons with disabilities; and in fact are among the first areas of haptics feedback research. Because of their personal and affective affordances and capability for continuous control, haptic interfaces may be the way to restore both improved usability and humanity to these interactions. The representation of virtual objects and sense of remote telepresence is perhaps the most widespread haptic application today.

The use of a 3D Touch interface as a diagnostic aid with digital MRI, X-ray or ultrasound images interactively conveys complex information to the operator. This is particularly useful when attempting to precisely locate a feature or to understand the spatial relationships of 3D structures. Minimally invasive surgery and microsurgery can be greatly enhanced by the application of a high-fidelity haptic device like the PHANToM interface for input and control.

The precision and high fidelity offered by the PHANToM interface open up a whole new world in advanced medical applications, such as:

- Visualization: data segmentation and dataset navigation

- Diagnosis: digital palpation and virtual endoscopy
- Surgical Simulation: surgical planning and surgical training
- Dexterity Enhancement: touch-enabled microsurgery and touch-enabled telesurgery
- Rehabilitation: simulated exercise environments

3.3.6 Molecular and Nano-manipulation

By linking the PHANTOM interface with nano-manipulation devices such as Atomic Force Microscopes, researchers can actually feel and manipulate viruses or position molecules on a substrate. Applications include: Cellular manipulation and Molecular docking

3.4 GHOST® SDK Overview

The GHOST® SDK [45] (General Haptic Open Software Toolkit) is a powerful, C++ software tool kit that greatly eases the task of developing touch-enabled applications. GHOST® works as the "physics of touch" engine which takes care of the complex computations and allows developers to deal with simple, high-level objects and physical properties like location, mass, friction and stiffness. Developers can use GHOST's libraries of 3D prismatic objects, polygonal objects, and touch effects to add a convincingly physical dimension to a variety of applications, including, medical, animation, visualization and CAD, among others. At the same time, GHOST's flexible,

extensible architecture makes it a powerful platform for haptics researchers and other developers who need to add new shapes and dynamics, as well as implement lower-level, direct force effects.

It is a object-oriented toolkit that represents the haptic environment as a hierarchical collection of geometric objects and spatial effects. The GHOST® SDK provides an abstraction that allows application developers to concentrate on the generation of haptic scenes, manipulation of the properties of the scene and objects within the scene, and control of the resulting effects on or by one or more haptic interaction devices. Thus, application developers need not be concerned with low-level force and device issues. In addition, the GHOST® SDK is highly portable, both in terms of supported computational environments and in terms of different haptic interaction devices. It supports the entire family of STI PHANTOM. haptic interface devices.

The GHOST® Application Programming Interface (API) enables application developers to interact with haptic interaction devices and create haptic environments at the object or effects level. Using the GHOST® SDK, developers can specify object geometry and properties, or global haptic effects, using a haptic scene graph. A scene graph is a hierarchical collection (tree) of nodes. The internal nodes of the tree provide a means for grouping objects, orienting and scaling the sub-tree relative to the parent node, and adding dynamic properties to their sub-trees.

The terminus of the haptic interaction device is represented as a point within the scene graph. The GHOST® SDK automatically computes the interaction forces between this point and objects or effects within the scene, and sends forces to the haptic interaction device for display. Applications can treat the haptic interaction point using the *GHOST SDK* as either 1) the physical location of the haptic interaction device terminus within its physical workspace, or 2) the computed location of the interaction point constrained to the surface of geometric objects. The latter point position is known as the Surface Contact Point (SCP). The SCP is used to generate all GHOST® SDK surface interaction forces.

The GHOST® SDK does not generate visual representations of objects within the haptic scene graph. If graphics are desired, the GHOST® SDK does, however, provide graphic callback mechanisms to facilitate integration between the haptic and graphic domains. GhostGL is a library that can render any GHOST® SDK scene using OpenGL. It provides a quick and easy way to add graphics to any GHOST® SDK application. Once a GHOST SDK scene graph has been created it can be passed to the GHOSTGL routines that traverse and render a graphical representation of each node in the scene.

A typical application using the GHOST® SDK *must*:

- Create a haptic environment through the specification of a haptic scene graph
- Start the haptic simulation process (the servo control loop)
- Perform application-specific (core) functions that include the generation and use of computer graphics

- Communicate with the haptic simulation process as needed
- Perform clean-up operations when the application ends

A typical GHOST® SDK application consists of at least two processes: a real-time GHOST® SDK process that handles the haptic simulation, and the application-level process. The GHOST® SDK automatically sets up and manages these processes. The call to `gstScene::startServoLoop` creates the real-time haptic simulation process and immediately returns control to the application. After returning from the `gstScene::startServoLoop` method, the application must begin its own application loop. It can then perform its own actions and interact with the haptic process. The haptic simulation process continues to run until it is explicitly stopped using the `gstScene::stopServoLoop` method, an error occurs in the servo loop, or the application exits. The interaction between the application and the haptic simulation process may be accomplished using either GHOST® SDK methods to directly set and query the state of the haptic scene graph and the haptic simulation, or using GHOST® SDK callback mechanisms. Callback mechanisms provide an efficient, asynchronous mechanism for updating the state of the application process when areas of interest in the haptic scene graph change (i.e., because of human or PHANToM® haptic interface device interactions, or because of GHOST® SDK object dynamic properties). The most common use and need for callback mechanisms is to synchronize an application's visual representation or scene graph with that of the haptics scene graph. When the application is ready to exit, the haptic simulation must be stopped using `gstScene::stopServoLoop` and any application-specific cleanup required is performed by the application.

The `gstScene` class maintains a pointer to the root of a node graph that becomes the scene database used for haptic simulation. The node graph pointed to by the `gstScene` object is referred to as the haptic scene graph or scene graph. Only objects within the scene graph become part of the GHOST® SDK haptic simulation. Every GHOST® SDK application should contain exactly one instance of a `gstScene` object. The `gstScene` instance maintains a pointer to the haptic scene graph and performs the haptic simulation by repeating the servo loop at a rate of 1kHz. The call to `gstScene::startServoLoop` creates the real-time haptic simulation process and immediately returns control to the application. After returning from the `gstScene::startServoLoop` method, the application must begin its own application loop. It can then perform its own actions and interact with the haptic process. The haptic simulation process continues to run until it is explicitly stopped using the `gstScene::stopServoLoop` method, an error occurs in the servo loop, or the application exits. The interaction between the application and the haptic simulation process may be accomplished using either GHOST® SDK methods to directly set and query the state of the haptic scene graph and the haptic simulation, or using GHOST® SDK callback mechanisms.

3.4.1 Adding Graphics with GHOST GL

GhostGL is a library that can render any GHOST® SDK scene using OpenGL. It provides a quick and easy way to add graphics to any GHOST® SDK application. Once a GHOST® SDK scene graph has been created it can be passed to the GHOSTGL routines that traverse and render a graphical representation of each node in the scene

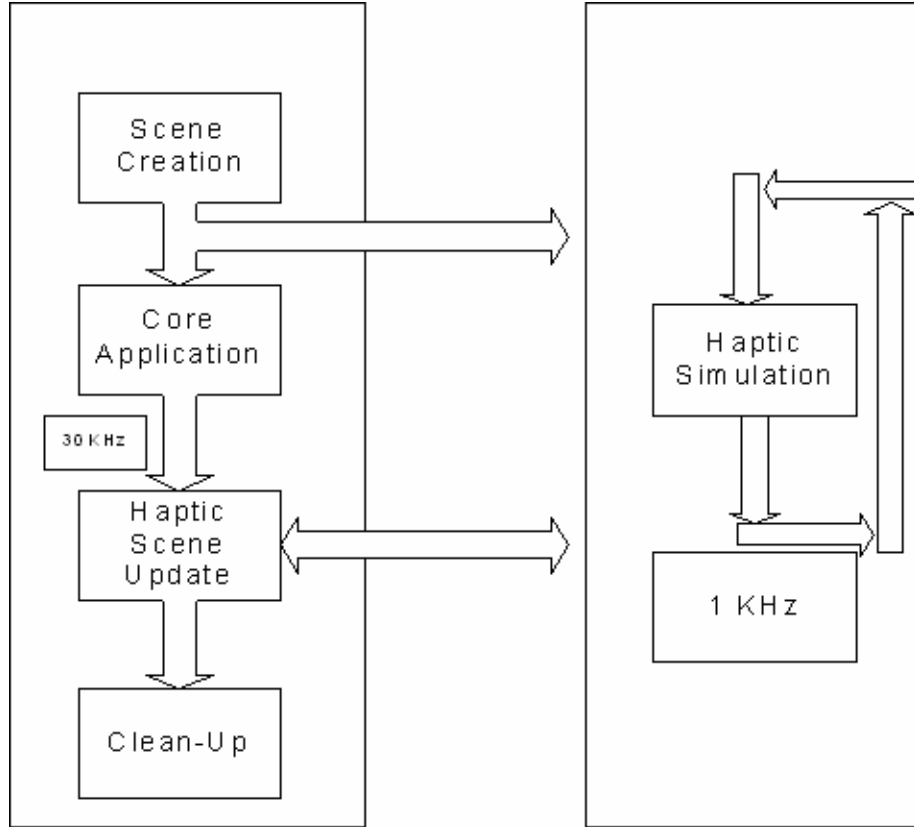


Figure 3.4 Haptic Simulation Process

GHOST GL can be used with GLUT to add OpenGL rendering to any GHOST SDK program. To do this simply the following four lines are added to the program:

- ghostGLUTManager header file `#include <ghostGLUTManager.h>`
- Instance of ghostGLUTManager.


```
ghostGLUTManager * glutManager = ghostGLUTManager::CreateInstance(argc,  
argv, "MoveTask");
```

- Loading the gstScene object into the ghostGLUTManager.

```
glutManager->loadScene(&myScene);
```

- Starting the ghostGLUTManager main graphics loop.

```
glutManager->startMainloop();
```

As objects in the GHOST® SDK scene move around the corresponding graphical objects will be redrawn in their new locations.

Chapter 4: SolidWorks Simulation of RRC Manipulator

4.1 Introduction

SolidWorks is one of the worlds leading solid modeling CAD programs. It is used for mechanical design of parts and assemblies. It contains powerful graphics and kinematics engine that allows realistic representation and manipulation of these parts and assemblies in its graphics window.

A SolidWorks assembly is a collection of solid parts, each of which has a set of physical characteristics and relationships with one another. Each part is made up of orderly arranged features such as extrude, cut etc., some of which are based on two dimensional sketches, others are truly three dimensional in nature and may be created on top of existing features. SolidWorks creates a hierarchical structure of all the data about these parts. An assembly enables relationship between mates. These mates constrain some parts in relation to others or to the world, so while one part may be fixed, another may slide on other parts or rotate in space.

The present chapter deals with the simulation of the Robotics Research Corporation Manipulator in SolidWorks. This is an offline programming and simulation environment

for the RRC Manipulator using the Phantom as an input device. An Add-In DLL was developed to interface the SolidWorks model with the Phantom Haptic Interface.

4.2 Robotics Research Corporation Seven DOF Manipulator

In order to move the robot end-effector to any arbitrary position in a three dimensional workspace, at least a three joint or degree-of-freedom manipulator is required. If one wants to control the position as well as the orientation, at least six degrees of freedom are required in the manipulator. When a mechanical system has more independent inputs (joints) than are necessary to define the desired output it can be labeled a redundantly actuated system or kinematically redundant system. Defining the position and orientation of a body in 3D space requires 6 variables (X, Y, Z, Roll, Pitch, and Yaw). A robot with seven or more kinematically independent joints is a redundant robot. Generally a finite number of configurations exist for a non-redundant manipulator to reach a certain position and orientation. However a redundant manipulator has infinite configurations to reach the same end effector position.

In recent years, a few seven degree-of freedom redundant manipulators have been build for operating in three dimensional space. These include the CESAR manipulator at the Oak Ridge National Laboratory, the Laboratory Telerobotic Manipulator (LTM) at the NASA Langley Research Center, and the commercially available Robotics research Corporation (RRC) Manipulator. Like a human arm which has seven degrees of freedom:

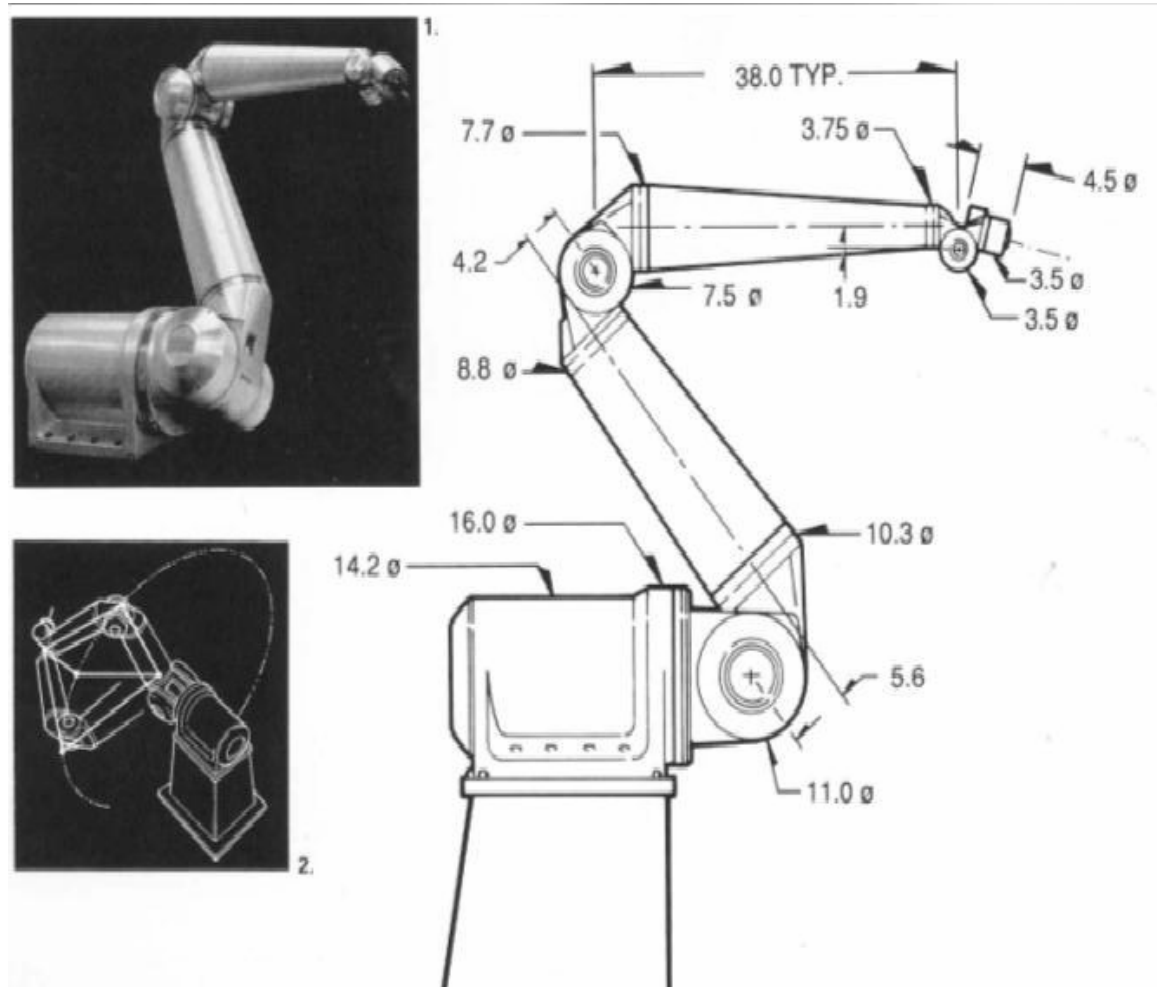
three in the shoulder, one in the elbow, and three in the arm below the elbow, these manipulators can change configuration without moving the end-effector.

The Robotics Research Corporation (RRC) model K-2107 robot has seven joints. Each joint is powered by an electric motor has an arthropod construction, which produces high stiffness but yields less than ideal dexterity. Harmonic drive motor modules with brakes are used throughout. These motors serve as either roll or pitch joints. The lightweight tubular structure and high performance motor/speed reducer modules provide the arm with an advantageous payload-to-arm ratio. Internal pressurization against environmental contamination is achieved through its all-metal revolute joints. The use of single revolute joints in the design results in the arm achieving high precision and high load capacity within the narrow limits of the joint's dexterity. The manipulator with its control system is shown in Figures below.

Seven joints provide it with a redundant degree of freedom, which may be used to avoid obstacles, joint limits, and singularities. It has a reach of 83", with a repeatability of 0.0005". However, its payload capacity is limited to about 5 lbs due to its long links and the weakness of the wrist pitch motor.

Joints 1, 3, 5, and 7 are roll type joints, while 2, 4, and 6 are wrist type joints. The total length of the arm when all the joints are positioned forward, reaches 2.1 meters, about seven feet. Figure also shows the schematic of the robot manipulator's seven joints

including and location of each joint and their respective travel limits. It has a reach of 83", with a repeatability of 0.0005".



However, its payload capacity is limited to about 5 lbs due to its long links and the weakness of the wrist pitch motor. The state-of-the-art in structured programming control technology is embodied in the RRC R2 controller. The manipulator uses a PC based

controller. The controller uses inputs from the computer's graphical user interface (GUI) or the teach pendant as the reference position for each of the seven joints. From these positions, the inverse kinematics are calculated, and seven joint commands are determined and sent to the low level controller. The robot controller is capable of position, velocity, and torque control for the motors for each of the seven joints to maintain the appropriate joint angles of the manipulator.

4.2.1 Controller Architecture

The Robotics Research Corporation (RRC) R² Control Software™ provides the most advanced robotic control capabilities in the industry. It supports an open architecture at each level of control to facilitate third party application and control development and integration. The R² Control Software offering has been packaged as a turnkey controller, which executes on the Windows NT® platform in conjunction with the INtime® operating system

The robot controller consists of two primary components of operation; the INtime real-time component (R2 RTC) and the NT client-server upper control level component. The R2 RTC provides deterministic, hard real-time control with typical loop times of 1 to 4 milliseconds. This component performs trajectory planning, Cartesian compliance and impedance force control, forward kinematics, inverse kinematics, and advanced heavy deposition welding.

R2 Architecture

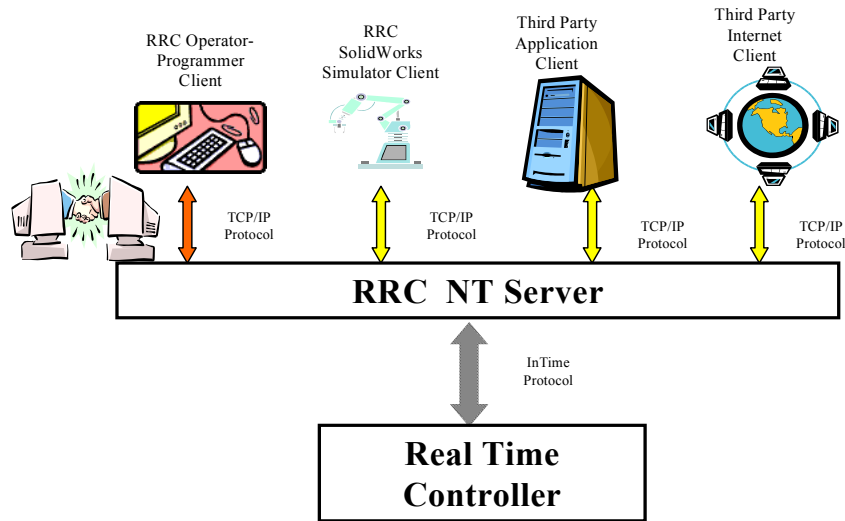


Figure 4.2. RRC R2 Control Architecture™

A customer can interface with any level of the controller. The controller can accept commands in the form of high level Cartesian goal points down to low level servo commands of joint position, torque or current.

4.3 SolidWorks and Application Programming Interface

The SolidWorks Application Programming Interface (API) can be used to automate and customize the SolidWorks software. The API contains hundreds of functions that you can call from Visual Basic (VB), Visual Basic for Applications (VBA), VB.NET, C++, C#, or SolidWorks macro files. These functions provide direct access to SolidWorks

functionality such as creating a line, inserting an existing part into a part document, or verifying the parameters of a surface. The SolidWorks API uses an object-oriented approach. It exposes its API functionality through standard COM objects. A COM interface is used here for creating the Add-In DLL project.

The SolidWorks Application wizard was used to create an application framework for a SolidWorks Add-In DLL. The DLL contains the following important files

- SWSim.cpp defines the interface to the solidworks and the initialization routines for the DLL.
- SWSim.rc contains all the Microsoft windows resources that the program uses.
- SWSim.def contains information about the DLL that must be provided to run with Microsoft Windows. It defines parameters such as name and description of the DLL.
- SWSim.clw contains information used by class wizard to edit existing classes or add new classes.
- CcItem.cpp implements the CControlItem class which serves as the foundation class for SolidWorks event handlers.
- ExampleApp.cpp implements the RRCSimApp class which is the basis for entire application.
- HandleEvents.cpp implements the event handlers for SolidWorks notifications.
- SWSimSocket.cpp establishes the socket communication between Solidworks and GHOST application using `m_swSimSocket.Create(1123, SOCK_DGRAM)`
- Other standard files include StdAfx.h, StdAfx.cpp and Resource.h

4.4 SolidWorks Simulation of RRC Manipulator

This is an off-line programming and simulation environment for the RRC Manipulator. A graphics model of the manipulator was designed in SolidWorks and is shown in Figure below.

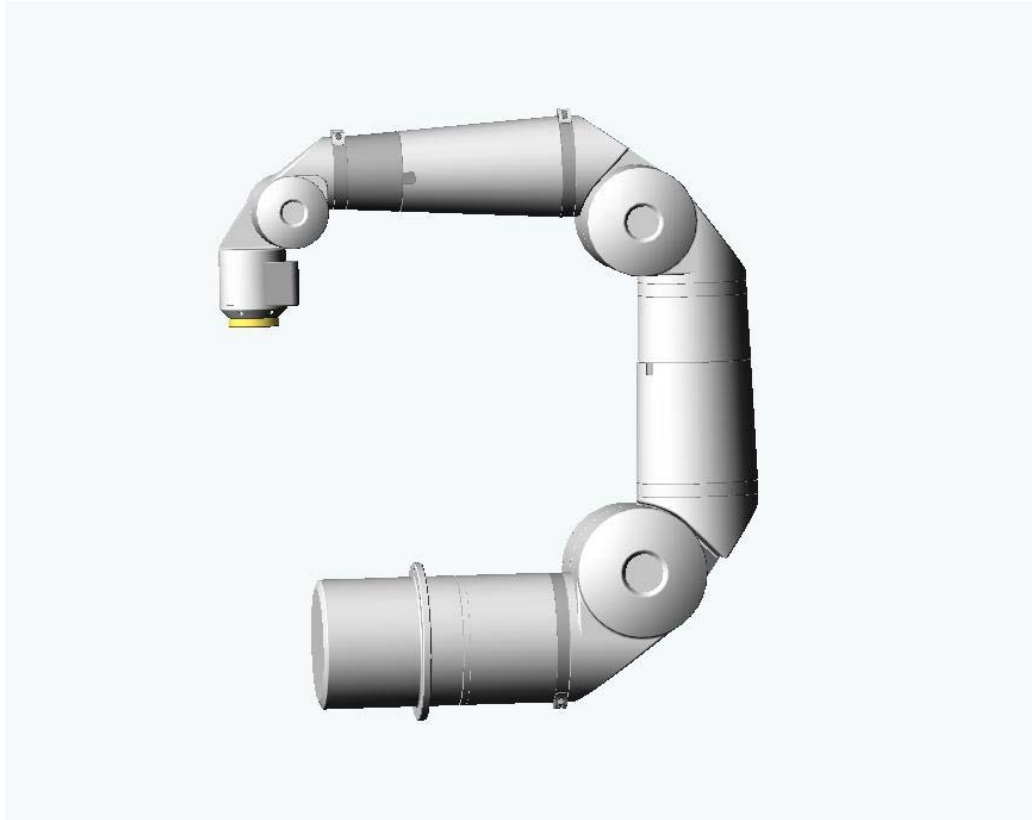


Figure 4.3. SolidWorks Model of RRC Manipulator

The Add-In DLL can be loaded from the ‘Add-Ins’ option in the Tools menu. As soon as the DLL loads, the menu item “RRC Simulation Using Phantom” is added to the SolidWorks Menu. This menu contains Dialog Simulation and Feedback Simulation and Phantom Simulation. The interface for these two modes is displayed in the figure below.

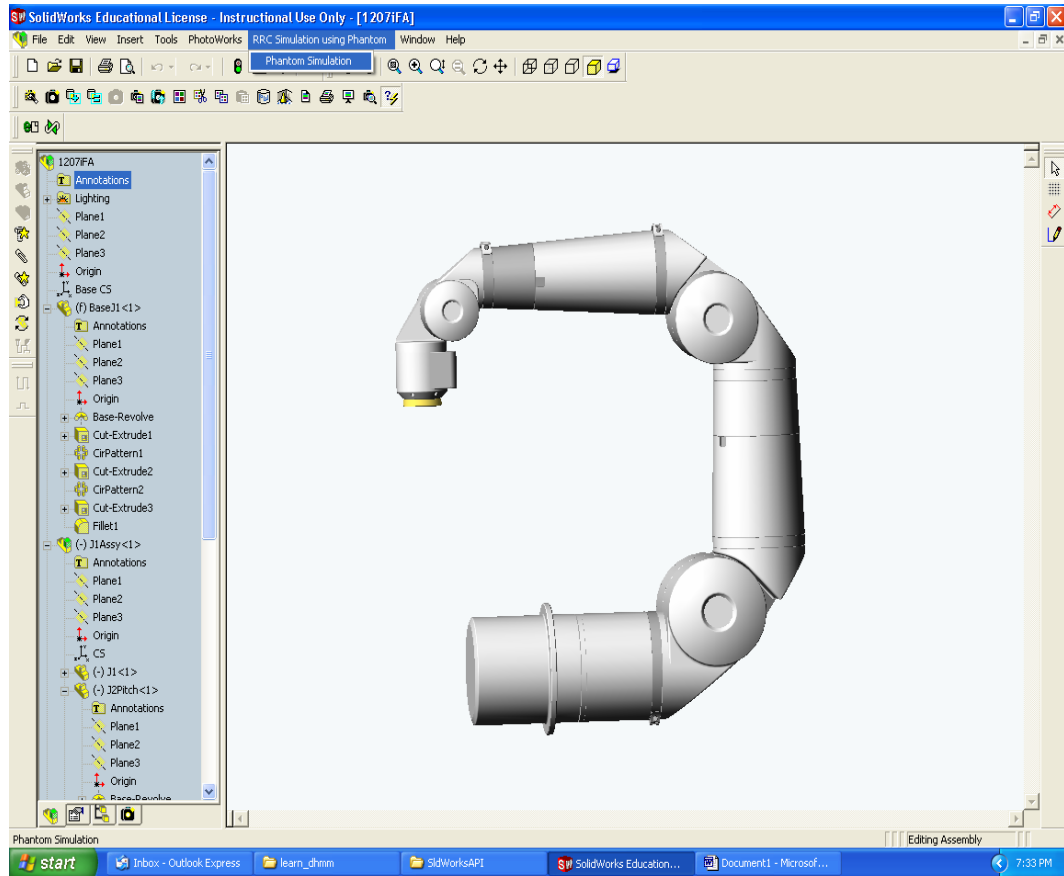


Figure 4.4 SolidWorks Simulation Screen Shot

4.4.1 Dialog Simulation

This interface is used to perform simulations of the RRC model within SolidWorks and it can also connect to the Phantom through Socket Communication. The dialog box for this simulation has the following features:

Edit Boxes J1 to J7: The values entered in these edit boxes J1 to J7 are the joint angles for the various joints of the manipulator. This framework calls its member function immediately before losing the input focus. This is done to make sure that the joint angle

entered lies within the lower and the upper limits of each joint specified in the configuration files (.CFG). The values entered here are copied on to global variables defined in SwSimGlbl.h

Button GO: The functionality of this button is present in OnButtonGo(). This function in turn makes a call to SwSimGo() where the joint offsets are added or subtracted to user specified joint angles. The current position of the manipulator joints are subtracted from the user specified angles and the joint values are passed by reference to MoveJntInc() which accepts pointer to a real array of variables. This function checks for the type of joint (prismatic or rotary) and it computes the proper rotated coordinate system about the specified axes. The corresponding transformation matrix is set to each joint in the function XtrfmAllJoints(). The final call is made to Rebuild (swUpdateMates) and also to GraphicsRedraw2() for updating the model to the required configuration.

Pose Data (position and rotation): The six-edit boxes within this group box contain the values of the Phantom's position and orientation send through a socket. A pointer to the dialog is created and the six values obtained from the network are copied into the member variables of the edit boxes added using the class wizard.

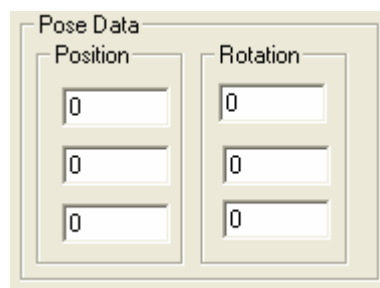


Figure 4.5. Edit Boxes for Position Data

Get Data Button: The Get Data button captures the position and orientation values of the Phantom from the 'NetGhost' Program. This does not update the values by itself.

Edit Boxes Joint 1 to Joint 7: These edit boxes get the values of the joint angles for each joint of the manipulator from the NetGhost Program. For converting the position and orientation data to joint angles the RRG Kinematix Library has been used. For each joint, the angle as well as the violation information is displayed.

Phantom Simulation		
	Angle	Violation
Joint 1	<input type="text"/>	<input type="text"/>
Joint2	<input type="text"/>	<input type="text"/>
Joint3	<input type="text"/>	<input type="text"/>
Joint4	<input type="text"/>	<input type="text"/>
Joint5	<input type="text"/>	<input type="text"/>
Joint6	<input type="text"/>	<input type="text"/>
Joint7	<input type="text"/>	<input type="text"/>

Figure 4.6. Edit Boxes for Holding Joint Angle Data

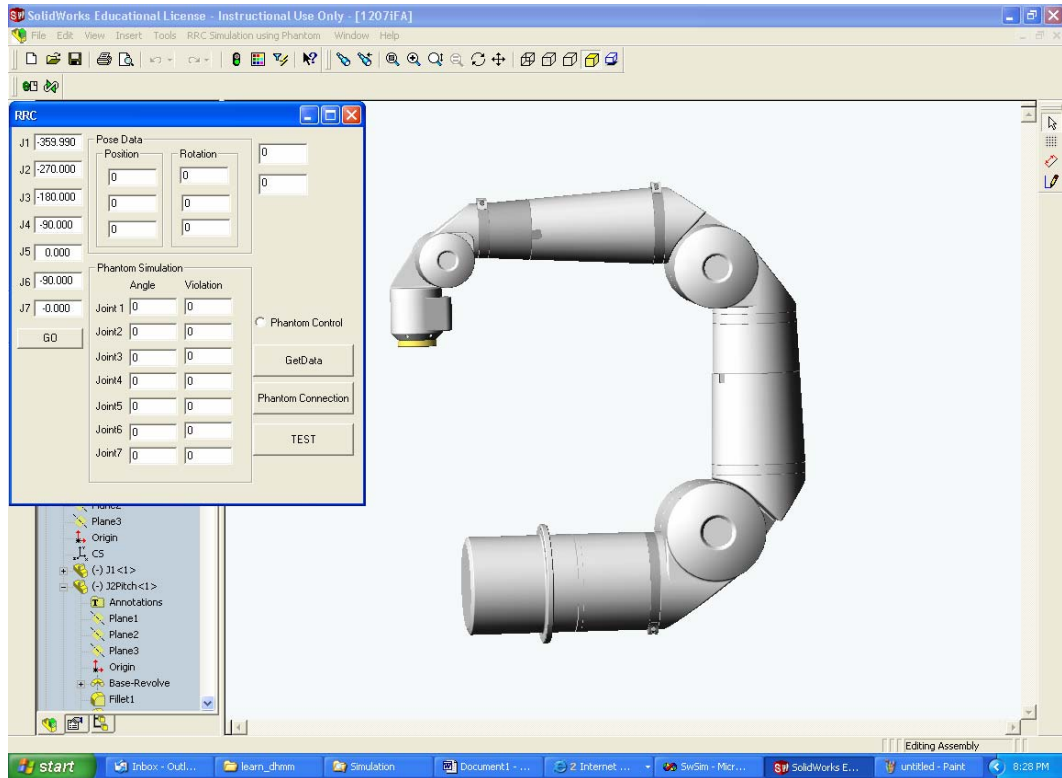


Figure 4.7. SolidWorks Screen Shot of the Phantom Simulation

4.4.2 Feedback Simulation

In this Simulation SolidWorks connects to the RRC R2 Controller to get the joint angle values and updates the SolidWorks model. The salient features of this interface are

List Feedback Box: This edit box displays the status of the connection to the RRC Controller. The messages are passed on to the SolidWorks through a CStringHandle. If the Client API handle is not created then it displays 'Connection to R2 Controller Failed. IP Socket Error : '. If the connection was successful the message 'R2 Connection Established' is displayed.

Client Edit Box: This edit box displays the current clients name. Since solidworks is connecting as a client, the message ‘SOLIDWORKS’ is displayed to the user.

R2 Host Address: This box displays the R2 host port address. For the current Simulation it displays ‘USF-ROBOT’.

R2 Connection: The functionality for the R2 Connection Button is provided in `OnButtonConnect()`. This makes a call to the `ServerConnect()` and starts the feedback process. This function also makes sure that the server is not sending the same old data. Once there is a change in the manipulator configuration, the solidworks receives those updated values to simulate the actual movements of the manipulator. The commanded robot joint positions are stored in `parJntPosCmd`.

4.5 NetGhost Interface

NetGhost was originally written in order to provide an interface between the PHANToM haptic interface and any networked server capable of accepting a network connection and processes the resulting data. NetGhost was then expanded to allow the raw input data from the PHANToM to be processed and convert into joint angle information. Numerical feedback is provided to the user in the form of raw input data. NetGhost contains a significant amount of code for managing the PHANToM, in the form of a class called “Snoop”. This name was chosen because this class passively

watches the PHANToM's position and reports this data over the network. The rest of the code is used to create an interface between the user and the snoop class is in NetGhost2Dlg class.

The data flowing from the NetGhost program to SolidWorks behaves as follows. The SolidWorks Simulation code expects only joint angle data. Since using the GHOST SDK we get the Phantoms position and orientation, these values are converted into joint angles by making use of RRG Kinematix Library.

The NetGhost Interface is shown in the figure below.

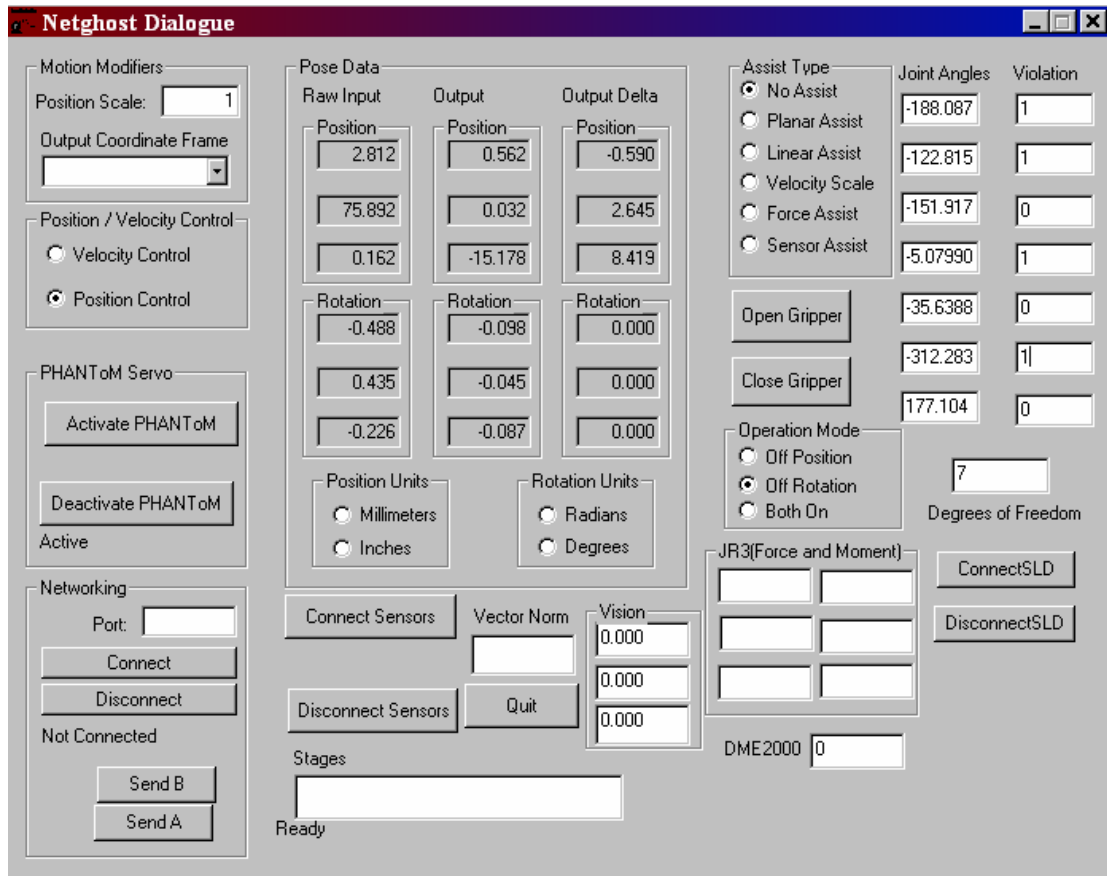


Figure 4.8 NetGhost Interface for the SolidWorks Simulation

The Snoop class relies on one function, `Snoop::calculateForceFieldForce()`, in order to perform its task. When the scene graph object is instructed to activate the PHANToM servo (using `Scene::startServoLoop()`), the `Snoop::calculateForceFieldForce()` method is called repeatedly. That is, the function is re-entrant, and it must compute a single vector and return without much delay.

Within this function, the position of the PHANToM at that particular instant is retrieved using `gstPHANToM::getPositionWC` and `gstPHANToM::getRotationAngles`, both of which are transformed as necessary using `fromWorld()`, which in this case, since the PHANToM and snoop are not separated by any `gstSeparators`, most likely does nothing. The typical use of the `calculateForceFieldForce` function inherited from `gstForceField` is to compute a force vector for the PHANToM to exert, and to return that vector. `Snoop::calculateForceFieldForce()` does this, but in this case, the return vector is always zero. Note that the servo loop on the PHANToM is highly sensitive to how long this particular function takes to complete, and taking too long will cause the servos to be disabled as a safety feature and an error to be issued. Particularly devastating are file I/O and network use, though this is sometimes unavoidable. There may be some solutions to this, including making the SDK tolerate long servo update times. If NetGhost stops the servo due to an error, it can simply be reactivated.

`CNetghost2Dlg::OnDataChange()` is where the NetGhost program does most of its real work. This function takes the raw position and rotation values from the PHANToM (note

that the input units are millimeters and radians), and then sends the result over the network to SolidWorks.

Finally, the joint angle data are only sent out at about 5Hz. This rate is determined by the update rate of the SolidWorks, which has difficulty running faster. Note that at present, the connection between the two is somewhat fragile. It may be possible for one side to close its socket without the other noticing quickly. This function is called within a very tight event loop, and must be allowed to exit within about 1 millisecond, or else Ghost will disable PHANToM servos automatically.

Here is a rough schematic of how this code is organized:

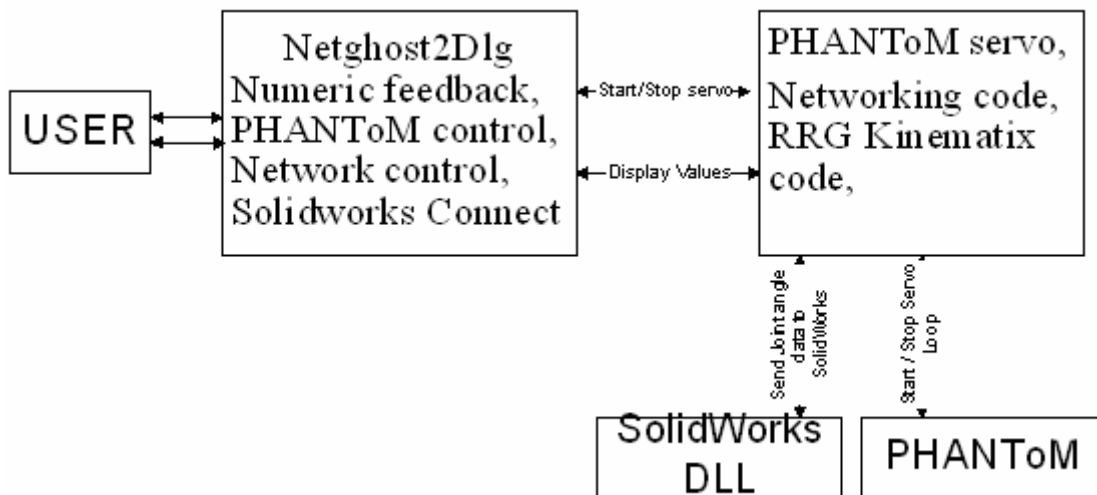


Figure 4.9 Block Diagram for the Code

4.6 RRG Kinematix

RRG Kinematix is a C/C++ library for generalized robot kinematics. It can be used to perform forward and inverse kinematics of any serial chain robot, with any number of Degrees of Freedom. All information about the robot is included in the DH parameter file that the user has to supply for their own robot. Also, the user can create and manipulate multiple robots at the same time. The library is supplied as a DLL file for Win95/NT. The LIB file supplied is to be included as one of the project files, while compiling programs made to use the DLL.

The usage is very simple, because the forward and inverse kinematics is performed by calling simple functions like 'GetHandPosition()' and 'GetJointPosition()' respectively. For those who like to fine tune the kinematics routines to their own needs, functions are provided that will adjust the parameters of the inverse kinematics routine. The functions added to the NetGhost Program are:

- *WINAPI Create Robot* creates a robot object and returns a handle for it. This is required for all function calls.
- *GetDOF* returns the Degrees of Freedom (DOF) of a robot.

- *GetHandJacobian* returns the Jacobian matrix of the robot. The user can specify the joint angle state based on which the Jacobian is to be calculated. If the user does not supply joint angles, the current joint angle state is used.
- *SetJointLimits*: The user can set the joints limits for the robot using this function. The joints limits are specified in a limits file, which is in a specific format.
- *SetInverseType*: Using this function the user can specify which inverse type is to be used in the inverse kinematics calculations. The two choices available are Pseudo Inverse and Avoid Limits Pseudo Inverse. The first one is the default and the second one can be used only if the robot is spatial and redundant.
- *Destroy Robot* is called at the end of the program and it destroys the robot object created using the create robot function.
- *Get Joint Position* performs the inverse kinematics. It takes the robot handle, end effector position and an array of double for holding the joint solutions.

All error return values have been prefixed with 'RRG_' in order to avoid namespace clashes. For example, FAILED will now be RRG_FAILED and BAD_DHFILE is now RRG_BAD_DATA_FILE.

The following section gives a description of the working steps for the PHANToM/SolidWorks project. The underlying goal of the project is to use the PHANToM as an input device that can drive the SolidWorks model. The way the system works is as follows:

The user will sit on the PHANToM PC and can interact with the PHANToM. On the same system lies the SolidWorks. The user first starts the SolidWorks and opens the SolidWorks model of the RRC Manipulator. Now the DLL has to be loaded in the SolidWorks menu. The dll can be loaded from the Tools → Add-In option of the SolidWorks menu. After the DLL is loaded the Menu Item “RRC Simulation Using Phantom” is displayed. Under this select “Phantom Simulation”. Once this is selected the Network connection is established from the SolidWorks side. Now connection has to be established from PHANToM’s side. To do this, execute the NetGhost program and click on ‘Connect SolidWorks’. Now the socket connection is established from both sides and the Phantom’s position and orientation data is ready be transferred to SolidWorks model. To accomplish this click on ‘Activate Phantom’. The status of the Phantoms connection is displayed in a text box that says ‘Phantom Activated’. Now you can see the values continuously updating in both the windows. The corresponding joint angle information is also seen updating and the manipulator moves accordingly.

Chapter 5: Assistance Concept and Hidden Markov Model Development

5.1 Introduction

The underlying idea of the assistance function concept is the generalization of position and velocity mappings between the master and slave manipulators of a telerobotic system. This concept was conceived as a general method for introducing computer assistance in task execution without overriding an operator's command to the manipulator. In this case we have used this assistance strategy to improve the dexterity of persons with disabilities. These assistances are designed based on the information of the environment obtained from different sensors.

In the next case we have used the Hidden Markov Model to successfully recognize the users motion intention and classified human actions and applied appropriate assistance when needed. This approach combines the environment information and also humans motion intent before applying appropriate assistance. Further we have used the Hidden Markov Model to select the most likely human performance and selected the best action sequence from all the previously measured action data and modeled the skill. That is if the performance is the one that the operator most likely performs, then we claim that the performance is a good representation of the skill. We have developed a test bed and

used this skill for skill learning applications. This method can allow a person with disability to learn human skill in certain tasks and also improves motion performance.

Human skill is the ability to apply the knowledge and experience earned in performing different tasks. Modeling human skill and transferring the skill to persons with disabilities has been an objective of research for many years. This problem is very important for developing the motion/manipulation skills of persons with disabilities.

For a disabled person task level learning emphasizes on achieving a certain goal by practice. Skill can be gained incrementally through learning and practicing. This learning mechanism can allow disabled persons to learn human skills and perform tasks more reliably. For the learning trajectory presented here the learning procedure is done in Cartesian space. A relatively complex task for persons with disabilities is used to be modeled by HMM.

In the following sections a detailed analysis of the several forms of assistance functions followed by the Hidden Markov Model approach to skill learning and users intention recognition has been discussed.

5.2 Assist Function Development for Office Environment

The assist function concept presented here is three-dimensional application solely for the purpose of testing in 3D space. It is assumed that the accuracy of the information

about the environment is known and fixed and can be gathered in negligible time. In these applications the operator is given the task of approaching and picking up the desired object and moving it to the location he wants. To optimize the performance of the task it is desirable to scale down all the unnecessary translations to minimize the probability of a hard impact with the shelf's and the walls. In real world the information about the distance is derived from the end-effector mounted laser range finder for determining the proper assist function. Here the position data obtained from the Phantom is used as sensor data so as to avoid the contact of the slave with the surfaces. In this way the information about the surfaces could be integrated. For $z_1 < z < z_2$ it would be desired to have different velocities for each range so that the slave velocities are scaled up or down.

The task to be performed involved an approach of the end-effector to pick the object and no instructions were given to the operator. Since the phantoms position is used as sensor data, and the slave will be moving to complete the task, while the availability of the phantoms data will vary depending on the progress of the task. The task is divided into four stages of available information. The four stages scale the input velocity unique to the scaling stage for optimal task operation. At the start of the task, the slave and the phantoms position are the same and this is referred to as stage zero and the approach task for grabbing the object corresponds to stage one. The second stage is the stage where the user grabs the object and moves it out of the shelf. The third stage is the stage of moving the object in free space and once the object is almost close to the destination, the fourth stage starts where the object has to be placed back into the shelf.

The test bed used for the simulation experiments was described in Chapter 6. The system interfaced with a virtual environment in order to simulate the execution of the chosen task. A mapping between the PHANToM™ inputs and the movements and forces performed in the virtual environment was simulated. It was necessary to develop and simulate such a mapping because the restricted range of motion and forces for a person with disabilities are limited and inadequate for performance of the task. A graphical model of the environment was developed using the GHOST® software development kit and controlled by use of the PHANToM™.

The forms of assistance employed in the experiments were position assist functions, velocity assist functions, planar assistance functions and attractive force feedback gravity wells. The approach used, in these experiments, was to impose the constraints on the master's side and assign the commanded positions to the slave. This was accomplished through the use of GHOST functions.

These tests provide information about the particular abilities of an individual, as well as, information about the specific disabilities that a person might present. In other words, it provides a method to assess people with disabilities based on their manipulative skills. A discussion of the various forms of Assistance functions follows.

5.2.1 Position Assistance Function

In this type of assistance, the master's position is scaled and transformed to obtain the desired slave's commanded position.

5.2.1.1 Linear Assistance Function

This assistance function constrains the motion of the slave to a desired linear trajectory. The phantom's position is read into the variable `phantomTrans` of the type `gstPoint`. The values in this variable keep updating every instance of time. This variable can be used for getting information about the environment and can be used as sensor data giving us the location of the various objects at different instances of time.

A simple approach to this assistance is to constrain the master's movement along a line and assign the commanded position to the slave. This is accomplished by the function `setline(gstPoint, gstVector)`, which is provided by GHOST. This function accepts a point and a vector to define the constraint line.

The master velocity in the base frame is obtained from the Phantom as:

$$V_{\text{master}} = \text{myPhantom} \rightarrow \text{getVelocity}() \quad 5.1$$

The master velocity in the slave's frame is assigned to the slave velocity as:

$$V_{\text{slave}} = R_C^T \cdot V_{\text{master}} \quad 5.2$$

The slave's scaled velocity is obtained as:

$$V_{\text{ScaledSlave}} = \text{scale_matrix} \cdot V_{\text{slave}} \quad 5.3$$

The X, Y and Z directions are increased by a scale-factor in the range, $0 < \text{scale_factor} < 1$. The commanded slave position in each coordinate, Slave_pos_x, Slave_pos_y and Slave_pos_z, are calculated by discrete integration using the previous position, Previous_pos_x, Previous_pos_y and Previous_pos_z, along with the velocity of the slave, Vslave_x, Vslave_y and Vslave_z, in each direction. These calculations are shown in 5.5 by

$$\begin{aligned} \text{Slave_pos_x} &= \text{Vslave_x} \cdot \Delta T + \text{Previous_pos_x}; \\ \text{Slave_pos_y} &= \text{Vslave_y} \cdot \Delta T + \text{Previous_pos_y}; \\ \text{Slave_pos_z} &= \text{Vslave_z} \cdot \Delta T + \text{Previous_pos_z}; \end{aligned} \quad 5.5$$

where ΔT is the sampling time.

5.2.1.2 Planar Assistance Function

In this form of assistance the motion of the slave manipulator is constrained to a desired plane. This particular function is useful especially if the operator suffers from some form of spasticity or tremor. Once again a simple approach employed which imposes a constraint on the master's side and assigned the commanded positions to the slave. This was accomplished through the use of the function

$$\text{MasterplaneConstraint} \rightarrow \text{setPlane}(\text{gstPlane}(0,0,1,0)),$$

which constraints the master's motion to an X-Y plane.

This approach is more useful in the case of a limited or impaired-motion user whose movements cannot reach the plane in which the task needs to be executed.

The velocity in the base frame is obtained from the Phantom as:

$$V_{\text{master}} = \text{myPhantom} \rightarrow \text{getVelocity}(); \quad 5.6$$

The velocity in the slave constraint frame is:

$$V_{\text{slave}} = R_C^T \cdot V_{\text{master}} \quad 5.7$$

Where R_C^T is obtained from (5.9).

The slave's scaled velocity is obtained as:

$$V_{\text{ScaledSlave}} = \text{scale_matrix} \cdot V_{\text{slave}} \quad 5.8$$

The commanded position, Slave_pos_x , Slave_pos_y and Slave_pos_z , are calculated by discrete integration using the previous position, Previous_pos , along with the velocity of the slave, V_{slave} , as follows

$$\begin{aligned} \text{Slave_pos_x} &= V_{\text{slave_x}} \cdot \Delta T + \text{Previous_pos_x}; \\ \text{Slave_pos_y} &= V_{\text{slave_y}} \cdot \Delta T + \text{Previous_pos_y}; \\ \text{Slave_pos_z} &= V_{\text{slave_z}} \cdot \Delta T + \text{Previous_pos_z}; \end{aligned} \quad 5.9$$

where ΔT is the sampling time.

5.2.1.3 Curve Trajectory Assistance

During this task the user's movements, along a straight line, are mapped onto a curved trajectory for obstacle avoidance purposes. In this way a limited range of motions

provided by the user with disabilities is translated into more complex motions on the slave side. The location of the Master is obtained by

$$\begin{aligned} \text{phantomXform} &= \text{myPhantom} \rightarrow \text{getCumulativeTransformMatrix}(); \\ \text{phantomTrans} &= \text{phantomXform} \cdot \text{getTranslation}(). \end{aligned} \quad 5.10$$

Once the cylindrical object is grasped, the position on the X-direction is determined by

$$\begin{aligned} \text{pos } x &= \text{phantomTrans} . \text{x}() \\ \text{pos } y &= \text{phantomTrans} . \text{y}() \end{aligned} \quad 5.11$$

The Z-coordinate is assigned according to the formula

$$\text{Pos } z = \left| \sqrt{w_s^2 - \text{pos } x^2} \right| \quad 5.12$$

where *pos x* is the master's current x position and *w* is the initial point.

5.2.2 Velocity Assistance Function

With this type of assistance function the mapping between the master and slave is based on velocities. During a particular task, this strategy is used to provide assistance in approaching the goal. Thus, the velocity scaling used varies according to whether the motion in a particular direction is serving to further the desired effect of the motion. With approach assistance, the velocity is scaled up if the motion is reducing the distance

between the current position of the manipulator and goal position. Otherwise, the velocity is scaled down.

Two different approaches were used to implement the scaling factor. In one case, the user was asked to enter a scaling factor between 0 and 1. The scale-factor introduced was used to scale up the velocity in the directions of motions in which the goal and end-effector were located and to reduce the velocity in the direction away from the goal.

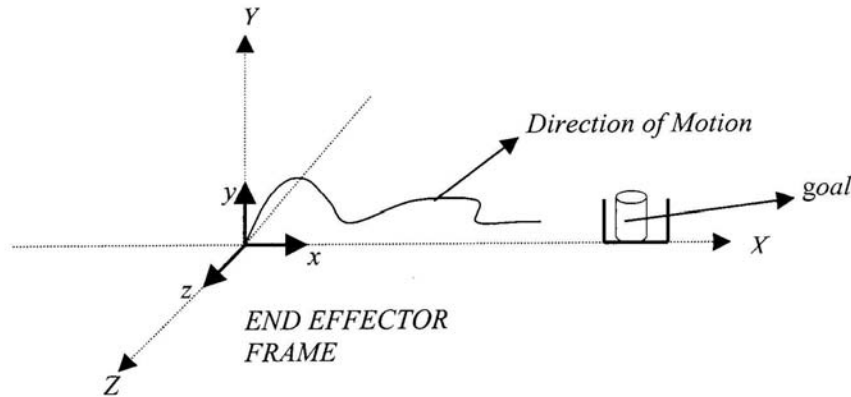


Figure 5.1 Slave's Desired Direction of Motion

The master velocity is obtained by

$$V_{\text{master}} = \text{myPhantom} \rightarrow \text{getVelocity}()$$

and its components are scaled and assigned to the slave manipulator by

$$\begin{aligned} V_{\text{slave}_x} &= (1 + \text{scale_factor}) \cdot V_{\text{master}_x}; \\ V_{\text{slave}_y} &= (1 + \text{scale_factor}) \cdot V_{\text{master}_y}; \\ V_{\text{slave}_z} &= (1 + \text{scale_factor}) \cdot V_{\text{master}_z} \quad (\text{for } z < 0) \\ &= (\text{scale_factor}) \cdot V_{\text{master}_z} \quad (\text{for } z > 0) \end{aligned} \tag{5.13}$$

In the second case, the scaling factor is increased or decreased proportionally to the location of the end-effector with respect to the goal and the workspace limits.

The scaling factor depends on the direction of travel towards the point *goal1* or *goal2*. The scaling factor depends on the task being executed and the direction of travel.

Two operational ranges around each goal ($[goal1 - 5, goal1 + 5]$, and $[goal2 - 5, goal2 + 5]$) were established in which there was no more scaling of velocities between the Master and Slave.

Given p_i , the position of the end-effector in the base frame where $i = x, y$ or z , *goal1*, the goal 1 point in the corresponding direction, and *previous_p_i*, the position of the end-effector in the base frame taken during the previous sampling time.

If $\| goal1 - p_i \| - \| goal1 - previous_p_i \| \leq 0$ and $(goal1 - 5 - p_i) \geq 0$, the scaling factor S_i is given by:

$$S_i = \left(2 - \frac{p_i}{(goal1 - 5)} \right) \quad 5.14$$

If the operator travels towards the goal 1 and misses it, the scaling factor is reduced and dropped to zero as shown in equation 5.30:

If $(p_i - (goal1 + 5)) \geq 0$ then,

$$S_i = 1 - \frac{p_i - (goal1 + 5)}{w_i - (goal1 + 5)} \quad 5.15$$

If the operator travels from goal1 to goal2 and the cylindrical piece has been grabbed, i.e.,

If $\|goal2 - p_i\| - \|goal2 - previous_p_i\| \leq 0$, the scaling factor S_i is given by:

$$S_i = -\left[1 - \frac{p_i - (goal1 + 5)}{w_i - (goal1 + 5)}\right] \quad \text{When } p_i \geq goal1 + 5, \text{ or} \quad 5.16$$

$$S_i = -\left[1 + \frac{(goal1 - 5) - p_i}{(goal1 - 5)}\right] \quad \text{When } 0 \leq p_i \leq goal1 - 5$$

Finally, as the slave approaches the goal 2, the scaling factor is given by:

$$S_i = -\left(2 - \frac{p_i}{(goal2 + 5)}\right) \quad \text{When } goal2 + 5 \leq p_i \leq 0, \text{ or} \quad 5.17$$

$$S_i = -\left[1 - \frac{p_i - (goal2 - 5)}{w_i - (goal2 - 5)}\right] \quad \text{When } p_i \leq goal2 - 5$$

The velocity of the master is obtained using the function

$$V_{master} = myPhantom \rightarrow getVelocity(),$$

with its coordinates given by

$$V_{master_i} ; i = x, y \text{ or } z.$$

The velocity is scaled and assigned to the slave as

$$V_{slave_i} = S_i \cdot V_{master_i}, \text{ where } i = x, y \text{ or } z. \quad 5.18$$

The commanded positions to be sent to the slave are calculated from

$$Slave_pos_i = V_{slave_i} \cdot \Delta T + previous_p_i, \quad 5.19$$

where ΔT is the sampling time, $Slave_pos_i$ is the current slave's position in the i -direction and $previous_p_i$ is the previous slave's position in the i -direction.

5.2.3 Force Assistance Function

The purpose for this kind of assistance is to augment the user's dexterity by imposing some constraints based on attractive or repulsive potential fields. These attractive or repulsive potential fields are virtual constraints that are implemented in the master's control in order to help the operator carry out complex tasks such as staying on a perfect line or moving away from the undesired or repulsive zones.

The current position in the base frame of the Master device is obtained using the function

$$pos = \text{fromWorld}(\text{phantom} \rightarrow \text{getPosition_WC}())$$

and

$$pos = (\text{pos. } x(), \text{pos. } y(), \text{pos. } z()). \quad 5.20$$

Then, the vector V from the starting point to the Master's position is calculated as

$$\vec{V} = \langle x_m - x_1, y_m - y_1, z_m - z_1 \rangle. \quad 5.21$$

Also, the vector between the starting and final points is:

$$\vec{B} = \langle x_f - x_1, y_f - y_1, z_f - z_1 \rangle. \quad 5.22$$

The projection of the operator's Cartesian position on the desired trajectory is obtained through use of the dot product as:

$$\vec{V}_{projection} = \frac{\vec{V} \cdot \vec{B}}{\|\vec{B}\|} \vec{B} \quad 5.23$$

Then the curve or control surface is surrounded by an attractive potential field whose amplitude increases with the distance between the end-effector and the projected point. The force vector is calculated as

$$\vec{F} = \vec{V} - \vec{V}_{Projection} \quad 5.24$$

Then, the corresponding attractive force on the haptic's device end-effector is:

$$\vec{F}_m^C = k \cdot \vec{F} \quad 5.25$$

In this case, the operator will easily move on the unconstraint directions, but will have to fight high torques on its master device to go away from it. In the move task, the constraints were imposed to help the operator move as follows:

$x = F_m \cdot pos \cdot x()$; which corresponds to the Phantom's position on the x-axis,

$y = F_m \cdot y()$; and

$z = F_m \cdot z()$; corresponding to the constraint attractive force on the Y-axis and the Z-axis respectively.

Thus, the attractive force vector to be applied on the haptic device's end-effector is obtained as the difference between the current's operator position and the new coordinate (x, y, z) imposed by the constraint force.

$$\vec{F}_m^C = \langle pos \cdot x() - x, pos \cdot y() - y, pos \cdot z() - z \rangle. \quad 5.26$$

Subsequently, the slave manipulator will follow the master device and move according to the predefined constraints. The method in which the force is sent back to the Phantom device during all iterations of the servo loop is force feedback.

5.3 Hidden Markov Model Based Skill Learning and Motion Recognition

Rich in mathematical structure, Hidden Markov Model (HMM) is a trainable statistical model, with two appealing features: 1: no prior assumptions are made about the statistical distribution of data to be analyzed, and 2: a degree of sequential structure can be encoded by the Hidden Markov Model. In robotics field, Yang and Xu applied HMM to teleoperation skill learning and gesture recognition, thus eliminating sluggish motion and correcting a motion command that the operator may mistakenly generate.

The Hidden Markov Model consists of the following stages.

5.3.1 Data Preprocessing

The velocity and position of the Phantom are sampled at 1000Hz rate. In order to visualize better, we constrained the movement in a plane. The data is represented as:

$$\begin{aligned} P &= \{P_x, P_y\} \\ V &= \{V_x, V_y\} \end{aligned} \tag{5.27}$$

where P_x, P_y, V_x, V_y are :

$$\left. \begin{aligned} P_x &= [p_{1,x} \quad p_{2,x} \quad \dots \quad p_{n,x}]^T \\ P_y &= [p_{1,y} \quad p_{2,y} \quad \dots \quad p_{n,y}]^T \\ V_x &= [v_{1,x} \quad v_{2,x} \quad \dots \quad v_{n,x}]^T \\ V_y &= [v_{1,y} \quad v_{2,y} \quad \dots \quad v_{n,y}]^T \end{aligned} \right\} \quad 5.28$$

where n number of sampling . Since each vector is similar, we just demonstrate the data processing of one of them, says P_x . From the sampled data, a 16-point width window with 50% overlap is used. Prior to spectral conversion, in order to minimize spectral leakage, a 16-point hamming window is first used to filter each frame. Then FFT analysis is applied for every windowed data. The FFT transform maps a k-length vector P_x to a k-length complex vector $Z=[z_1, z_2, \dots, z_{16}]$. Since FFT result is symmetrical about the half FFT frequency, we just have to use half of them. So we selected the FFT coefficients from 1 to 9. We call this vector as $M=[m_1, m_2, \dots, m_9]$.

$$m_i = \|z_i\| \quad 1 < i < 9$$

Using this symmetry property, the data computation can be reduced without decreasing processing accuracy. Finally, a set of 16-dimensional velocity vectors is mapped to a set of 9-dimensional spectral power vectors. We train the VQ codebook by those vectors and the codebook was produced by LBG algorithm. The LBG VQ(vector quantization) technique maps these 9-dimensinal vectors into a finite set of vectors $Y = \{y_i: i = 1, 2, \dots, M\}$, where M is the length of the codebook(it is determined 256 in this case). Each vector

y_i is called a code vector or a codeword and the set of all the code words is called a *codebook*. Associated with each codeword, y_i , is a nearest neighbor region called *Voronoi* region, and it is defined by:

$$V_i = \{x \in R^k: \|x-y_i\| \leq \|x-y_j\|, \text{ for all } j \neq i\} \quad 5.29$$

The 256 9-dimensional vectors in the codebook are 256 symbols in the output probability distribution functions for discrete HMM. Similarly, a codebook for the velocity component v_o vector and the 256 symbols are also obtained in the same way.

5.3.2 Hidden Markov Model Construction

A HMM consists of:

- A set of N states $S=\{S_1, S_2, \dots, S_N\}$;
- A set of possible observations $V=\{v_1, v_2, \dots, v_M\}$;
- A state transition probability matrix $A=\{a_{ij}\}$;
- Observation probability matrix $B=\{b_j(k)\}$;
- Initial state distribution $\pi=\{\pi_i\}$.

Each state is characterized by two sets of probabilities: a transition probability, and an output probability. The three model using problems are:

- Likelihood of a sequence of observations given a model;
- The most likely sequence of states the model traverses given an observation;

- Adjusting the model parameters to best account for a set of observation sequences (training).

We are trying to model the moving along labyrinth task skill using HMM. As stated in section 5.3.1, we have converted the measurements into finite symbols. The symbols from vector quantization represent the meaning of feature. In order to visualize the virtual therapist to the user more effectively, the trajectory of the movements is chosen as the skill for learning. Since we only consider the movement in a plane, P_x , P_y are used for skill learning, which means that HMM has 2 observable symbols at each time t . Obviously, they are two independent vector. So we design a 5-state, two-dimensional HMM. The observations are sequences of symbols obtained in LBG VQ algorithm. The answer of the third problem is to set the parameters of the model (A , B , π) to maximize the probability that the model matches with the observation sequence. The purpose of learning is to obtain the parameters of the model from observation sequences. For a given model, let $\alpha_{t+1}(j)$ represent the likelihood of observing vector o_1 to o_t and being in state j at time t . The partial likelihood of each model can be computed by the forward procedure:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \quad \begin{array}{l} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{array} \quad 5.30$$

where

$$\alpha_1(i) = \pi_i b_i(O_1) \quad 1 \leq i \leq N$$

The probability of the partial observation sequence is:

$$P(O | M) = \sum_{i=1}^N \alpha_T(i)$$

5.31

Once the skill model has been built, the model which produces the largest likelihood $P(O|M)$ given an observation is the learned skill.

Since we have 2-dimensional observations, a 2-dimensional HMM is used. A multidimensional HMM is an HMM that has more than one observable symbol at each time t for each state. In this case, the A matrix structure of a multidimensional HMM is the same as a one-dimensional HMM. But a multidimensional HMM has more B matrix, each one for characterizing stochastic distribution of each dimensional signal.

5.3.3 Skill Learning Approach and Tremor Suppression Filter

We have used the HMM to model the translation skill along a maze which can be used as a virtual therapist in therapy training. We employed a 5 state left-right HMM to model the skill. The moving task is confined to the XY plane and two symbols are observable at each instance of time. The two dimensional data is pre-processed and two independent vectors of observable symbols are obtained for each task execution. The training set is produced by repeating the task 12 times. The prior matrix A (5x5 matrix) and the observability matrix B (256x5 matrix) are initialized. The initialization and the updating calculation are the same for π and A for a 2D and 1D HMM. The forward algorithm was

used to score each trajectory. Based on the score results the best and the worst trajectories are obtained and in this case the best trajectory was found to be No. 7 and the worst happened to be No.6. The No.7 trajectory served as a virtual therapist in this case.

Extensive research has been done with respect to the characterization of tremorous movements produced by people with disabilities. “Tremor is an involuntary, rhythmic, and uncontrollable movement of the body that appears to be an oscillation superimposed on the voluntary movements”.

Physiological tremor (normal tremor) does not really affect the daily activities of a person. It exhibits very small amplitudes and an energy distribution located at frequencies higher than 8Hz. Pathological tremor can severely affect the daily activities and therefore the employment opportunities of a person. This kind of tremor is concentrated at low frequencies, ranging from 2-6 Hz and its oscillation amplitudes are significantly stronger than those of physiological tremor.

5.3.4 Methods of Evaluation

The 3D position of the Phantom is sampled at 1000 Hz. The following methods of evaluation are used.

- Time taken to complete the task
- Number of collisions with the walls

- Time duration of each collision
- Impact force
- Tremor Magnitude and Frequency
- Distance ratio (R) = $\frac{d_{actual}}{d_{skill}}$

R reflects the capability of optimizing the trajectory. It represents one of the important test outcomes. The numbers of collisions represent the total number of collisions with the walls (horizontal or Vertical).

Tremor frequency is relatively invariant with respect to the direction and speed of the movement. Tremor frequency during task performance is relatively constant. However, the intensity or amplitude of the tremor may vary. In general, tremor signals are sinusoidal with a range of frequencies from 3-11Hz (cerebellar 3-5 Hz, essential 6-11Hz., Parkisonian 3-7 Hz.) whereas the intended movement range is between 0.5 to 2.5 Hz [12].

The filter algorithms were implemented off-line, when the input signal values extracted from the data stored in a file. Given the general frequency range that characterizes the pathological tremor in patients and the fact that neither the reference of the trajectory nor the noise signal are available in the particular tasks developed in this work, low-pass filter were tested using Matlab / Simulink. The filters proved successful in removing the unwanted tremor noise from the signals of interest.

5.3.5 Tremor Filter Design and Matlab/Simulink Code

Tremor information is extracted by applying a high pass filter which has a cut-off frequency of $F_{\max}/10$. The tremor frequency can be obtained through discrete Fourier transform.

fc : cut-off frequency= 2Hz,

T : Sampling rate=0.001s.

N=512; where N is the number of points for FFT

Using the coordinates of the end points of the path boundary, the boundary of the maze path is drawn to give a clear picture of the collisions with the walls of the maze.

The coordinates are given by:

```
path_x1=[18 18 -18 -18 -72 -72 18 18 54 54 72 72 54 54 36 36];
```

```
path_y1=[-4 18 18 54 54 -72 -72 -54 -54 -18 -18 36 36 54 54 90];
```

```
path_x2=[18 -36 -36 -54 -54 0 0 36 36 54 54 36 36 18 18];
```

```
path_y2=[-4 -4 36 36 -54 -54 -30 -30 0 0 18 18 40 40 90];
```

The Data file 'Skill_Data.txt' is the most likely human performance obtained from the HMM skill learning algorithm described above. This file is read by the matlab and the trajectory is displayed in red.

```
skill = load ('Skill_Data.txt');
```

The data read from the file is copied into the arrays for use by the plot command for plotting the trajectory.

```
skill_x=skill(:,1);  
skill_y=skill(:,2);
```

The data obtained from a person with disability is read into a data file and its being read by matlab for plotting the trajectory within the maze.

```
maze = load('R6.txt');  
Px = maze(:,1);  
Py = maze(:,2);  
Vx = maze(:,4);  
Vy = maze(:,5);
```

Here Px, Py, Vx, Vy represent the translations and the velocities in the X and the Y Direction of the ball pointer.

The number of collisions which the ball pointer makes with the walls is read into the array coll. These are obtained from the program using the

```
int numColl = myPhantom->getNumCollisionObjs();
```

In the matlab program these are read into the array using the equation

```
coll = maze(:,8);
```

The force with which the ball pointer hits the walls are calculated from

```
PHANToMForce = myPhantom->getReactionForce_WC();
```

```
fx = PHANToMForce.x();
```

```
fy = PHANToMForce.y();
```

The arrays Fx and Fy read the sixth and seventh columns from the data file which are the forces in the X and the Y directions.

```
Fx=maze(:,6);Fy=maze(:,7);
```

The magnitude of the force is calculated from the equation

```
F_mag=sqrt(Fx.*Fx+Fy.*Fy);
```

A high pass filter and low pass filter are used to analyze the tremor information, including tremor magnitude and frequency. The code for the tremor filter is:

```
[b_hpf,a_hpf] = butter_hpf(fc,T);
```

```
[b_lpf,a_lpf] = butter_lpf(fc,T);
```

```
Px_lpf = filter(b_lpf,a_lpf,Px);
```

```
Py_lpf = filter(b_lpf,a_lpf,Py);
```

```
Vx_lpf = filter(b_lpf,a_lpf,Vx);
```

```
Vy_lpf = filter(b_lpf,a_lpf,Vy);
```

Where Px_lpf , Py_lpf , Vx_lpf and Vy_lpf represent the translation and velocity after tremor elimination.

```
Px_tremor = filter(b_hpf,a_hpf,Px);
Py_tremor = filter(b_hpf,a_hpf,Py);
Vx_tremor = filter(b_hpf,a_hpf,Vx);
Vy_tremor = filter(b_hpf,a_hpf,Vy);
```

Where Px_tremor, Py_tremor, Vx_tremor and Vy_tremor represent the translation and velocity tremor information.

The filter magnitude frequency response is plotted using

```
[Hh,Wh]=freqz(b_hpf,a_hpf,N,1/T);
subplot(211),plot(Wh,abs(Hh));
xlabel('frequency,Hz'),
title('HPF Magnitude Response(fc=3Hz,fs=1000Hz)');
[HI,WI]=freqz(b_lpf,a_lpf,N,1/T);
subplot(212),plot(WI,abs(HI));
xlabel('frequency,Hz'),
title('LPF Magnitude Response(fc=3Hz,fs=1000Hz)');
```

The following lines of code are for plotting the maze path as well as the trajectories of human skill and for the disabled person also.

```
plot(Px, Py, skill_x, skill_y,'r--', path_x1, path_y1, 'k-', path_x2, path_y2, 'k-'),
axis([-110 110 -110 110]);
legend('Actual Movement','Moving Skill');
```

```

xlabel('X-axis(mm)'),ylabel('Y-axis(mm)');
title('translation with tremor');

pause,figure

plot(Px_lpf,Py_lpf,skill_x,skill_y,'r--',path_x1,path_y1,'k-',path_x2,path_y2,'k-'),
axis([-110 110 -110 110]);

legend('Actual Movement','Moving Skill');

xlabel('X-axis(mm)'),ylabel('Y-axis(mm)');
title('translation after tremor elimination');

```

The translation tremor analysis is plotted using

```

pause,figure

F=[-N/2:N/2-1]/N;

subplot(221),plot(abs(Px_tremor));

axis([1 20000 0 15]);

title('x-translation tremor');

xlabel('number of sample'),ylabel('magnitude(mm)');

Px_C=abs(fft(abs(Px_tremor),N));

Px_C=fftshift(Px_C);

subplot(222),plot(F*1000,Px_C/N);

xlabel('frequency/fs'),title('x-tran tremor FFT');

subplot(223),plot(abs(Py_tremor));

axis([1 20000 0 15]);

```

```

title('y-translation tremor');
xlabel('number of sample'),ylabel('magnitude(mm)');
Py_C=abs(fft(abs(Py_tremor),N));
Py_C=fftshift(Py_C);
subplot(224),plot(F*1000,Py_C/N);
xlabel('frequency/fs'),title('y-translation tremor FFT');

```

The velocity tremor analysis is plotted as

```

pause,figure
subplot(221),plot(abs(Vx_tremor));
axis([1 20000 0 15]);
title('x-velocity trmor');
xlabel('number of sample'),ylabel('magnitude(mm/s)');
Vx_C=abs(fft(abs(Vx_tremor),N));
Vx_C=fftshift(Vx_C);
subplot(222),plot(F*1000,Vx_C/N);
xlabel('frequency/fs'),title('x-velocity tremor FFT');

```

```

subplot(223),plot(abs(Py_tremor));
axis([1 20000 0 15]);
title('y-velocity trmor');
xlabel('number of sample'),ylabel('magnitude(mm)');
Vy_C=abs(fft(abs(Vy_tremor),N));

```

```

Vy_C=fftshift(Vy_C);
subplot(224),plot(F*1000,Vy_C/N);
xlabel('frequency/fs'),title('y-translation tremor FFT');

```

To find the number of collisions with the walls a separate algorithm was written which maintains the collision number in a variable count and the time array is the impact time for each collision.

```

[count,time]=collisions(coll);
% co is the collision information,
% count is the number of collisions
% time is array of the time period of each collision
pause,figure
L=length(coll);
index=1:L;
subplot(311),plot(index,coll),
axis([1 L+200 0 2]);
xlabel('The number of sample'),ylabel('collision');
title('collision detection');
time = time*T;
The time interval equals the product of the sampling number and sampling rate.
subplot(312),stem(time),
xlabel('The number of collision'),ylabel('time interval(s)');

```

Plots of reaction force during moving are plotted as

```
subplot(313),plot(F_mag),  
xlabel('The number of sample'),ylabel('Reaction Force(N)');
```

5.3.6 Motion Recognition and Design of Assistances

After the users intention has been recognized, assistance is designed for move task. The move task can be defined to follow a certain path. In this case we need to have large velocities along the path and small velocity component orthogonal to the path. For grasping the object, an attractive force field is created so that the user spends less time trying to reach the object. This attractive force field can substitute for a replulsive field when an obstacle is encountered. In this way different assistances are used at different time depending upon the type of motion. The forces are applied to the manipulators end-effector. The force constraint changes from soft to hard and back forth depending whether the user is trying to position the object or to move the object.

Chapter 6: Virtual Environments and Simulation Experiments

6.1 Introduction

Virtual environments (VE) are computer simulations of objects and events that are perceptually very similar to real environments. The common feature of most VE systems is the generation of a visual three-dimensional world that can be freely explored in real time. Crucially, the user can interact with the objects and events in the simulation, and this interactivity often leads to a temporary sense of 'presence' within the artificial world. People with physical disabilities are particularly likely to benefit from the leisure potential of virtual environments as their range of real-world activities is restricted in comparison with that of able-bodied people. Since physical disability can result in poor development of spatial awareness, and, as the hub of VE systems is the generation of a three-dimensional world, it is possible that physically disabled people can reap the full benefits of this technology. Given the demonstration of relatively poor spatial performance by people with disabilities, which would be expected from their real-world performance, it is important to determine whether such people can improve their spatial learning efficiency with training in a number of different virtual environments.

If virtual environments are to be useful for improving the spatial competence of physically disabled people, it is crucial to show that what they learn in a virtual environment will transfer to a real-world equivalent environment. Not only is this theoretically interesting, but it suggests the kind of practical application for which virtual environment experience may prove useful. Virtual environments (VEs) could be of considerable benefit to persons with several disabilities such as Muscular Dystrophy, Cerebral Palsy, Multiple Sclerosis and Cerebrovascular Accidents which present symptoms including limited movements or maneuverability, reduction in strength, spasticity, tremor, and a wide range of dexterity problems. These symptoms do not necessarily involve any reduction in the touch and/or feel senses.

Recently, haptic interfaces can be considered as a type of a robot designed to interact with a human being via the sense of touch. We use our hands to reach and grasp objects, also to confirm what our senses suggest. Haptic interface techniques and technologies are becoming increasingly useful for applications in assistive technologies, for example they form a natural interface for people with various impairments or as a means to assist target selection for individuals with motor impairments. In this project, a haptic device has been used for learning techniques and intensive rehabilitation treatment for persons with disabilities. The device teaches the correct movement patterns, as well as correcting and helping to achieve point-to-point movements, using virtual, real and augmented environments. Different Models have been implemented to explain the trajectory planning of the human arm reaching movements.

This chapter also presents a discussion of the experiments that were performed in order to examine the effectiveness of using haptic rendering capabilities with various kinds of assistance and force feedback for motion-impaired users in virtual environments. Therefore, if haptic feedback can be incorporated into the human-machine telerobotics system these users can benefit from the enhanced interface from using touch and feel interactions. The execution times and performance are compared between the various forms of assistance functions for people with and without disabilities.

Significantly, our results also show the characteristics for the human reaching and returning movement for different tasks. The results obtained from our experiments suggest that our virtual trajectory is a good model for studying the human arm movements.

6.2 Development of the Virtual Office Environment

In our Virtual Office Environment application development approach, we have created a scenario (i.e. a work desk) to address functional processes that are relevant for a range of populations. This scenario is generally conceptualized as an “open platform” that could be used to study, test and train a variety of cognitive processes depending on the interests of the researcher.

As with the Virtual Environment, the user sits at a real desk, they see the scenes that make up a standard office setting. The virtual desk can contain a phone, computer monitor, and message pad etc. throughout the office and a virtual clock ticks in real-time. Also it would require the user to look for the object that changed its color, get the object to the desired location. Currently for the sake of simplicity primitive shapes like cubes, cylinders, spheres etc are used to represent all the objects. Essentially, functional work performance challenges typical of what occurs in the real world can be systematically presented within a realistic office VE.

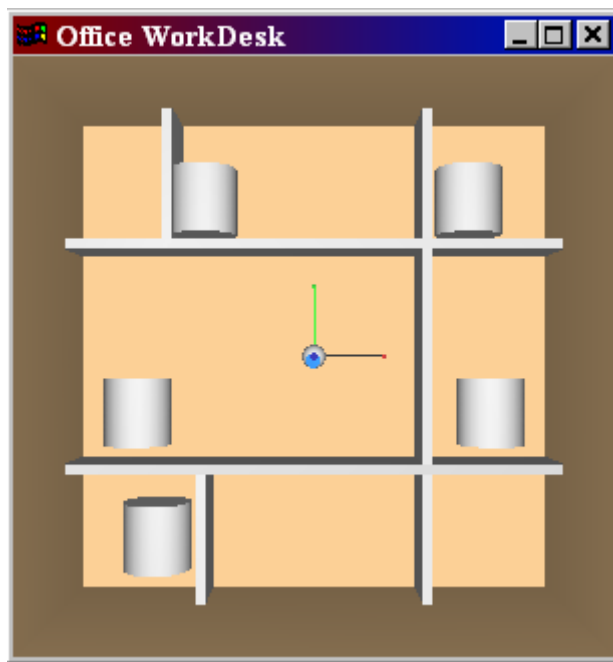


Figure 6.1 Virtual Office Environment

The Virtual Office is set-up to contain certain items to used daily and have been placed throughout the scenario. The target items can include common items, or those things

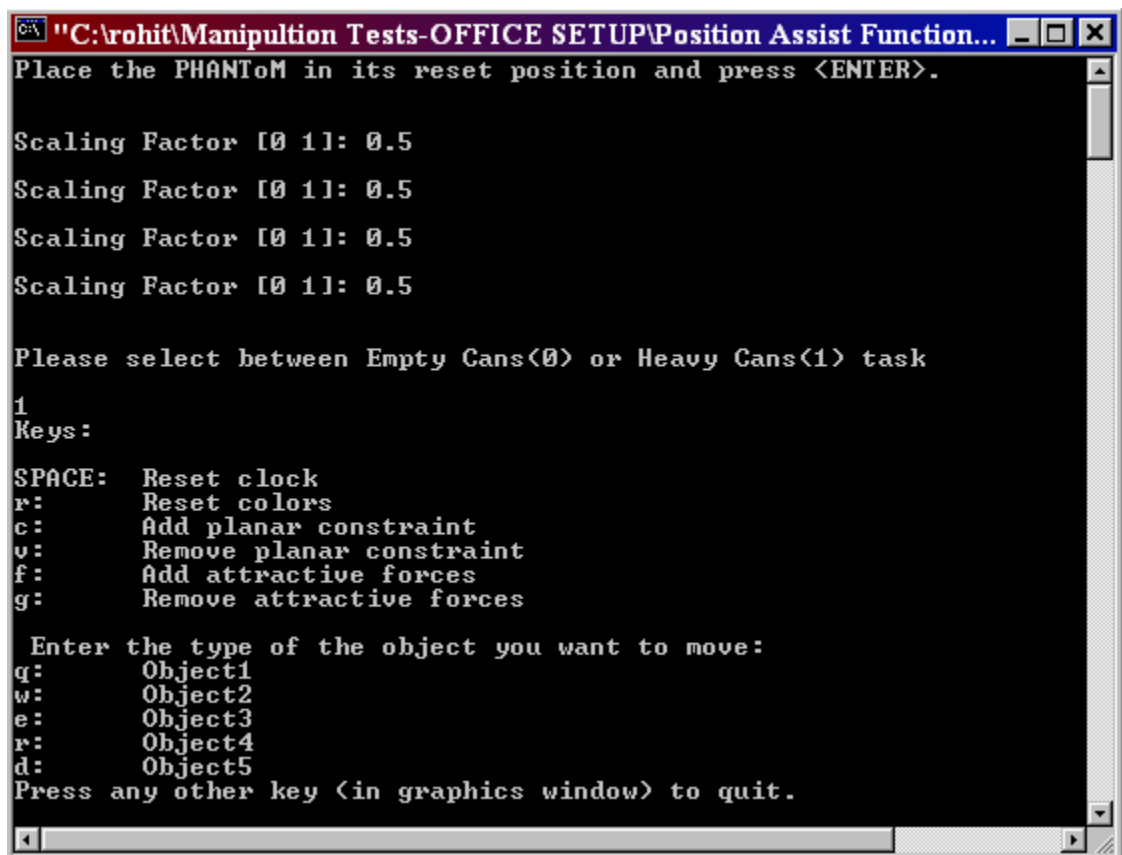
typically found in an office (e.g., notepad) and some uncommon items. Common items include: telephone, flower vase, calendar pad, pencil holder, clipboard, stack of books, coffee cup and a framed family photo Uncommon items include: dog, fire hydrant, stop sign, hammer, blender, guitar, basketball and blender.

The cornerstones of this kind of Virtual Environment (VE) technologies are “interactivity” and “immersion”. These kind of VE systems for movement and motion therapy are just beginning to be developed [29]. In developing these systems, the burden of proof rests with producing evidence that therapy in a virtual world is better than “real world” therapy at preparing individuals to function in the real world. One possible advantage is that it can augment a therapy environment, for example in this case by using haptic devices to provide novel sensory input.



Figure 6.2 Virtual Office Environment with Teach Pendant

The computer display above showed the desired movement trajectory, derived from an unimpaired subject, which could be used as a virtual, animated “teacher” to demonstrate the desired movement to the subjects. The teacher animation could also be adjusted in speed or displayed as a trace. It has been showed that training in the virtual environment for ten treatment sessions reduced reaching error performance by 50%. Using a similar enhanced virtual feedback approach, two subjects with cerebral palsy and spasticity were trained to perform tasks in this environment. The subjects also showed an increase in the performance.



```
"C:\rohit\Manipultion Tests-OFFICE SETUP\Position Assist Function...
Place the PHANTOM in its reset position and press <ENTER>.

Scaling Factor [0 1]: 0.5
Scaling Factor [0 1]: 0.5
Scaling Factor [0 1]: 0.5
Scaling Factor [0 1]: 0.5

Please select between Empty Cans<0> or Heavy Cans<1> task
1
Keys:
SPACE: Reset clock
r: Reset colors
c: Add planar constraint
v: Remove planar constraint
f: Add attractive forces
g: Remove attractive forces

Enter the type of the object you want to move:
q: Object1
w: Object2
e: Object3
r: Object4
d: Object5
Press any other key <in graphics window> to quit.
```

Figure 6.3 Interface for Virtual Environment

6.2.1 Tasks Definitions

One right-handed healthy subject participated in the experiment. He presented normal vision and perception of visual environment. The experiment consisted of a goal-oriented task, and was performed with the aid of a virtual environment and with/without assistance. The virtual environment resembled what the user saw on the table in the real world. A test bed with 4 different objects was placed and the same was replicated on the 3D graphical environment. This was done to provide a better understanding of depth, while presenting 3D information, on a 2D display. The task was a movement between two points presented on the screen. They involve reaching out from the body (extension of the arm) from a proximal point towards a distal point in the virtual workspace, and returning back near the body (flexion of the arm) to the proximal point and again place the object at the destination by reaching out from the body. Each movement was given a particular assistance for various times and for different objects. At the start of each exercise, the haptic interface guided the user to the start point of the movement. Subject was told: “The red cylinder represents the object you have to pick in the system, you should move from the initial position to the final position, when you place at the target, it will change the color to yellow. We need you to repeat this sequence for 10 times on each segment.” For each segment, the haptic interface detected users successful arrival to each target point with 2 centimeter accuracy and notified them by changing the color of the start point and the target point.

The tasks chosen were based on a perceived need to increase the manipulation capability of the individual by reaching and handling objects via Teleoperation in different environments. The approach task could be used to go and grasp the desired object and the move task could be used to move a light or a heavy object from one location to another. During this experiment, the haptic device was set to move freely, while the user interacted with the virtual environment. This configuration is also used in order to observe how the subjects performed reach and return movements with their arm.

Five cylinders were created inside an office desk. The subject was supposed to fetch the four cylinders by grabbing them with the stylus, which was linked to the cursor on the screen. The time required in order to move the cylinder was recorded and displayed as an output. Four different variants of the experiment were developed.

- The subject was asked to grab the four cylinders in a depending on what he wants in the three dimensional space.
- The subject can select the kind of assistance he/she wants to fetch an object.
- To optimize the performance of the task the probability of a hard impact with the back of the shelf has to be minimized.
- The constraint can be applied depending on the position of the master and the subject was asked to perform the same task.
- After taking the constraint off the subject was asked to perform the task.

Handling task is best described in terms of a number of distinct subtasks. Approach a randomly located object, grab the object, lift the object, move the object out of the shelf, move it to a different location in the reference frame chosen by the operator and release the object inside another shelf. The operator will see the object and the space in which it is to be moved and decide where to relocate the object. Then the operator grasps, lifts, moves and releases the object by guiding a robotic arm through all the actions. The idea is to assist the operator in guiding the robotic arm in a smooth trajectory by canceling all the distortions at the input. A representation of the scene for the handling task is displayed in Figure.

If the user knows the destination of the object he is trying to move, he can simply select the destination by clicking the stylus. Then appropriate assistance will be applied for that particular object from the initial location to the final destination. The various assistances for different objects are mentioned below. For convenience the objects are named from object1 to object 5.

Object 1: This object is located in shelf 1 at the coordinates (70,70,-25). If the user feels he can move the object to any location he wants to he can simply use the velocity assist function by pressing the key 'w' on the keyboard. By pressing this key the object changes its color to red to make sure that this is the object the person is trying to move. He can use this to move the object to any desired location within the virtual environment. The object will have no weight. If the user knows that the object will have weight, he can simply press the key 's' on the keyboard and he can feel the weight of the

object. He can also select a planar constraint while he is moving the object to avoid any movement beyond a particular plane. This will make sure that the object remains in a particular plane while he is moving to the destination point.

If the user wants to select a particular destination for the object, he can do so by changing the status variable to 1. The status value is set to zero by default. By changing it to one, the user tells the program that he wants to specify a particular point for moving the object. Before he approaches the object for grasping, all he needs to do is to specify a particular point for moving the object by clicking the stylus and once he clicks he selected that point as his destination. He can now approach the object and move it that location. In this case care has been taken so that all the motions away from the line joining the initial point and the destination are scaled down and the user can only move the object towards the destination. If the user wants to cancel all the rotations for the object while moving he can do so and all the rotations for that particular object are cancelled while moving.

Object 2: This object is located on shelf 2 at (80,-25,-25). As explained above if the user wants to move the object anywhere he wants he can do so by pressing the key 'q' on the keyboard. If he desired to have some weight for the object he can do that by pressing the key 'a' on the keyboard. He can also specify the destination by clicking the stylus and that point is recorded as his destination.

The above-mentioned assistances are also implemented for object3 located at (-70,-70, -25). The appropriate key presses for this object are keys 'e' and 'd'.

If the user wants to constrain the motion of the object along a particular line using force assistance he can do so by selecting the key 'f' from the keyboard. In this case a force constraint is produced along the line defined by the initial point and the destination point provided by the user. He can also move the object along a curve by selecting the curve assistance function so that the linear motion of the master is transformed into a curve trajectory for the slave. To select this assistance the user hits the key 't' on the keyboard. The user can also select a variable assistance where the scaling factor is increased or decreased proportionally to the location of the slave with respect to the goal.

The focus of all these tasks was to measure, with a high degree of accuracy and repetitiveness, the dexterity and reaction time of subjects with specific disabilities. People with disabilities such as those associated with Multiple Sclerosis and Parkinson's disease and Cerebral Palsy would benefit the most from these experiments.

6.2.2 Test-Bed for the Experiments

The test bed used for the simulation experiments was described in Chapter 3. The system interfaced with a virtual environment in order to simulate the execution of the chosen task. A mapping between the PHANTOM™ inputs and the movements and forces

performed in the virtual environment was simulated. It was necessary to develop and simulate such a mapping because the restricted range of motion and forces for a person with disabilities are limited and inadequate for performance of the task. Since the tasks were to be performed by an operator from a remote location, a graphical model of the environment was developed using the GHOST[®] software development kit and controlled by use of the PHANToM[™].

6.2.3 Haptic Interaction Simulation

To use the Phantom Haptic Interface device to actually feel the objects in a scene graph or to have to move the objects dynamically in the scene the `gstPHANToM` class has to be included minimally. For all the simulations presented here the `gstRigidBody` is used from the derived classes of `gstDynamic` class.

In this particular case, it is the Slave representation that is in contact with objects such as the cylindrical piece. Also, there are interactions between this piece and another object, being the boundaries of the holes, shelves, walls and boxes. GHOST does not provide a direct way to handle those interactions. Therefore, the development of a short algorithm was mandated in order to simulate the various interactions.

In order to simulate the interaction of the slave object with the different cylindrical objects, classes named `Frobject`, `Velobject`, `Varobject` of the type

gstRigidBody, were created. This class possesses a function that updates the dynamics of this object at every sampling time (1ms). The cylindrical piece is then added as a node of the Frobjct class. When the dynamics of the Slave are updated, the algorithm checks whether the cylindrical piece has been grabbed.

Once the cylindrical piece is grabbed, the dynamics of the slave should match the dynamics of the object that the slave is carrying. These are assigned to the cylindrical pieces using the setLocation(translations) and setRotation(rotation) functions of the objects classes. Here 'translations' and 'rotation' correspond to the Cartesian position and the rotational angles of the slave respectively.

To simulate the weight of the objects, a class called "PointAttractor", of the type "gstForceField" (provided by GHOST SDK), was created. This class has a function named "calculateForceFieldForce". The function calculates the force field effect of the bounded haptic object created, which, in this case covers the whole workspace.

The function boundBySphere(gstPoint(0, 0, 0), 1000) implements the force field effect at the origin (0,0,0) with a radius of 1000, which is larger than the workspace. In this way it is assured that the weight effect of the cylindrical piece will be felt all over the space. Finally, the force field effect is then calculated as a vector with the force pointing in the negative Y-direction, which simulates a gravitational force due to the cylinder's weight.

Additionally, this same function called “calculateForceFieldForce” is used to simulate the contact force of the cylinder, once it touches the shelf (a flag name “shelf” has the Boolean value of 1 in this case), with an opposite force pointing in the positive Y-direction.

6.2.4 Recording Techniques and Data Analysis

During a session, we recorded the positions, velocities, grabbed status, and the forces of both the master and the slave. Position values were used to plot the trajectories of the master and the slave. Velocities were also recorded in the same way as position values. For each task, the mean values for time of completion were also calculated. . The results obtained from the additional investigations are presented in Chapter 7.

6.2.5 Therapy and Treatment

Cerebral palsy is characterized by an inability to fully control motor function, particularly muscle control and coordination. Depending on which areas of the brain have been damaged, one or more of the following may occur- muscle tightness or spasticity, involuntary movement, disturbance in gait or mobility, difficulty in swallowing and problems with speech. In addition, the following may occur: abnormal sensation and perception; impairment of sight, hearing or speech; seizures; and/or mental retardation and learning disabilities. Since no two persons are affected by cerebral palsy in exactly

the same way, individual treatment programs vary widely. But because all the persons with cerebral palsy have movement problems and learning disabilities, one of the important component of treatment will be a therapeutic exercise program.

Sensory Integration Therapy is one of the approaches to help people with cerebral palsy achieve their optimal level of functioning. This therapy helps to overcome problems in absorbing and processing sensory information. Encouraging these abilities ultimately improves balance and steady movement. Therapies include stimulating touch sensations and pressures on different parts of the body.

Occupational Therapy specializes in improving the development of the small muscles of the body, such as the hands, feet, face, fingers and toes. Therapists also teach daily living skills such as dressing and eating, as well as handling tasks in different environments. They may teach your child better or easier ways to write, draw, move, pick, place and feed themselves.

6.3 Hidden Markov Model Based Skill Learning Approach

People with spastic cerebral palsy (CP) display a number of difficulties when reaching and grasping objects, including slow speed of the affected limb, delay in initial reaching, slow/fast flexing and movement of the fingers and weak grasp. Motion analysis of these patients with neurological diseases allows us to study the kinematic and dynamic motor

abilities of different limbs. Various measurement setups for kinematic and dynamic gait analysis in rehabilitation environment exist and are widely used in rehabilitation institutes worldwide. Such a system constitutes of optical position measuring device and floor mounted force/torque sensors. Here a haptic device is used for the analysis and has an advantage over classical gait analysis systems due to the freedom of exploring the contact forces.

Some studies have shown that repetitive task-oriented movements are of therapeutic benefit. With the use of haptics and VE technology, patient attention and motivation can be enhanced by means of ‘Active Feedback’ that will further facilitate motor recovery through brain plasticity. Four different levels for ‘Active Feedback’ have been identified: visual, haptic, auditory and performance cues.

Visual cues: In some cases, subjects tend to be confused about what they see. The brain needs to be reeducated to associate (for example) colors and objects. As a result of the need of cognitive re-learning it is important that visual cues be simple, yet stimulating. Visual cues can be represented using real tasks based on the ones used in occupational therapy sessions, to realistic and accurate goal oriented 3D computer environments. This can be anything from a virtual office (with desks and objects) to an interactive game.

Haptic cues: Kinesthetic feedback can help to discriminate physical properties of virtual objects, such as geometry. It can also be used to deliver physical therapy to a human subject using haptic interfaces. The force delivered in this way can be very therapeutic dependent on the way we apply this force to human muscular and skeletal

systems. It will undoubtedly play an important role when manipulating objects, either virtual or real. In conjunction with interactive virtual and augmented tasks, it can simulate the shape of a virtual pen, bingo card or the friction/drag when writing on the virtual bingo card.

Here, the methodology is based on creating a virtual environment, using a PHANTOM haptic interface. This is used for providing the tactile feedback to the patient. In virtual environment a labyrinth was created in patients frontal plane. By moving the stylus of the haptic interface the patient was able to move the ball pointer through the labyrinth. The patient's primary task was to pass through the labyrinth as quickly as possible with as few contacts with walls as possible.

The aim of the present study is to devise a test enabling the objective assessment of the functional capacity of different persons with disability utilizing a haptic interface. The test is reliable, easy to perform and would be an indicator of the success of therapists work in rehabilitation process. The labyrinth presented here guarantees a wide range of test complexities.

In this experiment we also demonstrate how a recognition system can be used for skill learning/evaluation purposes. Two persons with disabilities, both of them with no prior experience using the system will complete a dynamic task in the virtual environment and their performances were recorded. Our approach is to record the performance of the users in the virtual environment and use this to draw conclusions about the skill of the user.

This kind of training based on skill learning when applied to persons with disabilities helps in improving their motion performance. Also a tremor filter has been designed to suppress the tremor inputs coming from a person with disabilities.

Much evidence suggests that intensive therapy improves movement recovery. But such therapy is expensive, because it requires therapists on a person-to-person basis. Recently there has been increased interest in restoring functions through robot-aided therapy. This approach is to design therapy platform to substitute some of the therapist's work. The role of a therapist, like a coach, is replaced by the learned skill in the haptic interface. Since Hidden Markov Model is feasible to model a stochastic process, such as speech or a certain assembly skill, it was used to characterize the skill of moving along a labyrinth path.

6.3.1 Experimental Test Bed

The core of the measurement setup is the PHANTOM haptic interface and it is used as a force feedback generator. A complex and movement demanding VE is displayed on the patient's frontal plane. The patient can move the ball pointer by moving the haptic interface. The tactile information is fed to the user and he can feel the reactive forces when the ball touches the walls. This will ensure that he can get a realistic impression as if he/she was reacting with the virtual environment.

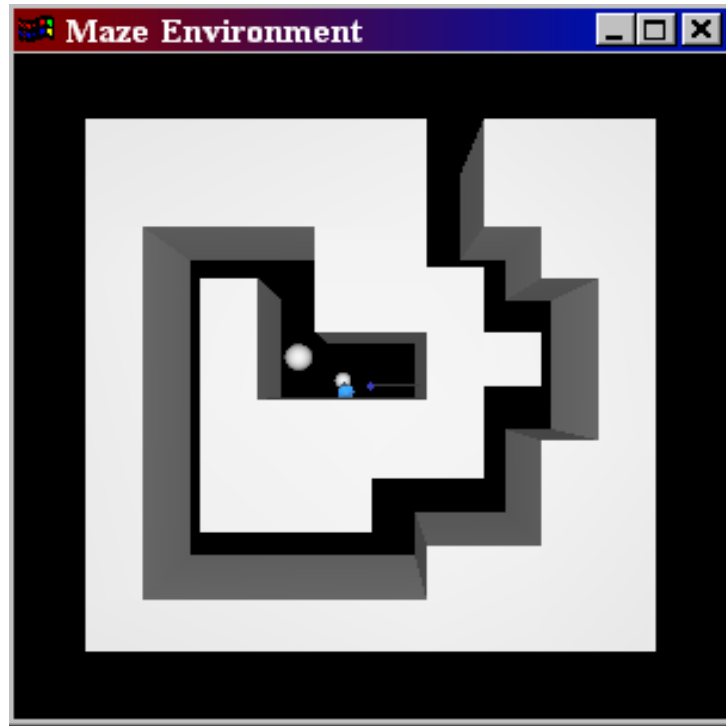


Figure 6.4 Maze Simulation Environment

6.3.2 Haptic Interaction Simulation

All the nodes here are derived from the `gstShape` class represent the physical geometry of an object to be haptically rendered (for example, cubes and sphere in this case). Geometric objects simulated with the GHOST SDK are currently limited to geometries consisting of rigid surfaces. Thus, when the position of a `gstPHANToM` node (representing the physical position of a PHANToM haptic interface end-point) passes through the surface of a `gstShape` object, the `gstShape` object does not deform – the geometry representing the object. Instead, a Surface Contact Point (SCP) is maintained for each `gstPHANToM` node. The SCP is forced to a point on the surface of any object

intersected by that `gstPHANToM`. The SCP, therefore, never penetrates the surface of an object, even if the position of the associated `gstPHANToM` node does penetrate the logical boundaries of the surface. When computing and sending a force representative of the `gstPHANToM` object touching an intersected `gstShape` node, the force normal to the `gstShape` surface is calculated using a spring-dashpot (springdamper) model for the surface. The force is proportional to the difference in the position of the SCP from the actual position maintained in the `gstPHANToM` node. In addition, force components tangential to the intersected surface are computed using a stick-slip friction model. Figure below shows how the SCP is used to track the surface contact point and calculate contact forces for a `gstPHANToM` node. This is done to actually feel and interact with the geometric objects placed in the haptic scene graph. The `gstPHANToM` class is used to add dynamic properties to the geometry nodes.

Typically, the `gstPHANToM` node is put in a separator directly on the root of the scene graph. It is usually desirable to have the PHANToM closely related to the world coordinate system. The `gstPHANToM` node is a derived class of `gstDynamic`, and it is used to inform the `gstScene` object of the PHANToM haptic interface's end-point position and orientation. This information is used to send forces back to the PHANToM. The `gstScene` object recognizes `gstPHANToM` nodes in the scene graph and ensures that forces generated from touching geometry objects are sent to the appropriate `gstPHANToM`.

6.4 Hidden Markov Model Based Human Motion Recognition

The Box and Blocks test was developed in 1957 by occupational therapists who assessed gross manual dexterity of patients with Cerebral Palsy using blocks and dropping them into a bowl. It has evolved into a task of a two sided box and 150 one inch cubes. The Box and Blocks can be used to test the right and left hand separately. Because the task tests for gross movements it may be easily adapted by robotics. However, the picking up of blocks is random and requires adaptation at the user interface. Preliminary tests have been done in simulation using the Phantom.

The master's position is obtained using the GHOST functions

```
thisXform = getCumulativeTransformMatrix();  
  
myTrans = thisXform.getTranslation( ); (position); (5.54)  
  
Vmaster = myPhantom → getVelocity( ); (velocity).
```

The current position and velocity of the operator (Master) are tracked through the coordinates

$myTrans . x()$, $myTrans . y()$ and $myTrans . z()$ (*position in each coordinate*)

$Vmaster . x()$, $Vmaster . y()$ and $Vmaster . z()$ (*velocity in each coordinate*)

6.4.1 Experimental Test Bed

We have implemented the algorithm described in the previous chapter in a virtual environment consisting of a graphic scene and a haptic device. In this experiment the object placed at $(-80, -65, 0)$ has to be grasped and moved to $(80, -65, 0)$. The obstacle wall has a width of 20 mm and height of 65 mm. The right hand destination has a slot for placing the object and has a width slightly more than the width of the object. The environment can be seen in the figure 6.5 below.

6.4.2 Haptic Interaction Simulation

In this experiment the representation of the end-effector in the virtual environment is the phantom node. The virtual environment is created using Visual C++ and the entire environment is contained in a box. The limits of the environment are shown by bounding cubes. The environment contains a moving target i.e the phantom node and a small cube that can be picked up and has the translations and rotation of the phantom.

The geometric objects simulated in this environment are rigid surfaces. Thus, when the position of a `gstPHANToM` node passes through the surface of the obstacle or the object, the `gstCube` object does not deform. The `gstPHANToM` node is a derived

class of `gstDynamic`, and it is used to inform the `gstScene` object of the PHANToM haptic interface's end-point position and orientation.

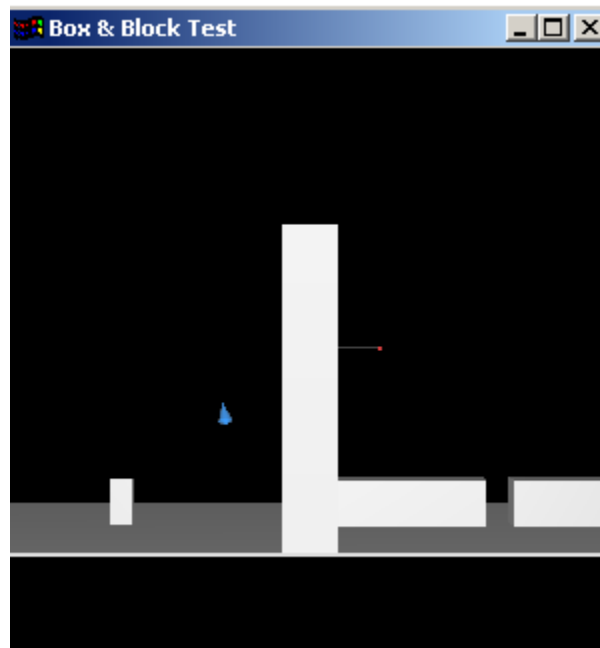


Figure 6.5 Box and Block Simulation Environment

Chapter 7: Results and Discussion

7.1 SolidWorks Simulation of RRC Manipulator Using the Phantom

A program was developed in C++ to send the phantoms position and orientation data over to SolidWorks simulation of the RRC Manipulator. The user interface includes dialog box interface on the Phantoms side as well as SolidWorks side which gives us numerical feedback of the phantoms position and orientation. The RRG Kinematix library was integrated into the NetGhost program allowing us to convert the Phantoms position and orientation into joint angle data and also performs a check for joint limits. These joint values can be sent to the SolidWorks simulation as the code accepts only joint angle information. Since the RRC Model was very complex the SolidWorks cannot update the model as fast as the Phantoms update rate. This new interface allows the user to manipulate the RRC SolidWorks graphics model in real-time using the phantom.

7.2 Tests Executed in the Office Environment

The results for the tasks execution in the office environment are discussed in the following sections. The patient's primary task was to perform the task as quick as possible. The comparison in performance with and without assistance has been carried

out in these subjects. The objective of these tests was to assess the hand function as an important part of the evaluation of a person's functional capabilities. The hand function is influenced by many different variables. Among others, these variables include age, diseases and the mental state of the person. Therefore, in order to test hand function it is necessary to utilize tasks representative of everyday activities.

7.2.1 Tests Executed by People with Disabilities

Two different operators, with severe forms of cerebral palsy volunteered to participate in some of the experiments. Their results are presented in the following sections.

All the graphs and the tables below represent the results of the operator with cerebral palsy. The data shows that with the incorporation of the various forms of assistance functions the execution times are shortened. These tests measure the skill and hand movements in the rapid manipulation of objects. Also as the subject performed the task many number of times, the time of execution for a specific task was significantly reduced each time. The graphs of the trajectories show the kind of assistance for each task.

Table 7.1 below shows the values of the time execution and the increase in performance when an operator performed the task on object at location A. The task completion time without assistance was 15.662 seconds. The last column in the table

represents the percentage decrease in time when assistance was provided to him for performing the task. Since the person was made to do the task for a number of times, the decrease in the time execution ranged from 20% to a maximum of 47 %.

Table 7.1 Results Using Planar Assist Function For Object Position A

Task No.	Approach Time (sec)	Move Time (sec)	Total Time (sec)	% Decrease
1	6.087	6.518	12.605	19.59
2	4.612	8.462	13.074	16.52
3	4.452	7.007	11.459	26.82
4	4.68	9.166	13.856	11.59
5	4.402	7.327	11.729	25.11
6	4.31	6.956	11.266	28.06
7	4.504	6.035	10.539	32.71
8	4.066	4.154	8.22	47.52

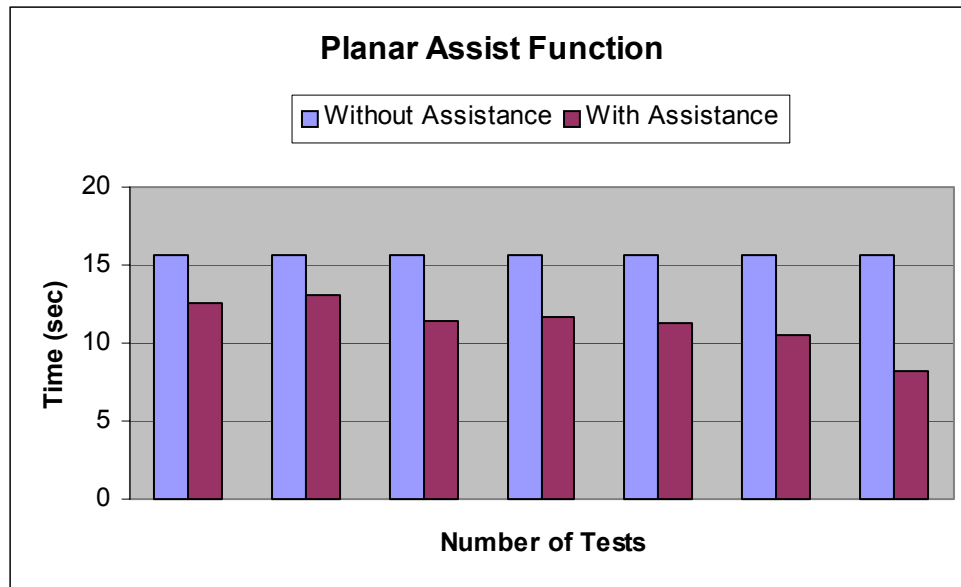


Figure 7.1 Graphical Analysis of Execution Times for Object Position A

The plots below show the trajectories in each coordinate and the kind of assistance incorporated in the execution of the above task. Figure below shows the approach task for grasping the object.

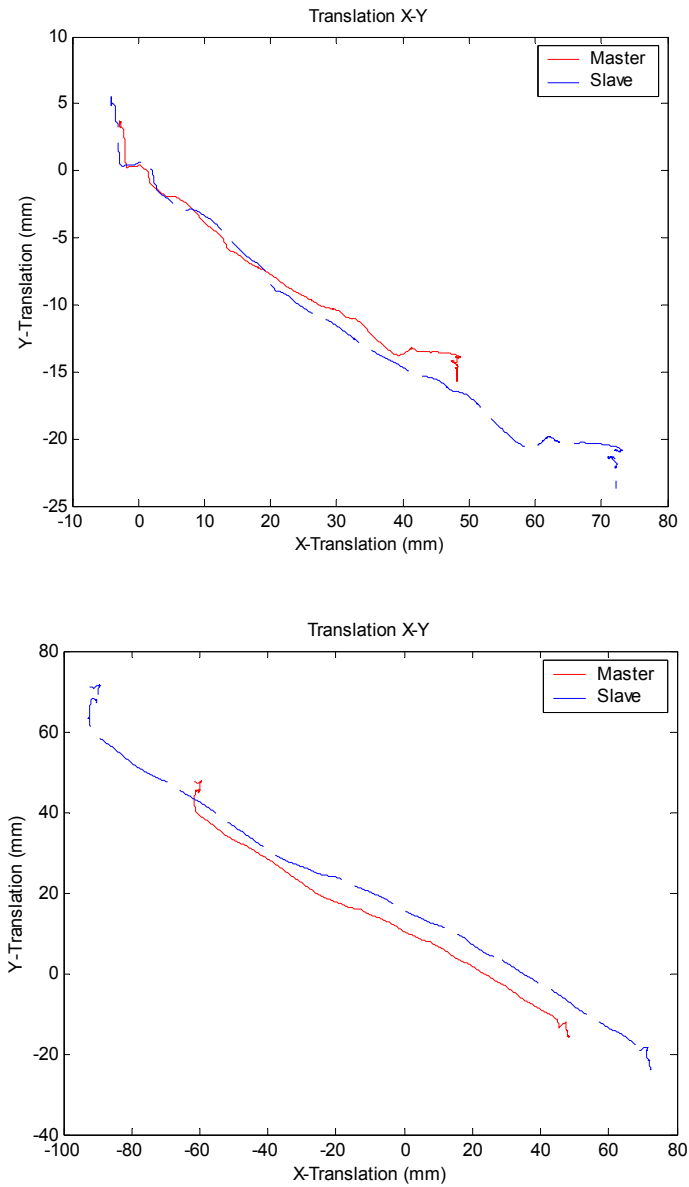


Figure 7.2 Approach and Move Trajectories for Object Position A

The figure below represents the trajectories in the XZ plane to show the planar constraint for a certain period of time. This constraint will be removed once the slave is

close enough to the destination. In this way the operator spends less time in free space to move the object. Unnecessary movement in the positive Z direction is eliminated.

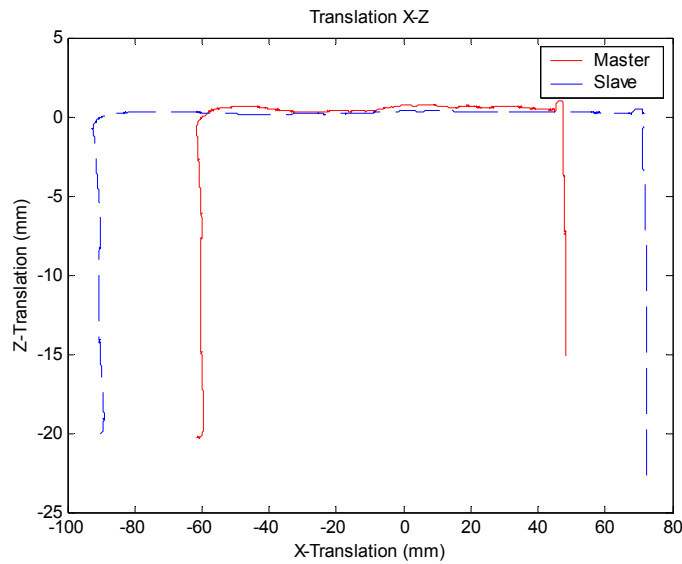


Figure 7.2 Continued

For the tests described before the operator had to move objects which were weightless. If you are working with a haptic interface, the force feedback capability has to be utilized so that the person can actually feel the weight of the object he is manipulating. The objects were given weight randomly and the results in the table 7.2 show that the person had completed the task in 16.305 seconds without assistance. When assistance was provided the decrease in the time ranged from 16% to 50%. Training in this manner helps a person to improve his/her hand function by therapeutic procedures such as physical therapy. The following plots show the results for these tests.

Table 7.2 Results Using Velocity Assist Function For Object Position B

Task No.	Approach Time (sec)	Move Time (sec)	Total Time (sec)	% Decrease
1	5.176	8.438	13.614	16.50
2	5.078	5.579	10.657	34.63
3	5.113	5.441	10.554	35.27
4	5.872	2.741	8.613	47.17
5	5.835	4.339	10.174	37.61
6	5.939	3.926	9.865	39.55
7	5.303	3.131	8.434	48.27
8	5.438	2.43	7.868	51.74

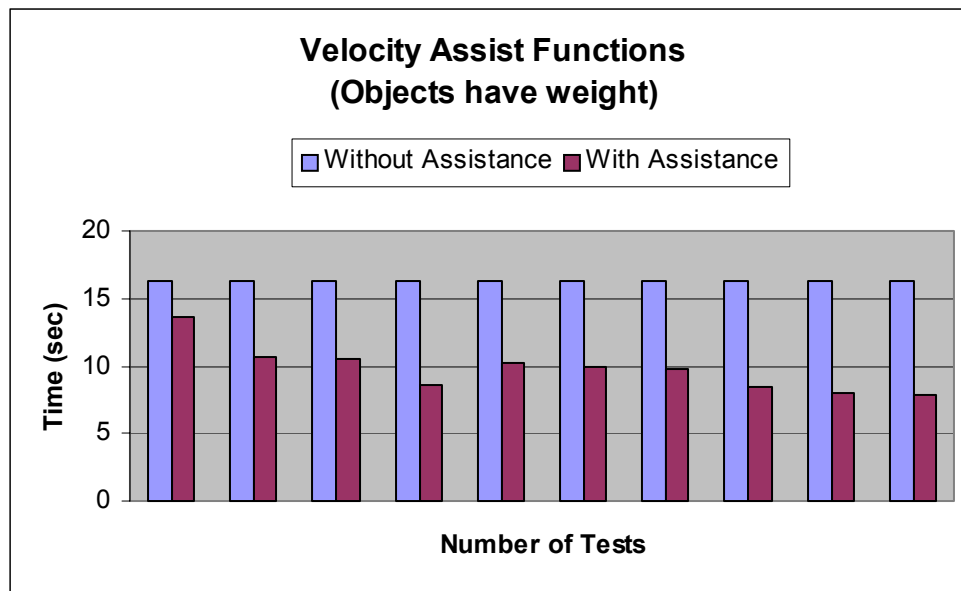


Figure 7.3 Graphical Analysis of Execution Times for Object Position B

Figures below show the Slave trajectories with respect to the Master trajectories for the various cases with the use of assistance functions. Additionally, the weight for the object significantly affected the execution time when no assistance was provided.

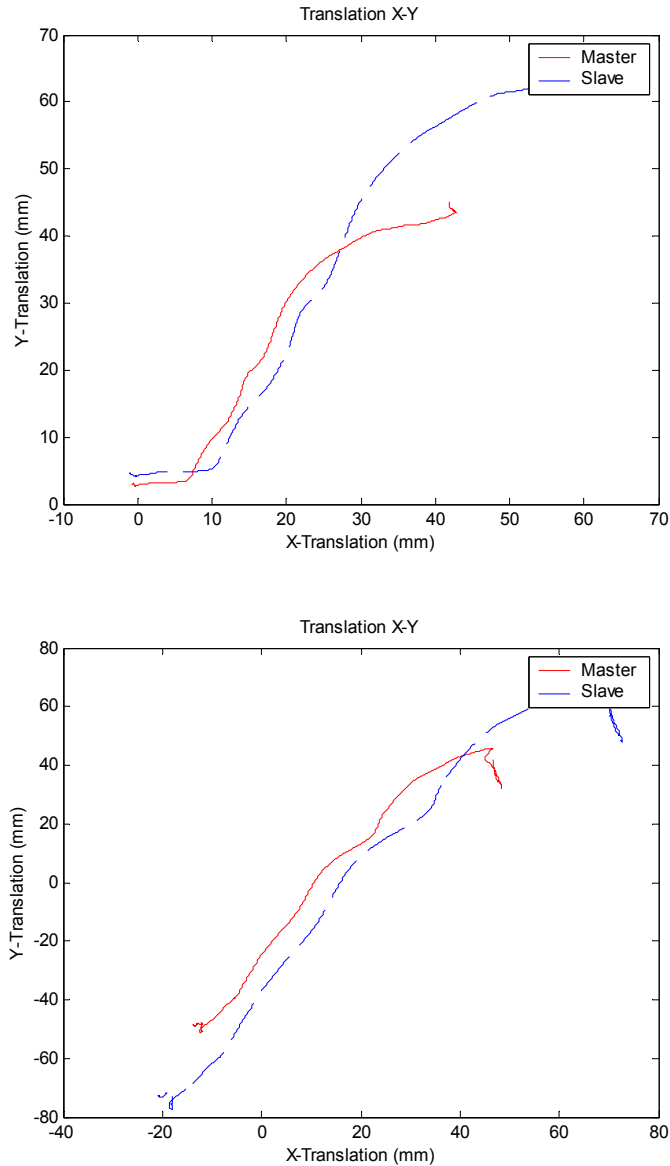


Figure 7.4 Approach and Move Trajectories for Object Position B

A scale factor of 0.5 was instrumental in reducing the execution time required to approach the object. The operator was able choose his destination and completed the task of moving the cylindrical piece to that goal location. In any event, the need for scaling in the undesired directions was apparent for the operators with disabilities.

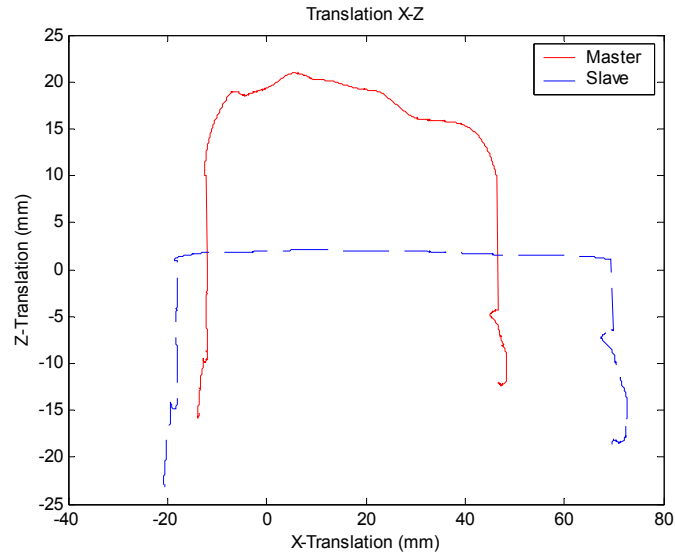


Figure 7.4 Continued

To avoid this the user interface allowed selecting a destination point for the object before he starts moving in to direction to grasp the object. Once the destination point is known, it will be recorded and a proper linear assistance will be provided in the direction of the goal point.

The velocity assistance function with constant scaling produced the results shown in Figure 7.4. In this case, the velocity in the x and y directions are scaled up using the scaling factor. The table below summarizes the time executions for the task of moving the object from location C to his/her own location.

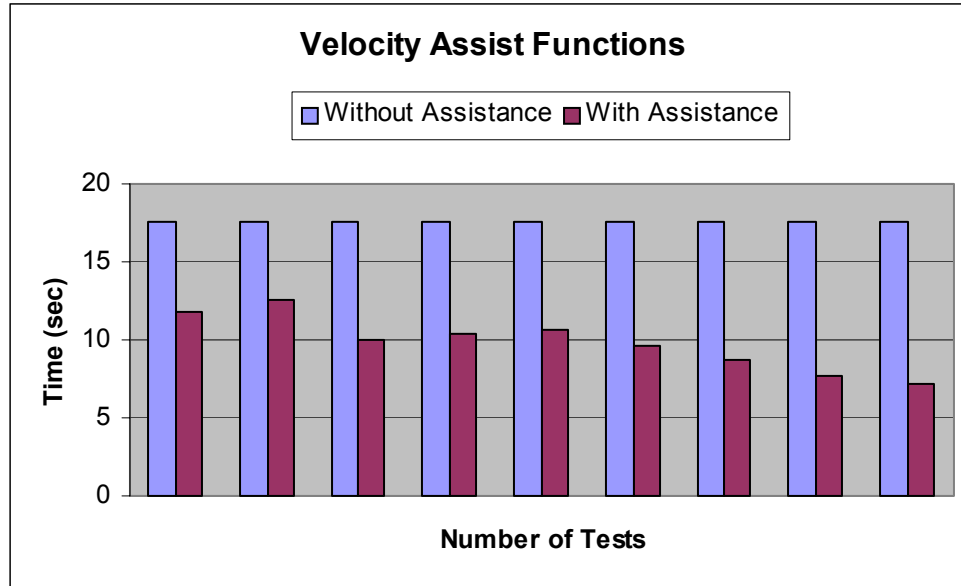


Figure 7.5 Graphical Analysis of Execution Times for Object Position C

Table 7.3 Results Using Velocity Assist Function For Object Position C

Task No.	Approach Time (sec)	Move Time (sec)	Total Time (sec)	% Decrease
1	4.426	7.416	11.842	32.46
2	5.256	7.309	12.565	28.34
3	4.558	5.437	9.995	42.99
4	4.731	5.614	10.345	41.01
5	5.120	5.555	10.675	39.12
6	4.380	5.200	9.58	45.36
7	5.164	3.616	8.78	49.92
8	4.54	3.19	7.73	55.91
9	4.045	3.151	7.196	58.96

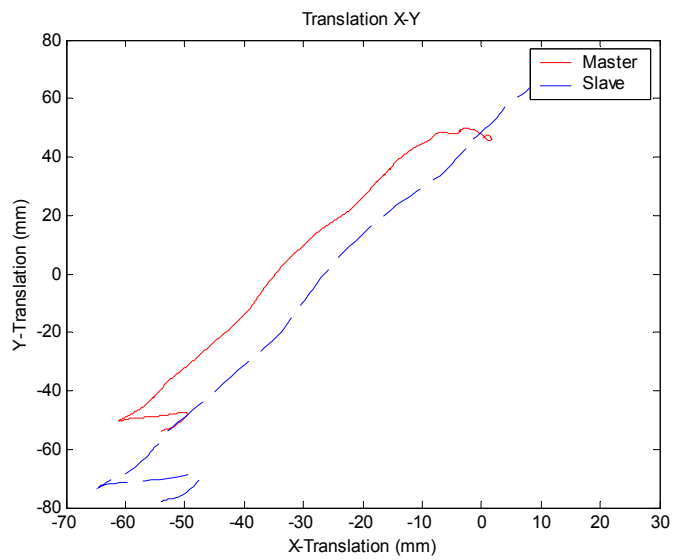
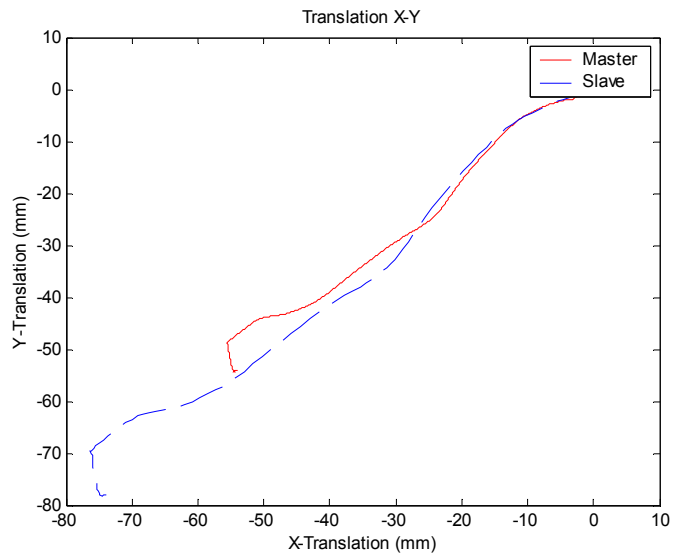


Figure 7.6 Approach and Move Trajectories for Object Position C

Figure 7.7 presents the results obtained from the move test when the Velocity Assistance function with a variable scaling factor was employed. The principle associated with this type of assistance function is explained in chapter 5. The rapid increase in speed in the x-axis due to the variable scaling factor is evident. The main difference with respect to previously used functions was the reduction in execution time.

Table 7.4 Results Using Variable Scaling Function For Object Position D

Task No.	Approach Time	Move Time	Total Time	% Decrease
1	4.375	7.289	11.664	33.48
2	5.887	7.242	13.129	25.12
3	3.536	6.566	10.102	42.38
4	2.399	6.052	8.451	51.80
5	2.660	4.806	7.466	57.42

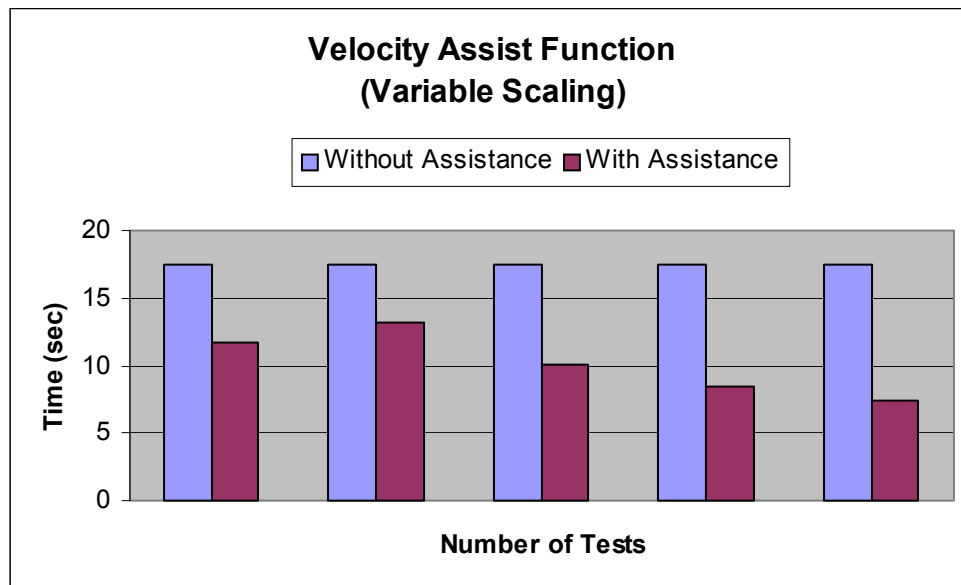


Figure 7.7 Graphical Analysis of Execution Times for Object Position D

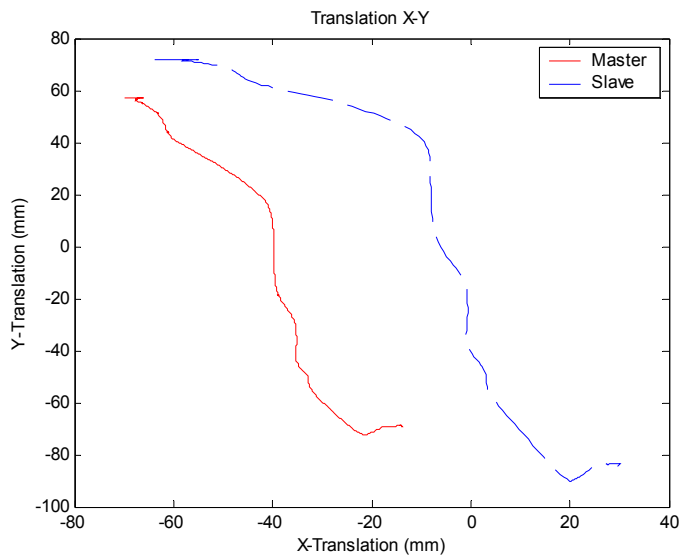
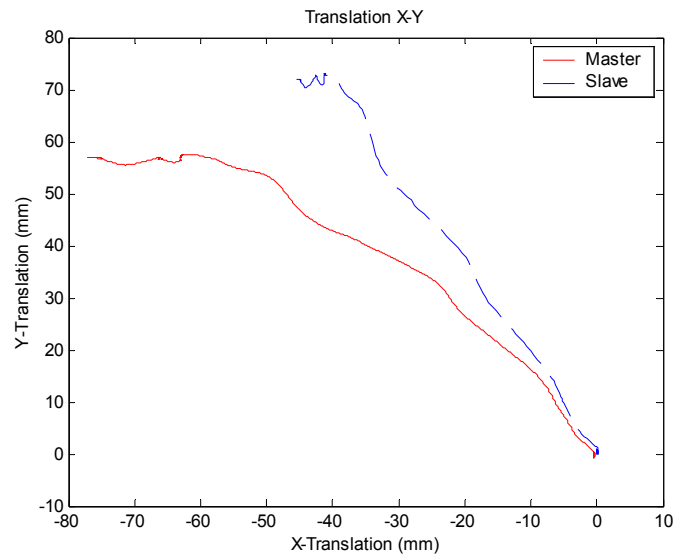


Figure 7.8 Approach and Move Trajectories for Object Position D

Figure 7.9 presents the results when the operator utilized a curved trajectory assistance mapping for obstacle avoidance. The data show that, even though the operator

had tremor and limited strength and limited motion-range, the operator was able to perform the task. During this test, a simple horizontal motion at the input was mapped onto a curved trajectory that allowed for obstacle avoidance (in this case the walls of the shelf). The plots of the execution times and the numerical values are shown below.

Table 7.5 Results Using Curve Assist Function For Object Position E

Task No.	Approach Task (sec)	Move Task (sec)	Total Time (sec)
1	7.34	7.290	14.63
2	6.16	5.320	11.48
3	6.358	9.292	15.65
4	6.084	7.088	13.172
5	3.927	8.998	12.925
6	5.035	5.707	10.742

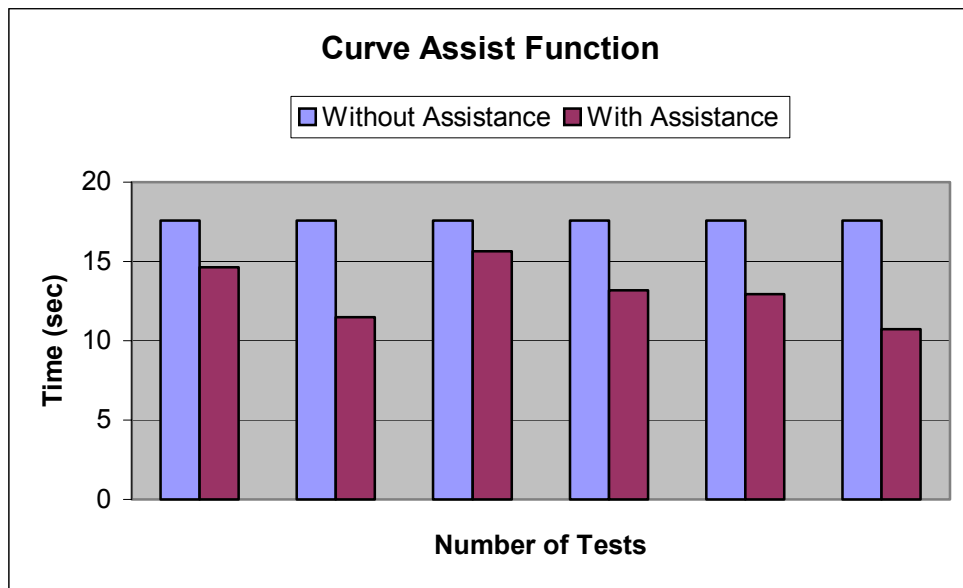


Figure 7.9 Graphical Analysis of Execution Times for Object Position E

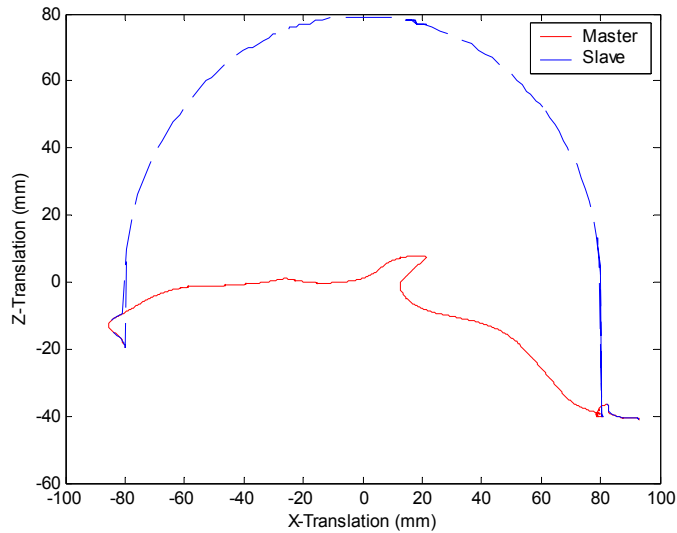


Figure 7.10 Trajectories for Master and Slave for Object Position E

The function below consists of mapping the horizontal movement of the operator onto a diagonal trajectory that defined the shortest path from the point where the operator was to grab the object to the point he selected as his destination (before he approached the object). The plot show that these forms of assistance functions were instrumental in helping to reduce the execution times associated with the tasks of moving the objects to a known destination.

The graph below also shows how a horizontal movement of the master is mapped on to a different path for the slave depending on the destination. The rapid decrease in the Y direction for the master is scaled down for the slave.

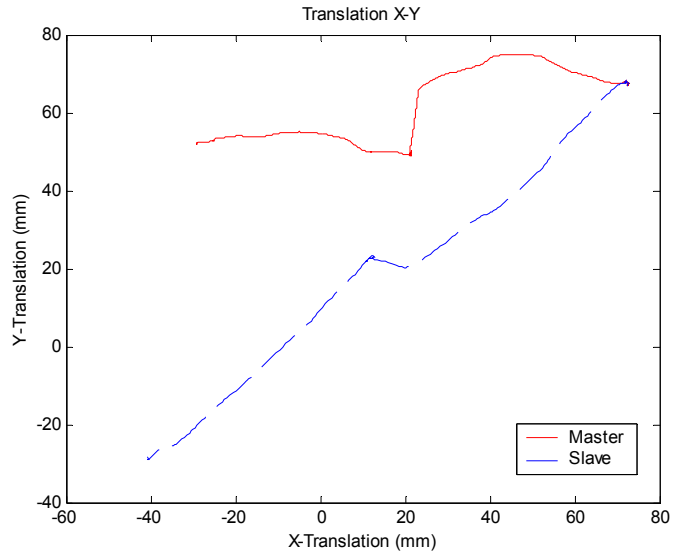


Figure 7.11 Trajectories for Master and Slave for the Move task for Object Position B

In all these experiments the performance is measured by the time it takes to execute a specific task. The reason for choosing commonly used occupational therapy tests is to assess the human-robot haptic system performance and to bring the two disciplines of occupational therapy and robotics engineering closer.

7.3 Results for the HMM Based Skill Learning Approach

Four different operators, two persons with disabilities and two normal persons, ran the set of experiments. The results are discussed in the following sections. The patient's primary task was to pass the labyrinth as quickly as possible, with as few contacts with the walls as possible. It has been applied to two persons with cerebral palsy as well as to healthy subjects. The comparison in performance has been carried out in these subjects.

7.3.1 Tests by Persons without Disabilities

Two normal persons without disabilities have performed in the maze environment and the data is collected 15 times. The data collected are stored in a data file and pre processed using the HMM. The trajectories of the normal persons are displayed below.

Figure below shows the trajectory paths along the labyrinth for normal subjects. As seen from the figure all the healthy subjects performed the tests well and showed good overall performance. All the subjects showed significantly better movement control judged on the parameters like the time execution, number of collisions etc described in the previous chapters.

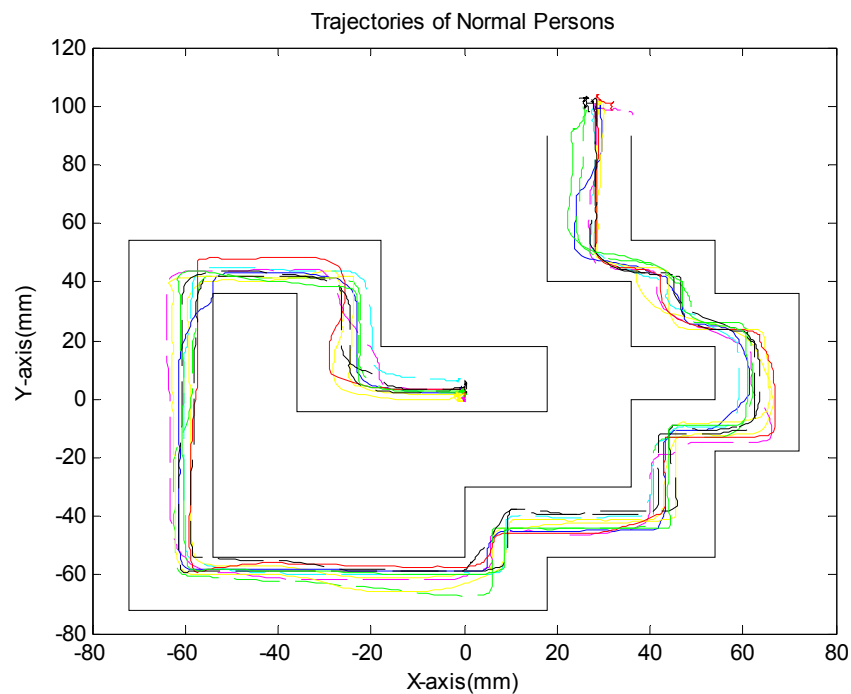


Figure 7.12 Trajectories of Normal Persons for Skill Learning

7.3.2 Tests by Persons with Disabilities Before HMM Training

Figure 7.13 show the sample trajectory paths along the labyrinth for persons with disabilities before training them using the HMM. Two of the trajectories will be selected and discussed further to demonstrate the individual characteristics and significant movement differences that appear.

Figure 7.15 show the contact forces occurring with the walls for the subjects. These forces result from the collision with a horizontal or the vertical wall. In the same figures we see the collision information showing the number of collisions and the impact duration time that represents the longest time interval the ball pointer is in contact with the wall.

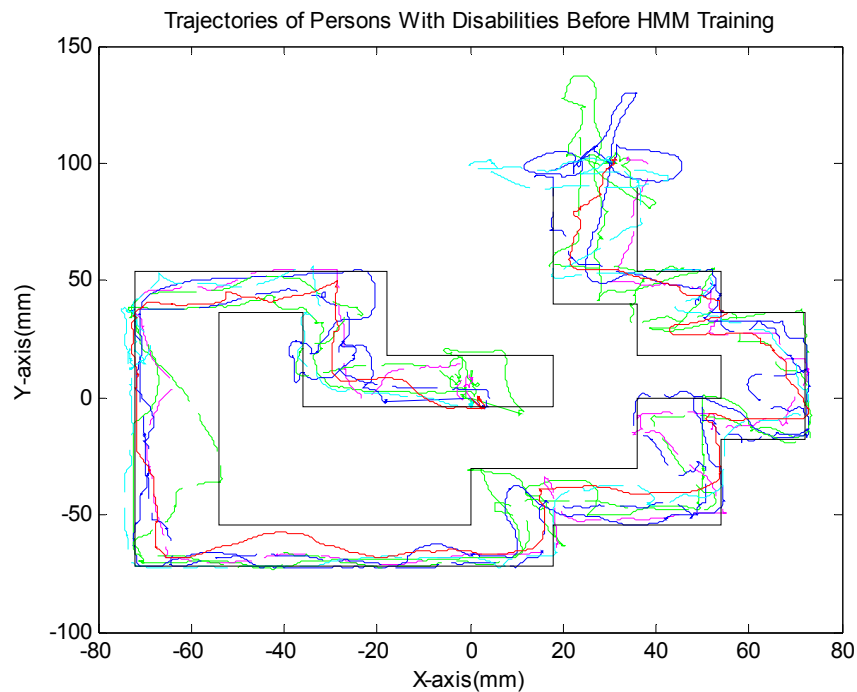


Figure 7.13 Maze Test – Trajectories of Disabled Persons Before Training

The trajectory shown in figures 7.14 below shows one of the trajectories. The patient showed an overall good performance but with tremor towards the end of the experiment. The tremor amplitude and frequency are analyzed in the figures that follow. The peak frequency is also shown in the graphs.

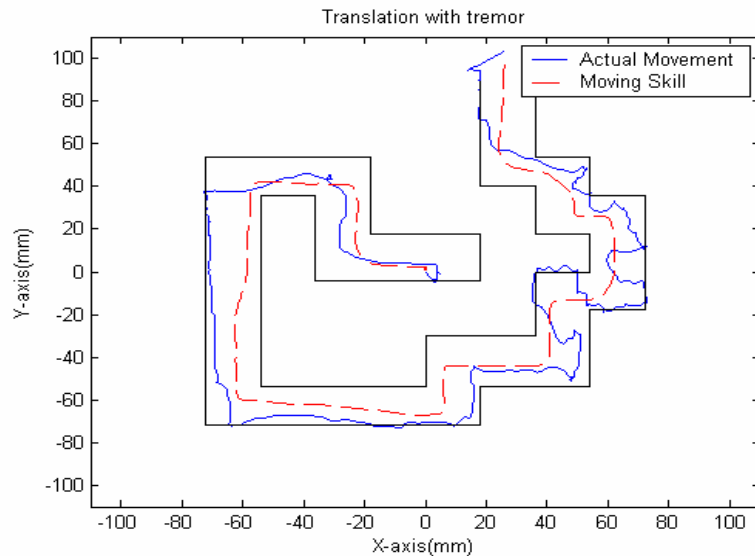


Figure 7.14 Trajectories of Disabled Person With Tremor

The figure below also show that the trajectory contains several smooth regions along the wall and a knot shaped region. The trajectory along the walls show that there has been a collision and some drag along the wall and the knot shaped region in one graph show the absence of movement for that instance of time.

The degree of trajectory complexity seems to be high showing some movement problems which is not seen in normal subjects. In all these cases the patient starts moving in the desired direction most of the time.

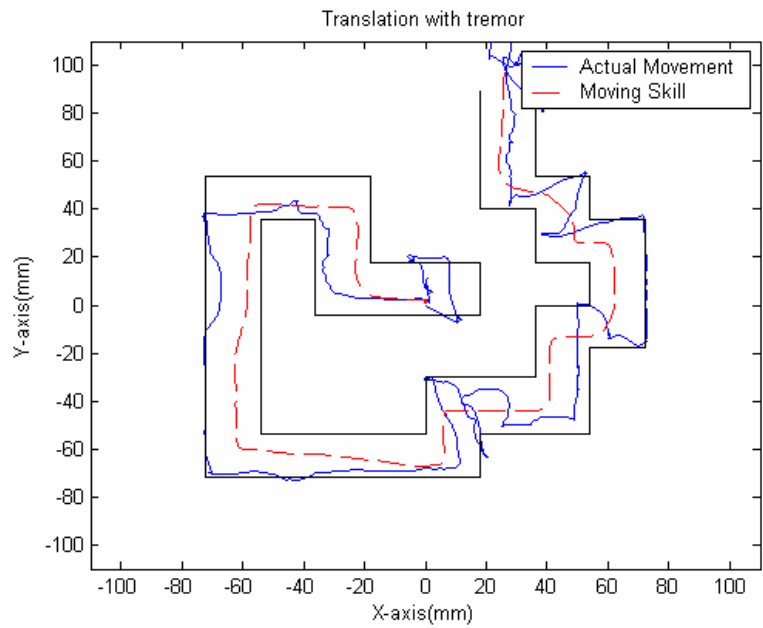
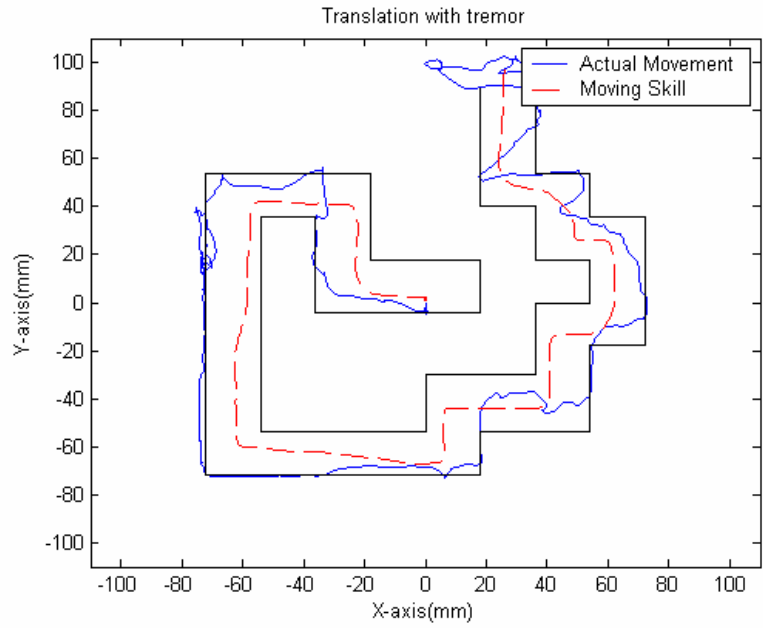


Figure 7.14 Continued

The plots below show the collision and force information for the trajectories discussed above.

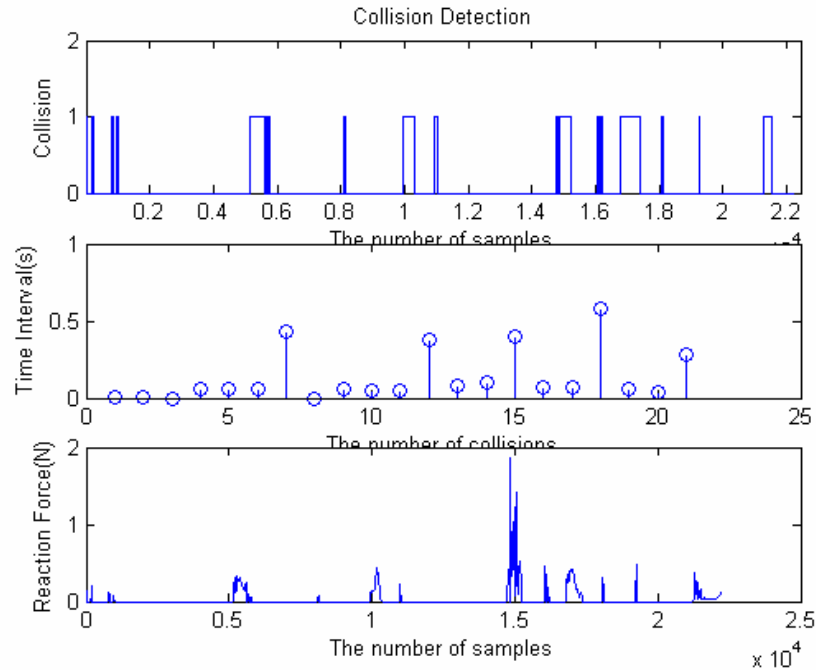


Figure 7.15 Collision/Force Plots of a Disabled Person

Sometimes in the trajectories shown above the patients were sliding the ball along the wall. In such cases the Y axis of the collision plot shows the time interval of each such collision. The bold lines in the force plots show the collision and the reaction force for each collision. All the non-zero forces show a collision has occurred. The maximum impact force was 2.467 N and the minimum impact force was 0.585 N. The maximum time for the collision event was found to be 3.23 seconds

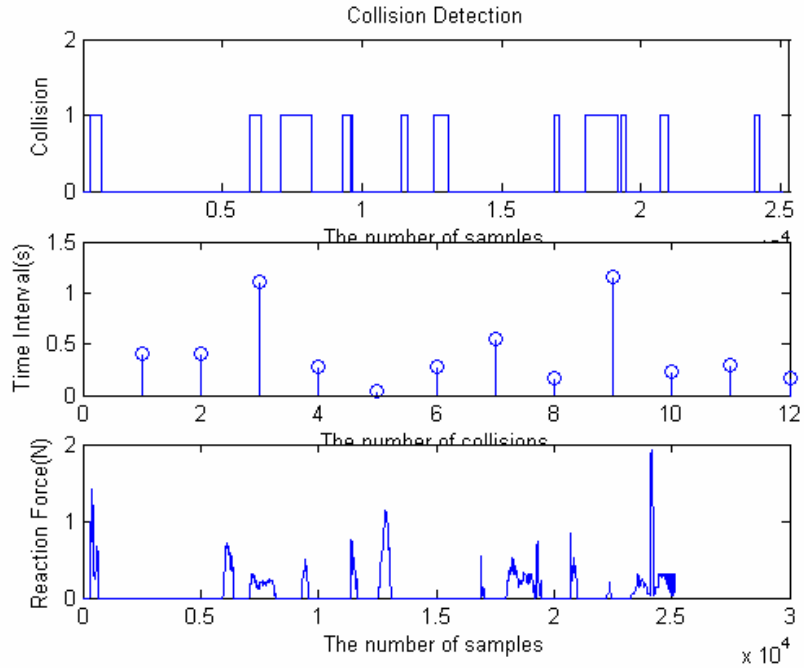


Figure 7.16 Collision/Force Plots of a Disabled Person - 2

The low contact interval times shown in the table originate from the quick withdraw of the ball and also from oscillations. The time taken for all the above tasks ranged from 17.953 to 25.165 seconds. The collisions were at a high of 25. The data including the forces, collisions, times taken etc are summarized in the table below.

Table 7.6 Numerical Results for the Labyrinth Before HMM Skill Learning

Number	Time Taken (sec)	Collisions	Max Impact Duration (sec)	Min Impact Duration (sec)	Max Force (N)
1	22.344	21	0.585	0.037	1.861
2	24.047	25	3.233	0.028	2.467
3	25.165	15	1.161	0.03	1.922
4	17.953	19	1.27	0.001	1.865
5	25.407	16	1.64	0.07	1.464
6	21.755	17	0.548	0.021	1.137
7	19.828	10	0.348	0.004	0.585
Std Dev	2.775	4.756	0.986	0.023	0.613
Mean	22.357	17.571	1.255	0.0273	1.614

Since a tremor filter has been designed to remove the tremor in these patients, it has been used and the following plots represent the trajectories after the introduction of tremor filter.

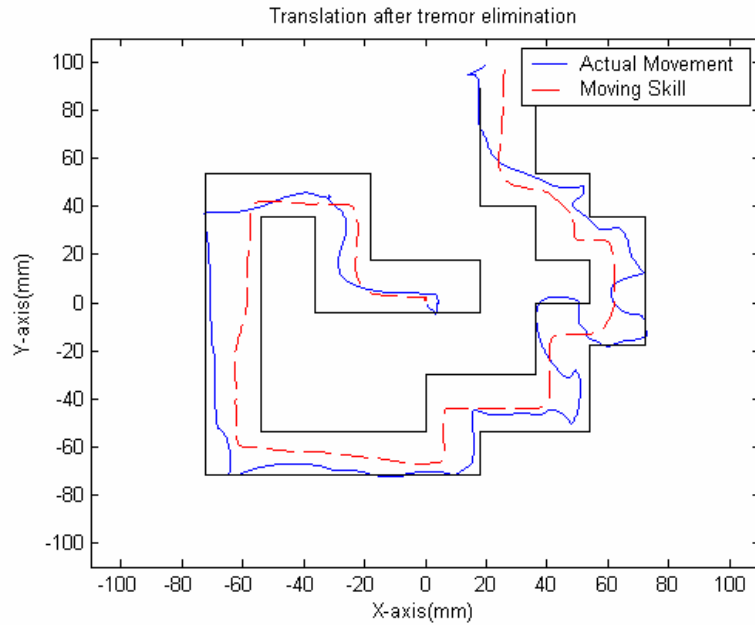


Figure 7.17 Trajectories After Tremor Elimination

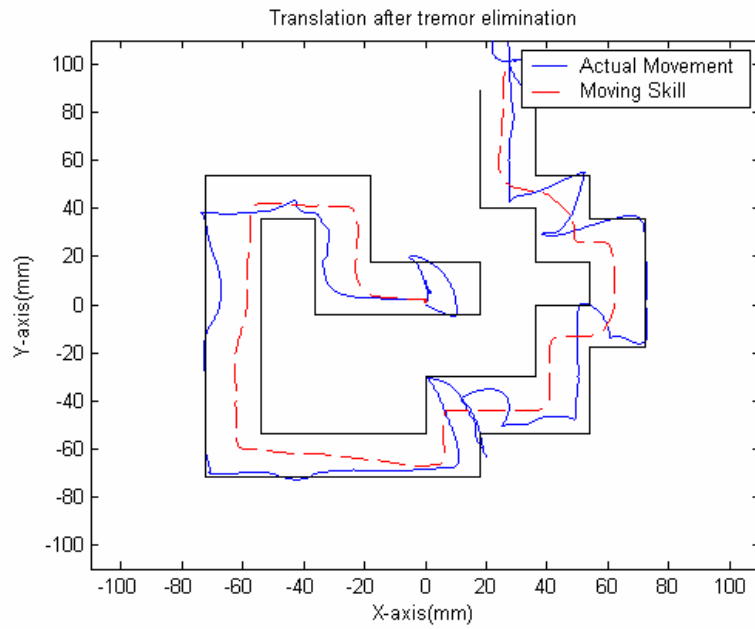
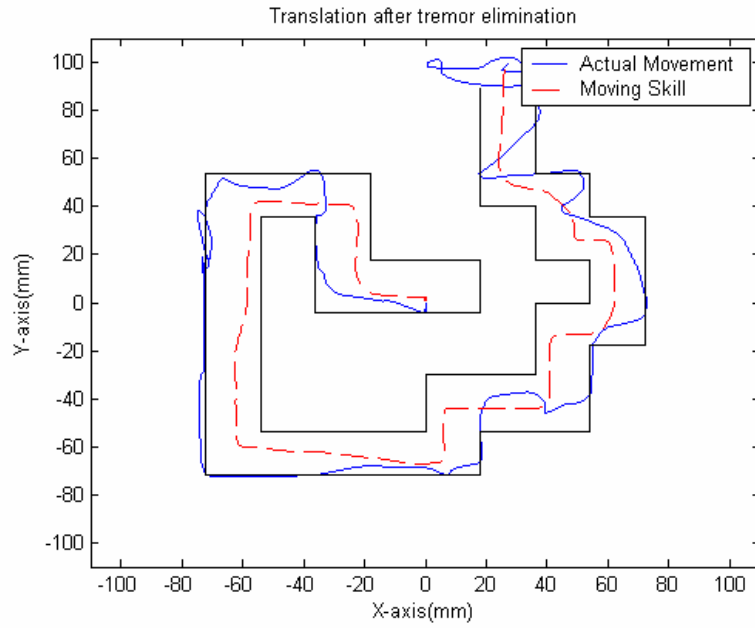


Figure 7.17 Continued

It is evident from the graphs that the tremor has been eliminated successfully and the trajectories appear to be much smooth than the previous ones.

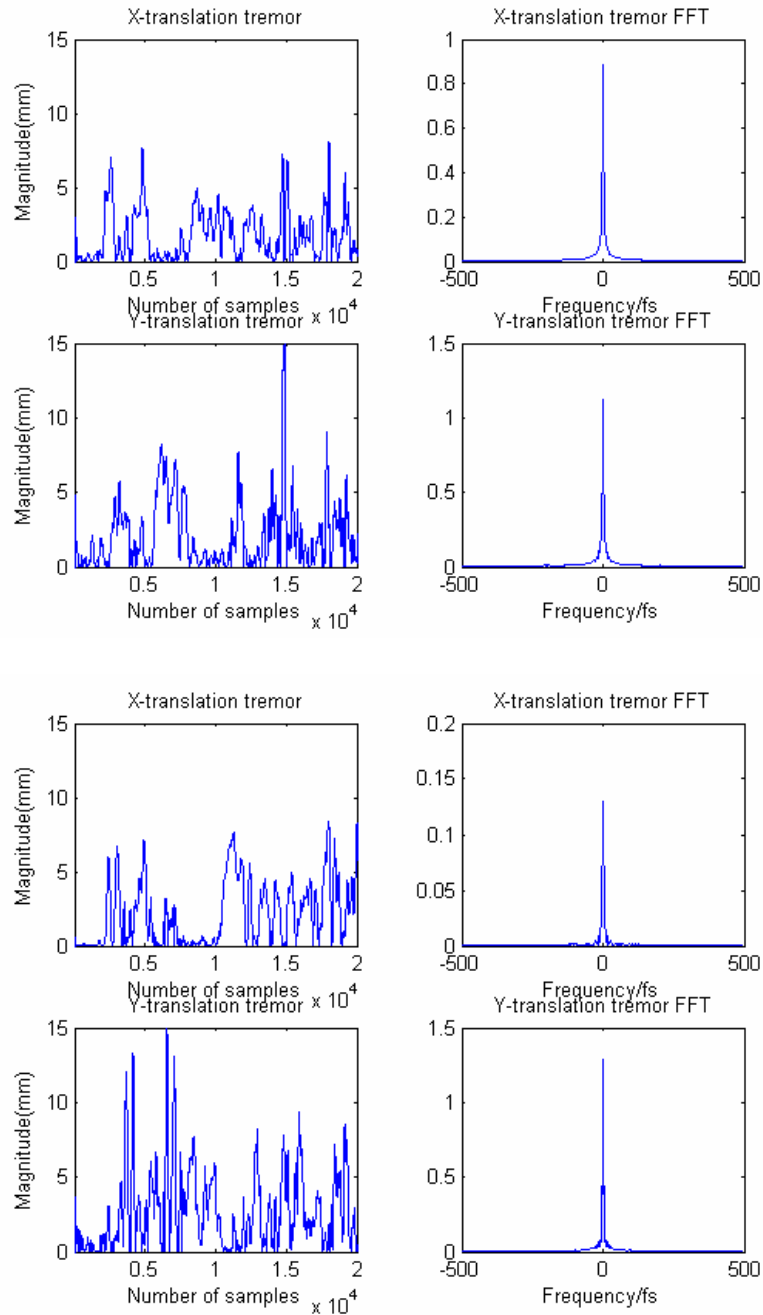


Figure 7.18 Tremor Plots Before Training

7.3.3 Tests by Persons with Disabilities After HMM Skill Learning

When humans perform a task, their actions reflect the skill associated with that task. For a particular task when one does it many times and each of his actions are measured, each set of data is different although it represents the same skill for the same task. We have modeled the action so that the underlying nature of the Human action is revealed and can be used to transfer the skill to people with disabilities. We want the person with disability to learn the human skill. The skill is gained incrementally and improved through learning practice. Learning from observation is a paradigm where one observes other peoples performance and learns from it. In fact the ideal trajectory represented in all the figures represent the skill humans have in achieving certain goals and following trajectories. For a disabled person much of the learning occurs through observation with feedback or tutoring. This kind of observation learning is an open loop learning process.

The plots below show the trajectories of the same disabled person after he has been trained by continuous display of path within the maze. The skill considered here is to learn the human actions (eg. movement) to achieve a certain goal in a certain task.

Figure 7.19 show the trajectory along the labyrinth for persons with disabilities after training them using the HMM. As seen from these trajectories there has been significant improvement in movement. The trajectory is much smooth without any knots

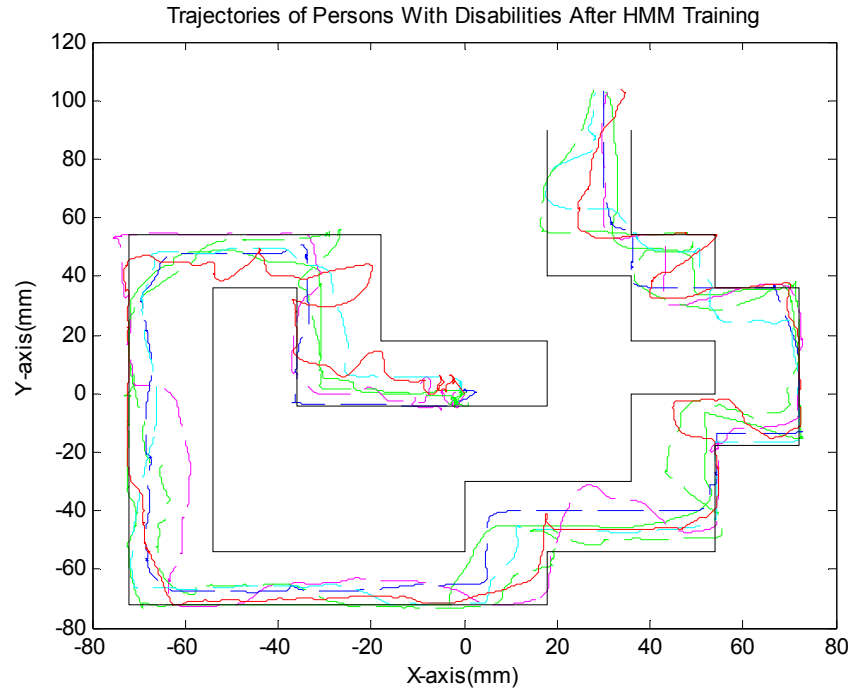


Figure 7.19 Trajectories of Disabled Persons After Training

The person has been able to pass the labyrinth in less time, without many collisions with the walls. The maximum impact force is also reduced to 1.89 N and as low as 0.446 N.

Figures 7.21 show the contact forces occurring with the walls for the subjects. In the same figures we see the collision information showing the number of collisions and the impact duration time which represents the longest time interval the ball pointer is in contact with the wall.

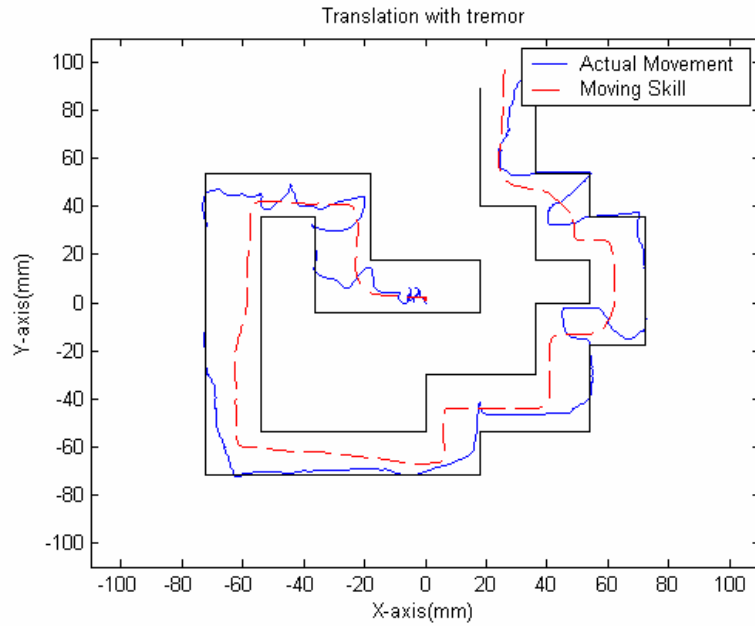


Figure 7.20 Trajectories After HMM Training

During the initial stages of the training process, the trajectory smoothness did not seem to improve significantly even though he had less collisions and time execution was less. This could be because of the person had quickly withdrawn the ball after the collision has occurred to follow his continuously updated trajectory.

The following trajectories are obtained when the test proceeded for many times. The operator had controlled his trajectory to avoid collisions and make his trajectory as smooth as possible. The person displayed some bodily movements to avoid unnecessary movements and postured himself accordingly.

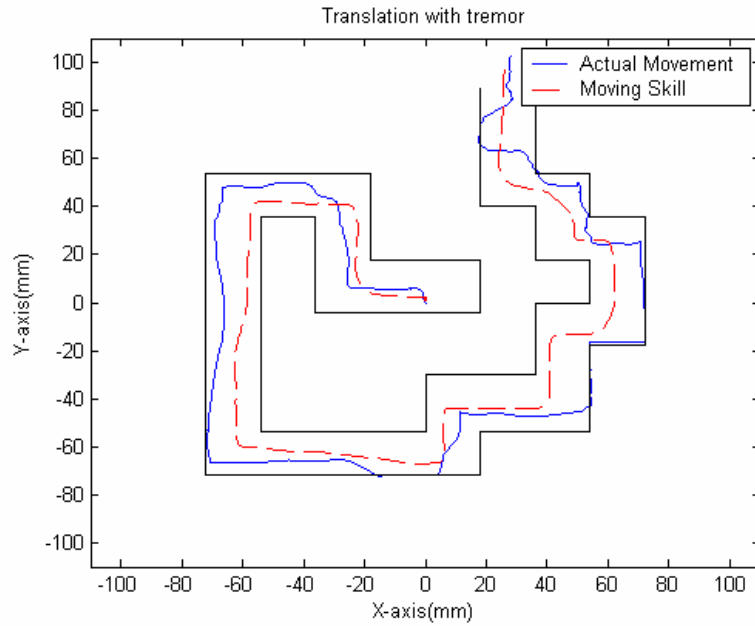


Figure 7.20 Continued

The following plots show the collision and force information. As expected the contact intervals were short. The maximum number of collisions were high as 16 and in once case even reduced to 5 in one case.

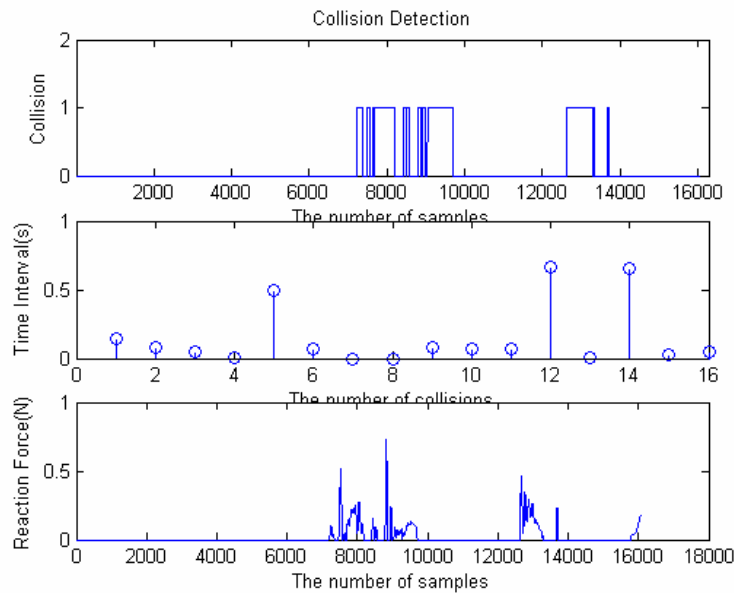


Figure 7.21 Collision/Force Plots of a Disabled Person After Training

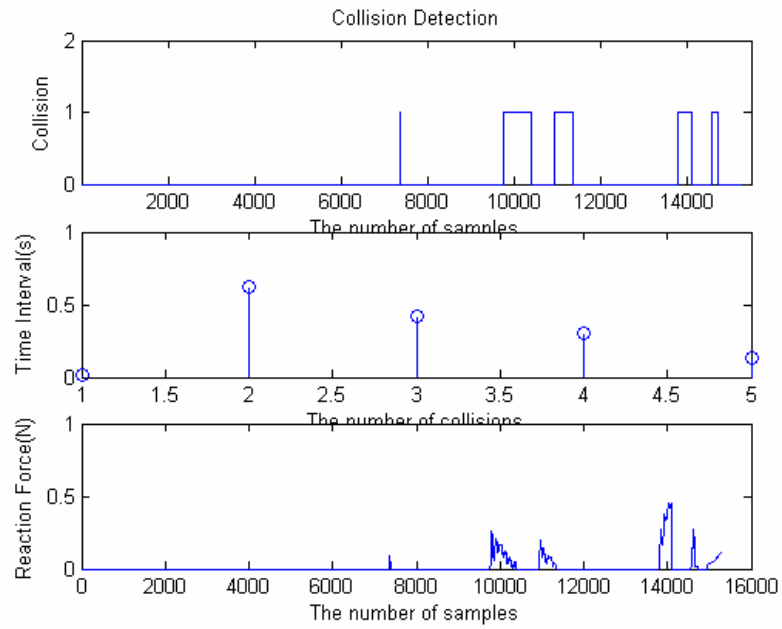
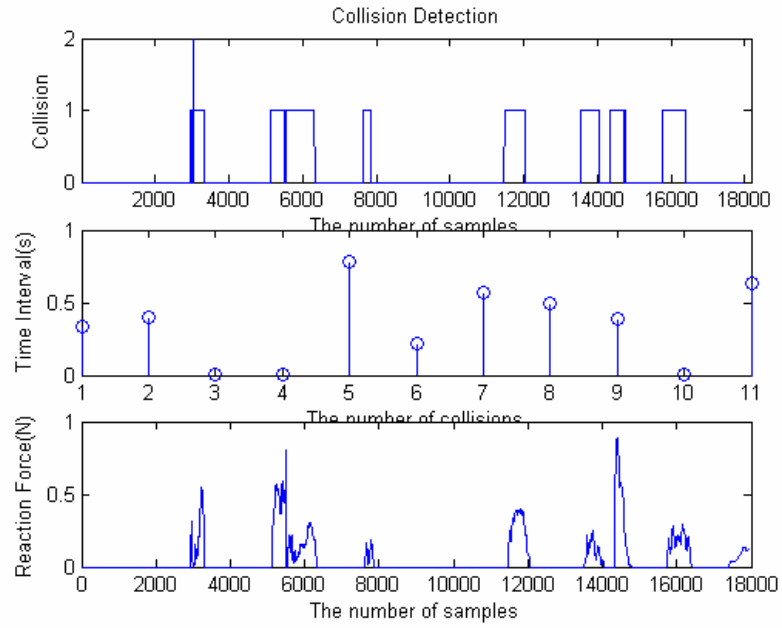


Figure 7.21 Continued

The tremor elimination algorithm has been implemented on all the above trajectories and the plots below show the paths after tremor elimination.

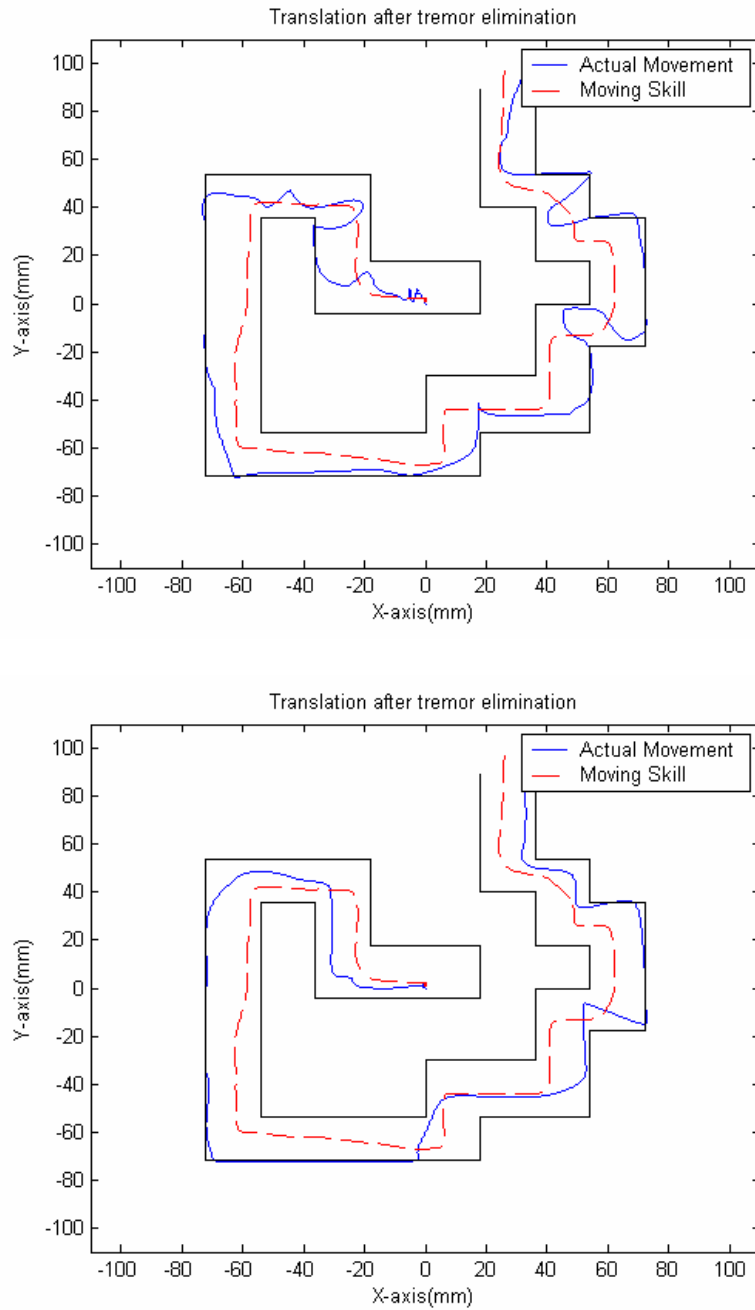


Figure 7.22 Trajectories of a Disabled Person After Training Without Tremor

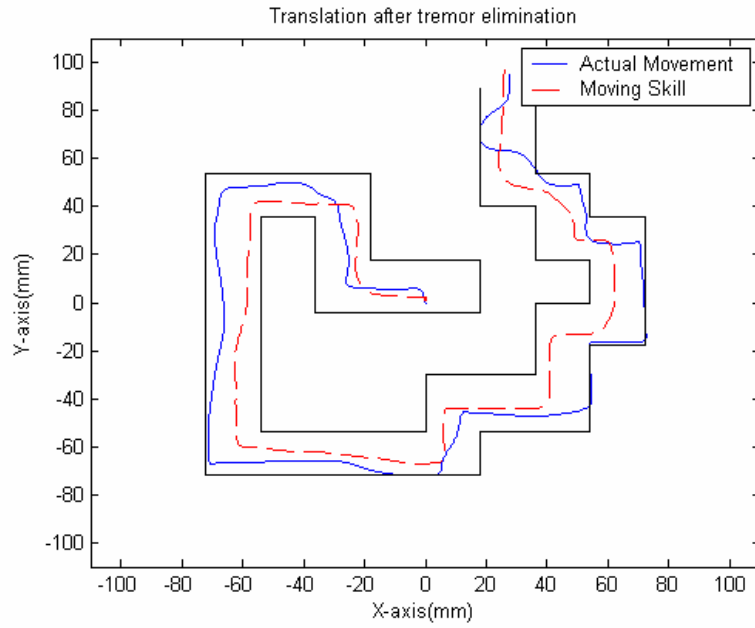


Figure 7.22 Continued

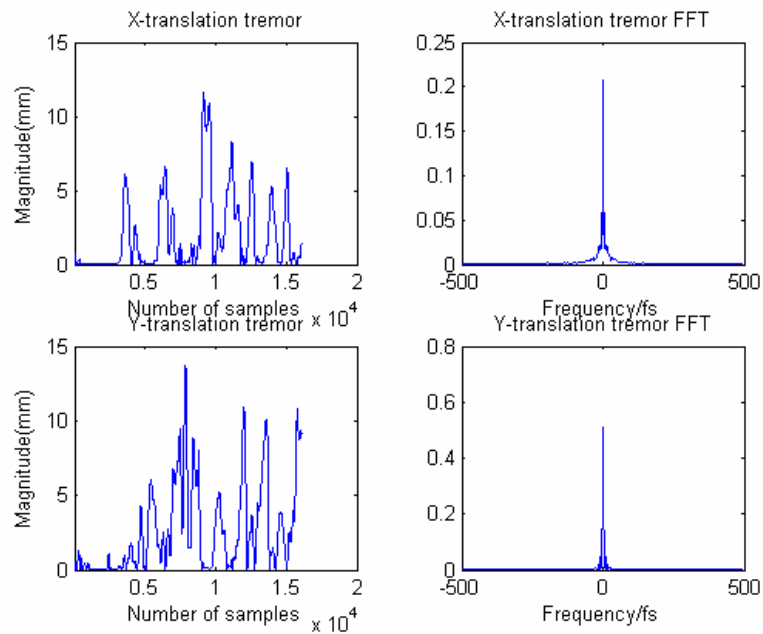


Figure 7.23 Tremor Plots After HMM Training

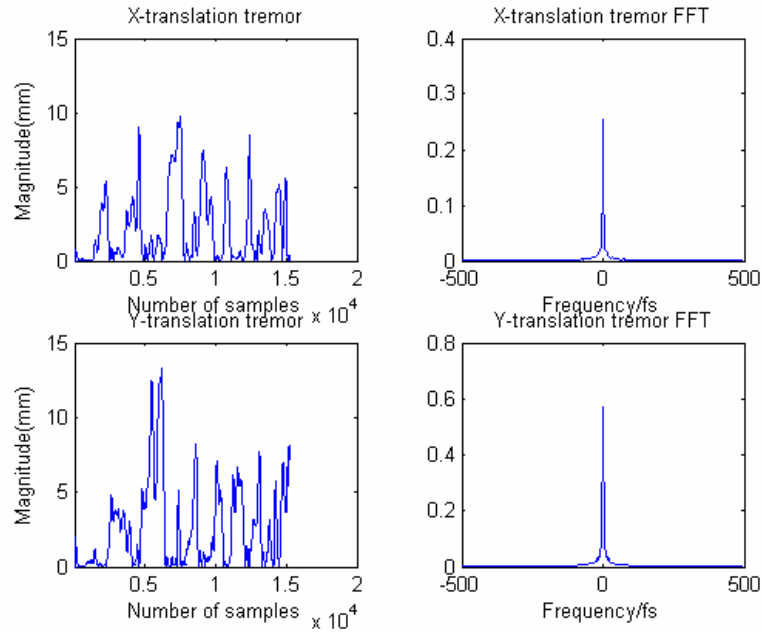


Figure 7.23 Continued

Table 7.7 shows the numerical results of all the experiments with HMM Skill Learning.

Table 7.7 Numerical Results for the Labyrinth After HMM Skill Learning

Number	Time Taken (sec)	Collisions	Max Impact Duration (sec)	Min Impact Duration (sec)	Max Force (N)
1	21.157	9	0.582	0.023	0.682
2	16.124	16	0.666	0.004	0.732
3	15.079	14	1.54	0.01	1.896
4	16.968	15	0.444	0.022	1.300
5	16.169	10	0.76	0.056	0.446
6	15.393	5	0.42	0.021	0.455
7	18.062	11	0.637	0.002	0.884
Std Dev	2.0873	3.867	0.381	0.0182	0.521
Mean	16.993	11.428	0.721	0.0197	0.913

The following table shows the tremor performance of both the subjects before and after therapy training.

Table 7.8 Results for the Tremor Analysis Before and After HMM Skill Learning

	Subject 1				Subject2			
	Before		After		Before		After	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
X Tremor mm	10.478	4.86	4.26	2.33	7.77	3.61	5.13	2.03
Y Tremor mm	10.212	6.73	6.432	2.16	8.42	4.34	5.43	1.92
Tremor Freq	3.5 Hz		2.1 Hz		2.8 Hz		2.0 Hz	

7.3.4 Discussion

The X-Y representations of the plots show the movement pattern among the patients before and after training. It offers a rough visual estimation for these tasks. The subjects showed better movement control based on the results in the tables 7.7 and 7.8. The patients were holding the haptic interface stylus with their fingers and their wrist was unsupported. This experiment can also serve as a test bed for the assessment of the limbs of patients in rehabilitation. Sequential tests could be performed to track the progress of the functional state process. The labyrinth complexity can be selected depending on the performance and skill. These functional assessments can play a key role in physical

therapy effects and helps the therapists to make important decisions about planning an optimal treatment for different persons. The movement ability can be compared between subjects.

7.4 Box and Blocks Test

Figures below presents the trajectories followed by different persons for a sequence of tests performed with the cube when no assistance was provided to the operator.

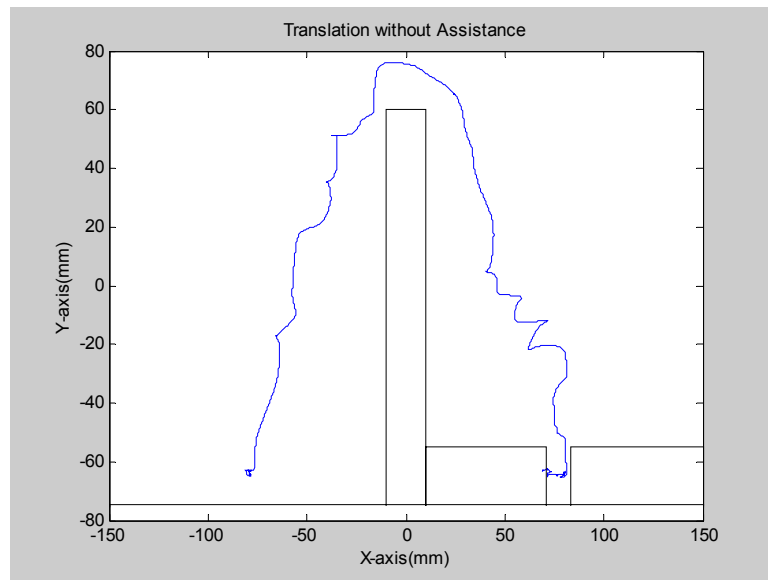


Figure 7.24 Box and Blocks Test - No Assistance Provided

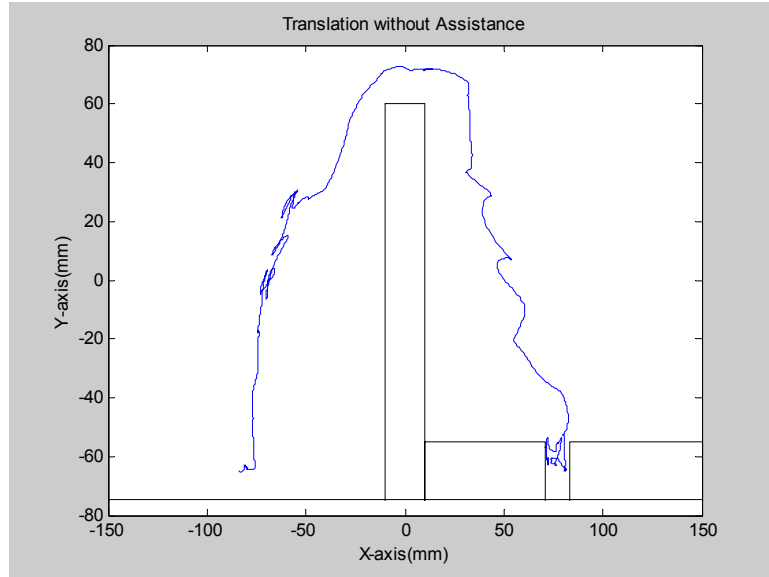


Figure 7.24 Continued

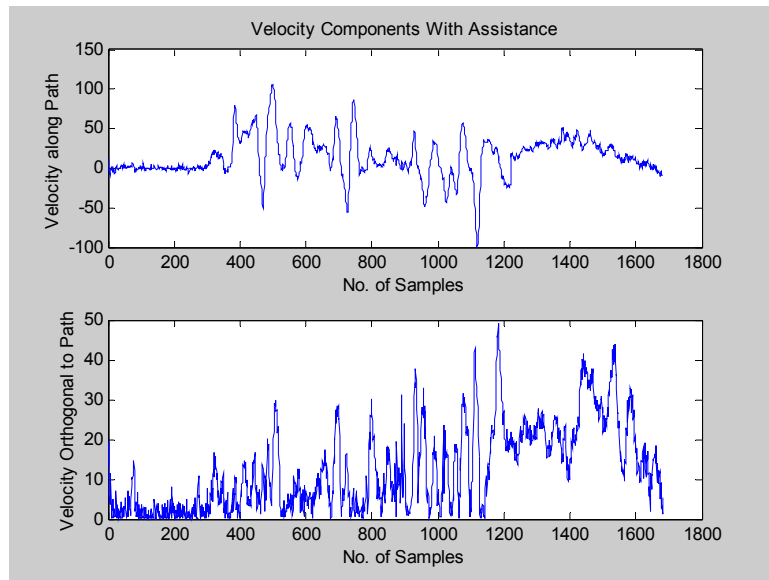


Figure 7.25 Box and Blocks Test – Velocity Components Without Assistance

In the tests performed without assistance the velocity component orthogonal to the path is not small when compared to the velocity components along the path. Also from the

plots of trajectories it can be seen that the user has much difficulty in going around the obstacle and wasted a lot of time giving unnecessary movements.

Figures below show the plots when the forms of Assistance Function explained in chapter 5 were provided. As mentioned before appropriate assistance is provided depending upon the classification of the stages. If the stage is path following a fixture is applied in order to move the end-effector along a certain path defined. It is noticed the difference in the mapping of the trajectories in each direction. If the motion is positioning at the target the above assistance is replaced by attractive force field. With these assistances the user executed the task for several times.

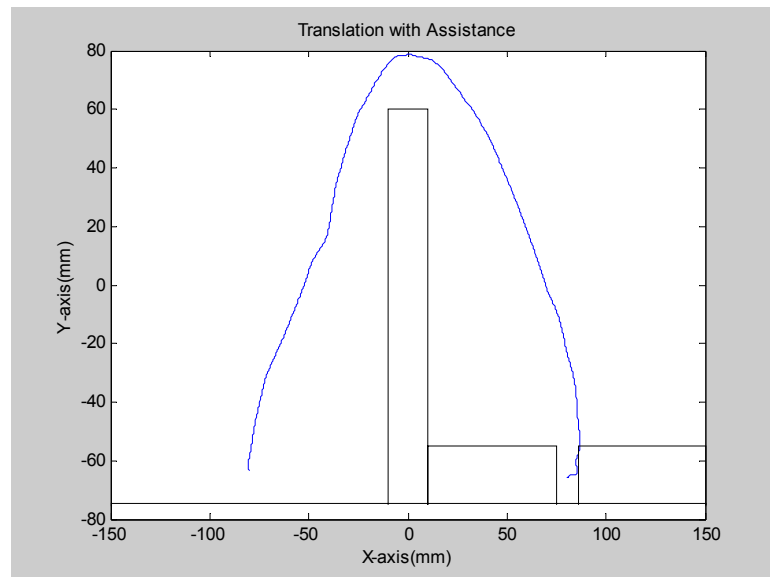


Figure 7.26 Box and Blocks Test - Assistance Provided

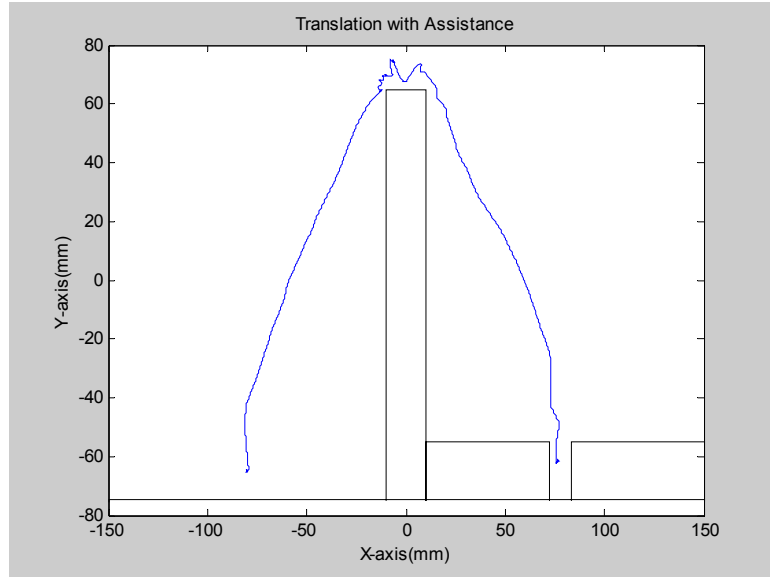


Figure 7.26 Continued

The trajectories in this case are much smooth and there is a significant reduction in the time execution. Figure 7.28 summarizes the execution times for all the tests.

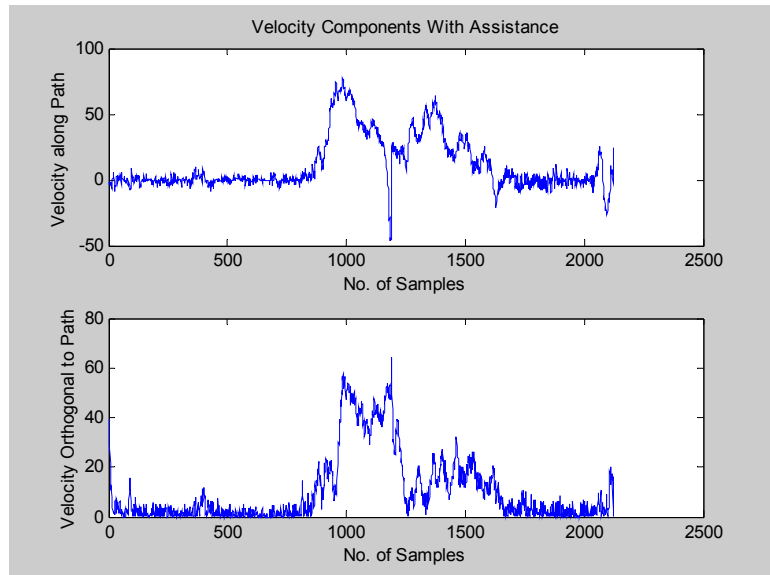


Figure 7.27 Box and Blocks Test – Velocity Components With Assistance

A sample of time executions for seven cubes is shown in Figure below. The data shows that the form of assistance function helped to reduce the execution times considerably.

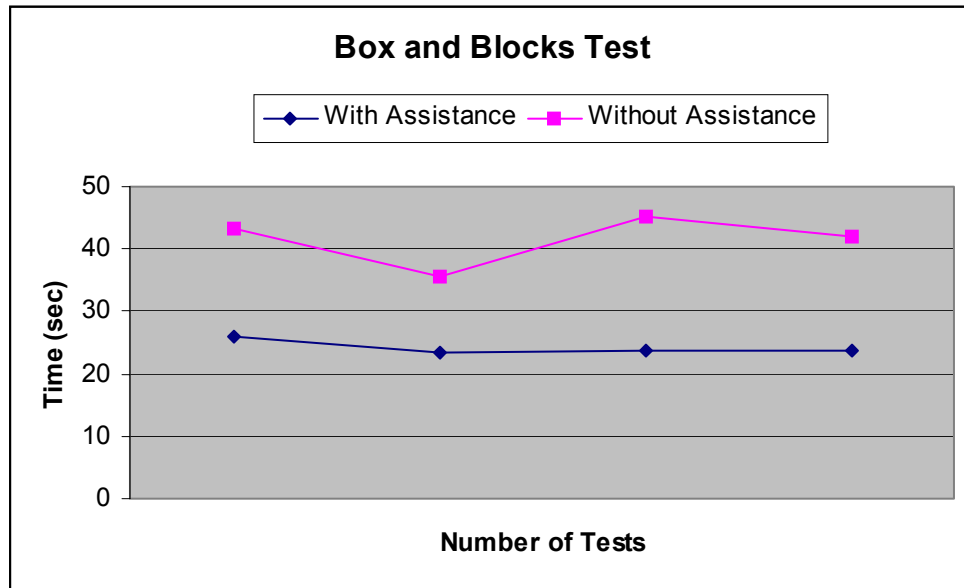


Figure 7.28 Box and Blocks Test - Execution Times

Chapter 8: Conclusions and Recommendations

This thesis demonstrates that training in virtual environment could potentially improve motion performance in the execution of various tasks in the areas of rehabilitation. We incorporated various forms of assistance for the execution of complex job related tasks by individuals with disabilities using a robotic haptic device.

The assistance functions implemented were based on the regulation of the mappings of positions, velocities and forces between of the Master and Slave robotic devices. These forms of assistance increased the operator's performance and helped reduce task execution times in a typical office virtual environment set-up. For the same set-up it was also shown that spatial performance of people with disabilities in a virtual environment could improve from training. The spatial information gained from these virtual environments can transfer directly to an equivalent real environment. At the present time, the virtual office set-up can have high potential as an efficient, cost-effective and scalable tool for conducting motion performance measurements beyond what exists using traditional methodologies. Additionally, the capacity to integrate measures of movement via the tracking technology (i.e. using Phantom) further adds value to this form of assessment when compared to traditional tests. It has also been shown that training in this virtual environment for the treatment sessions reduced the execution time by 50%.

The Hidden Markov Model approach to skill learning was implemented in simulation for the maze environment. This test-bed can also serve as a therapy platform to substitute a therapist's work in rehabilitation. So once the skill is learned, this labyrinth can be used for tremor reduction, upper limb coordination and motion control capabilities. The test is reliable, easy to perform and would be an indicator of the success of therapists work in rehabilitation process. The labyrinth presented here guarantees a wide range of test complexities. This test was applied to two subjects with cerebral palsy resulting in upper limb movement impairment as well as to two healthy subjects. A tremor filter was designed and implemented to cancel any tremor at the input of the haptic interface. The results indicated that the chosen tasks or similar tasks could be used to train individuals with disabilities, specifically those with disabilities involving pathological tremor and restricted range of motions. Sequential tests could be performed over large time intervals to track the motion improvements. This method also enables the therapists to make important decisions about planning an optimal treatment based on the recorded performance.

In the box and block virtual environment for motion recognition using HMM, the results verified that with the help of assistance all the undesired random errors are removed or reduced. This test tells us that HMM based assistance is useful for improving the performance accuracy and reducing the execution time. This motion recognition allows the computer to classify human actions and map the actions into corresponding tasks. If a person with disability is trying to approach an object or avoid an obstacle the computer can perform the judgment and appropriate assistance can be applied.

The system used in this investigation can be used as a model for future technology development that augments the performance of individuals with disabilities. The objective of choosing these manual dexterity tests was to assess the human-robot haptic system interface performance by subjecting it to tests that are regularly employed in the occupational therapy field. This could facilitate a better understanding between robotics designers and rehabilitation engineers. Also a good understanding of the range of motion and strength of the person is needed in order to make an accurate assessment of the type of assistance that would work best for patient. Nevertheless, both the disabled subjects expressed an interest and satisfaction with the philosophy behind these concepts and their potential applications.

References

- [1] R. Erlandson. "Applications of Robotic/Mechatronic Systems in Special Education, Rehabilitation Therapy and Vocational Training: A Paradigm Shift". IEEE Transactions on Rehabilitation Engineering. Vol 3, No 1, March 1995.
- [2] R. D. Jackson. "Robotics and its Role in Helping Disabled Persons". Eng. Sci. Education Journal. December 1993.
- [3] TIDE. "1993-1994 Work Plan" Commission of the Europe Communities, 1993.
- [4] S. A. Napper, R. L. Seaman. "Applications of Robots in Rehabilitation". Robotics Autonomous Syst. Vol. 5, pp 227-239, 1989.
- [5] R. C. Goertz. "Fundamentals of General Purpose Remote Manipulators". Nucleonics, Vol. 10, November 1982.
- [6] T. Sheridan. Telerobotics, Automation, and Human Supervisory Control. The MIT press, 1992.
- [7] S. Hayati and S. T. Venkatraman. Design and Implementation of a robot control system with traded and shared control capability, in Proceedings of the 1989 IEEE International Conference on Robotics and Automation, Scottsdale, AZ, pp. 1626-1631.
- [8] Paul G. Backes, "Multi-sensor based impedance control for task execution," in Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, May 1992, pp. 1245-1250.
- [9] Yasuyoshi Yokokohji, Akira Ogawa, Hitoshi Hasunuma, and Tsuneo Yoshikawa, "Operation modes for cooperating with autonomous functions in intelligent teleoperation systems," in Proceedings of the 1993 IEEE International Conference on Robotics and Automation, Atlanta, GA, May 1993, vol. 3, pp. 510-515.
- [10] G. T. Marth, Tzyh-Jong Tarn, and Antal K. Bejczy, "An event based approach to impact control: Theory and experiments," in Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, May 1994, pp. 918-923.

- [11] William S. Harwin, Tariq Rahman, and Richard A. Foulds, "A Review of Design Issues in Rehabilitation Robotics With Reference to North American research", IEEE Transactions on Rehabilitation Engineering, VOL. 3, NO.1, March 1995.
- [12] J.R. Allen, A.Karchak, Jr., and E.L. Bontrager, "Design and fabrication of a pair of Rancho anthropomorphic arm," Attending Staff Assoc. Rancho Los Amigos Hospital, Inc., and Tech. Rep., 1972.
- [13] John L. Dallaway, Robin D. Jackson, and Oaul H. A. Timmers, "Rehabilitation Robotics in Europe", IEEE Transactions on Rehabilitation Engineering, VOL. 3, NO.1, March 1995.
- [14] M. Topping, " 'Handy 1', a robotic aid to independence for severely disabled people," in Proc. 3rd Cambridge Workshop Rehabilitation Robotics, pp.13-16, 1994.
- [15] J.L. Dallaway and R.D. Jackson, "RAID a Vocational robotic workstation," in ICORR 92-conference proceedings, 1992.
- [16] H. F. Machiel Van Der Loos, "VA/Stanford Rehabilitation Robotics Research and Development Program: Lessons Learned in the Application of Robotics Technology to the Field of Rehabilitation," IEEE Transactions on Rehabilitation Engineering, VOL. 3, NO.1, March 1995.
- [17] Vijay Kumar, Tariq Rahman and Venkat Krovi, "Assistive Devices For People With Motor Disabilities", Wiley Encyclopaedia of Electronics Engineering" 1997.
- [18] M. L. Aisen, H. I. Krebs, N. Hogan, F. McDowell, and B. Volpe, "The effect of robot assisted therapy and rehabilitative training on motor recovery following stroke," *Arch. Neurol.*, vol. 54, pp. 443-446, 1997.
- [19] Myer Kutz, McGraw-Hill, "Biomedical Engineer's Handbook", 2002.
- [20] Peter S. Lum, Machiel Van Der Loos, Peggy Shor, Charles G. Burgar, "A Robotic System For Upper-Limb Exercises to Promote Recovery of Motor Function Following Stroke," in International Conference on Rehabilitation Robotics, Stanford, CA, U.S.A.
- [21] Toshiro Noritsugu and Toshihiro Tanaka," Application of Rubber Artificial Muscle Manipulator as a Rehabilitation Robot", in IEEE/ASME Transaction on Mechatronics, Vol.2, No.4, December 1997.
- [22] M.J. Johnson, H. F.M. Van Der Loos, C.G. Burgar, P. Shor, L.J.Leifer, "Designing a Robotic Stroke Therapy Device to Motivate Use of the Impaired Arm", in 7th International Conference on Rehabilitation Robotics, France, May 2001.

- [23] F. A. Mussa-Ivaldi, J.L. Patton, “ Robots can Teach People how to Move their Arm”, in Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, U.S.A., April 2000.
- [24] David J. Reinkensmeyer, Clifton T. Pang, Jeff A. Nessler, Chris C. Painter, “ Java Therapy: Web-based Robotic Rehabilitation”, in 7th International Conference on Rehabilitation robotics, France, May 2001.
- [25] H. F. Machiel Van Der Loos, “VA/Stanford Rehabilitation Robotics Research and Development Program: Lessons Learned in the Application of Robotics Technology to the Field of Rehabilitation,” IEEE Transactions on Rehabilitation Engineering, VOL. 3, NO.1, March 1995.
- [26] Vijay Kumar, Tariq Rahman and Venkat Krovi, “Assistive Devices For People With Motor Disabilities”, Wiley Encyclopaedia of Electronics Engineering” 1997.
- [27] Toshiro Noritsugu and Toshihiro Tanaka,” Application of Rubber Artificial Muscle Manipulator as a Rehabilitation Robot”, in IEEE/ASME Transaction on Mechatronics, Vol.2, No.4, December 1997.
- [28] M.J. Johnson, H. F.M. Van Der Loos, C.G. Burgar, P. Shor, L.J.Leifer, “ Designing a Robotic Stroke Therapy Device to Motivate Use of the Impaired Arm”, in 7th International Conference on Rehabilitation Robotics, France, May 2001.
- [29] David J. Reinkensmeyer, Clifton T. Pang, Jeff A. Nessler, Chris C. Painter, “ Java Therapy: Web-based Robotic Rehabilitation”, in 7th International Conference on Rehabilitation robotics, France, May 2001.
- [30] J. Hollerbach. “Some Current Issues in Haptics Research”. Proceedings of the 2000 IEEE International Conference on Robotics and Automation. San Francisco, 2000.
- [31] K. Maclean. “Designing with Haptic Feedback”. Proceedings of the 2000 IEEE International Conference on Robotics and Automation. San Francisco, 2000.
- [32] J. Schuyler, R. Mahoney. “Assesing Human-Robotic Performance for Vocational Placement”. IEEE Transactions on Rehabilitation Engineering. Vol. 8, No 3, September 2000.
- [33] G. Bolmsjo, H. Neveryd, H. Efring. “Robotics in Rehabilitation”. IEEE Transactions on Rehabilitation Engineering. Vol 3, No 1, March 1995.
- [34] C. Stanger, C. Angling, W. Harwin, D. Romilly. “Devices for Assisting Manipulation: A Summary of User Task Priorities”. IEEE Transactions on Rehabilitation Engineering. Vol 2, No 4, December 1994.

- [35] Steven E. Everett, Rajiv V. Dubey. "Model-based variable position mapping for Telerobotic Assistance in a Cylindrical Environment". Proceedings of the IEEE International Conference on Robotics and Automation, Detroit, MI.. May 1999, pp. 2197-2202.
- [36] R. Dubey, K. Manocha, N. Pernalet. "Variable Position Mapping Based Assistance in Teleoperation for Nuclear Clean Up. The International Conference on Robotics and Automation (ICRA) 2001. Seoul, Korea. May 21-26, 2001.
- [37] R. Dubey, S. Everett, N. Pernalet, K. Manocha. "Teleoperation Assistance Through Variable Velocity Mapping". Accepted for October 2001 publication at the IEEE Transactions on Robotics and Automation.
- [38] K. Kawamura, S. Bagchi, M. Iskarous, M. Bishay. "Intelligent Robotic Systems in the Service of the Disabled". IEEE Transactions on Rehabilitation Robotics, Vol. 3. No. 1, March 1995.
- [39] Burdea, Popescu, Hentz and Colbert. "Virtual Reality-Based Orthopedic Telerehabilitation". IEEE Transactions on Rehabilitation Engineering, Vol.8, No.3, September 2000.
- [40] Kazuhiro Kosuge, Koji Takeo, and Toshio Pukuda, "Unified approach for teleoperation of virtual and real environment manipulation based on reference dynamics " in Proceedings of the 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan, May 1995, pp. 938-943.
- [41] S. E. Everett. "Human-Machine Cooperative Telerobotics Using Uncertain Sensor and Model Data". Doctor of Philosophy in Mechanical Engineering, University of Tennessee, August 1998.
- [42] Robotics Research Corporation, P.O. Box 206, Amelia, OH 45102. Robotics Research Servo Level Interface Users Manual, Rev. 0490, 1990. Publication UM-400-90.
- [43] J. Schuyler, R. Mahoney. "Job Identification and Analysis for Vocational Robotics Applications". Proceedings RESNA 1995.
- [44] Sensable Technologies. <http://www.sensable.com/products/phantom.htm>
- [45] GHOST® SDK Programmers Guide Version 4.0.
- [46] Rehabilitation Engineering and Technology Program, University of South Florida. <http://retp.eng.usf.edu>