8-11-2004

Intelligent Telerobotic Assistance For Enhancing Manipulation Capabilities Of Persons With Disabilities

Wentao Yu
University of South Florida

Intelligent Telerobotic Assistance For Enhancing Manipulation Capabilities Of Persons With Disabilities

by

Wentao Yu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Mechanical Engineering
College of Engineering
University of South Florida

Major Professor: Rajiv V. Dubey, Ph.D.
Glen Besterfield, Ph.D.
Daniel Hess, Ph.D.
Shuh-Jing Ying, Ph.D.
Wilfrido A. Moreno, Ph.D.
A.N.V. Rao, Ph.D.

Date of Approval:
August 11, 2004

Keywords: Rehabilitation, Hidden Markov Model, Motion Intention Recognition , Virtual Fixture, Skill Learning, Therapy

Acknowledgements

First, I would like to thank Dr. Rajiv Dubey for being a great professor and giving me the opportunity to pursue research work in an area that I enjoy; I will never forget the opportunity that you gave me. I would also like to thank Dr. Glen Besterfield, Dr. Daniel Hess, Dr. Shu-Jing Ying, Dr. Moreno and Dr. Rao for serving on my advisory committee.

I would like to thank Aaron Gage for setting up the Ghost SDK for the PHANToM, and making some software for our system. I would like to thank Norali Pernalete for making such an interesting previous research for the lab. Michael Jurczyk provided invaluable assistance in setting up the hardware. He got involved with the vision system getting the camera to work. His work led to the Halcon software configuration. I would like to thank Dwayne Polzer for attaching the sensors on the end-effector. I would like to thank Redwan Alqasemi for being a good friend and making careful correction on my dissertation. Thank you for your time and hard work.

I would also like to acknowledge the financial support for the rehabilitation robotics research which I have been involved in the past few years. Stephen Sundarrao and his Rehabilitation Engineering and Technology Program provided the funding for research work I did in this lab.

Finally, my thanks go to the lab members that I have met: Ben Fritz, Ed McCaffrey, Sashi Konda, Ashwin Upadhyay, Kevin Edwards etc.

Table of Contents

List of Tables

List of Figures

Intelligent Telerobotic Assistance for Enhancing Manipulation Capabilities of Persons with Disabilities

Wentao Yu

ABSTRACT

This dissertation addresses the development of a telemanipulation system using intelligent mapping from a haptic user interface to a remote manipulator to assist in maximizing the manipulation capabilities of persons with disabilities. This mapping, referred to as assistance function, is determined on the basis of environmental model or real-time sensory data to guide the motion of a telerobotic manipulator while performing a given task. Human input is enhanced rather than superseded by the computer. This is particularly useful when the user has restricted range of movements due to certain disabilities such as muscular dystrophy, a stroke, or any form of pathological tremor.

In telemanipulation system, assistance of variable position/velocity mapping or virtual fixture can improve manipulation capability and dexterity. Conventionally, these assistances are based on the environmental information, without knowing user's motion intention. In this dissertation, user's motion intention is combined with real-time environmental information for applying appropriate assistance. If the current task is following a path, a virtual fixture orthogonal to the path is applied. Similarly, if the task is to align the end-effector with a target, an attractive force field is generated. In order to successfully recognize user's motion intention, a Hidden Markov Model (HMM) is developed.

This dissertation also describes the HMM based skill learning and its application in a motion therapy system in which motion along a labyrinth is controlled using a haptic interface. Two persons with disabilities on upper limb are trained using this virtual therapist. The performance measures before and after the therapy training, including the smoothness of the trajectory, distance ratio, time taken, tremor and impact forces are presented.

The results demonstrate that various forms of assistance provided reduced the execution times and increased the performance of the chosen tasks for the disabled individuals. In addition, these results suggest that the introduction of the haptic rendering capabilities, including the force feedback, offers special benefit to motion-impaired users by augmenting their performance on job related tasks.

Chapter 1: Introduction

1.1. Motivation

Physical disabilities make it difficult or sometimes impossible for individuals to perform several simple job related tasks such as pressing a button to operate a machine, moving light objects etc. While considering employment, the true potential of individuals with disabilities can be enhanced by technology to augment human performance. New developments in telerobotic systems can allow greater number of individuals with disabilities to compensate for their lost manipulation skills. In the past two decades, researchers in rehabilitation robotics have designed and developed a variety of passive/active devices to help persons with limited upper-limb functions to perform essential daily manipulation tasks. Since the user is inside the control loop, most of these research or commercial products have adopted telemanipulation system, in which the user issues robot motion commands through an interface [3]. However, practical results are limited, mainly due to the fact that although telemanipulation may relieve the user of the physical burden of manipulative tasks, it introduces the mental burden of controlling the input device [4]. With typical telemanipulation, the user is in the control loop, sensing the environment information such as the location and the distance of the target and providing the appropriate control signal to the input device. In literature [84], after training all operators for a certain time (normal subjects), only 60% of them were skilled enough to complete teleoperation tasks. A general method for introducing computer

assistance in task execution without overriding an operator's command to the manipulator is used. The appropriate movement for the task is kept or even enhanced, but the undesirable movements are reduced. This is done using assist functions, which scale the input velocity according to the task. This methodology has been previously employed by the author in the execution of manual dexterity assessment tasks with fully able individuals [53].

Beside this functional approach in rehabilitation, robotics applications can also assist clinically in therapy. 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.

1.2. Dissertation Objective

The goal of this dissertation is to design an intelligent telerobotic system that can maximize the manipulation capabilities and reduce the mental burden for persons with disabilities on the upper-limb:

1. Develop sensor-based assistance functions to increase the limited motion range and enhance manipulation accuracy.
2. Implement these assist functions to perform a common vocational rehabilitation test referred to as a Box and Blocks. During task operation, adjust the scaling according to the available sensory data.

3. Develop an algorithm to recognize operator's motion intention by using Hidden Markov Model (HMM). Apply appropriate fixture assistance based on operator's motion. If the recognized motion is following a path, a virtual fixture orthogonal to the path is applied. If the task is to align the end-effector with a target, an attractive force field is generated. Similarly, if the task is to avoid obstacles, a repulsive force field is produced.
4. Develop a robotic therapy system based on skill learning through Hidden Markov Model. Since HMM is feasible to model a stochastic process, such as speech or a certain assembly skill, it can be used to characterize the skill of moving along a labyrinth path. The skill of moving along a labyrinth is learned and considered as a virtual therapist, which replaces the role of a physical therapist for motion therapy. Perform motion experiments with two subjects with disabilities.

The contribution of this dissertation is that telerobotic system with intelligent operation can enhance the manipulation capabilities and reduce the mental burden, and learned skill of a specific task can be used as a robotic therapist to do motion therapy.

1.3. Dissertation Outline

The history and the background of rehabilitation robotics and telemanipulation system areas related to this work are discussed in chapter 2. The concept of rehabilitation robotics, haptic interface and teleoperation assistance are traced through history to the present state of knowledge in these areas. Chapter 3 describes a telemanipulation system to assist persons with disabilities perform dexterous manipulation tasks. In this chapter,

assistance functions are used for mapping such that human input is enhanced and "Box and Blocks" is chosen to test the effectiveness of this sensor-based assistance function. The Hidden Markov Model (HMM)-based human motion intention recognition is developed in chapter 4 and then the implementation of appropriate virtual fixture assistance is applied to teleoperation. Chapter 5 describes the Hidden Markov Model based skill learning and its application in motion therapy system using a haptic interface. Chapter 6 concludes with a discussion of the experimental results, and suggested future work.

Chapter 2: Background

2.1. Rehabilitation Robotics

Physical and cognitive disabilities make it difficult or impossible for individuals to perform several simple work and household tasks such as pressing a button to operate a machine, opening a door, moving light objects etc. A study by J. Schuyler *et al* concluded that a slight increase in manipulation ability, mobility and strength results in substantial increase in the number of jobs for which an individual might be eligible [31]. In many instances, such enhancements may mean the ability to do a task that the person is otherwise unable to perform. Assistive devices have attempted to fully or partially restore the lost functions and enable people with disabilities to perform many Activities of Daily Life (ADL) affecting their employment and quality of life [1, 7, 3, 4, 17].

The earliest research in this area (prosthetics and robotic arms) began in the late 1960s [2]. The Rancho "Golden" arm, developed at Rancho Los Amigos Hospital in Downey, California in 1969 was the first successful rehabilitation robot manipulator [32]. It used seven tongue switches in a sequential mode to successfully maneuver the arm in space. Johns Hopkins arm [1, 5], evolved from prosthetics, could execute tasks in pre-programmed and direct modes through a chin manipulandum and other body-powered switches. The Heidelberg Manipulator was the earliest example of the workstation-based approach to the implementation of robotic systems [6, 7]. Spartacus project proposed that mounting a manipulator arm on a wheelchair would increase the effectiveness of

manipulation rehabilitation [8, 9]. Though all these assistive devices saw limited use by consumers, they established the foundation for further research.

Figure 2.1 RAID Workstation

Figure 2.2 Manus Manipulator

Since the 1980's, considerable progress has been made in the field of rehabilitation robotics technology. One example is the workstation robotic device. The goal of a workstation robotic device is to enable the user to perform tasks typically encountered in office or at home. These tasks include moving books from a shelf to a reading board, opening the book and flipping through its pages, inserting CD-ROMs and floppy diskettes into a computer. The most commonly used robotic workstation available to users with disabilities is the RAID (Robot for Assisting the Integration of the Disabled, Figure 2.1) workstation [12]. DEVAR (desktop assistant robot for vocational support in office settings) [16] can be used to handle paper, floppy disks, pick up and use the telephone, and retrieve medication. RAA (Robotic Assistive Appliance) offers a human size manipulator at a workstation with 6 degrees of freedom with either programmed or direct control [17] and is currently undergoing testing to assess its advantages over an attendant [18]. The other kind of device is wheelchair-mounted robot. A power wheelchair is used as a mobile base where a mechanical manipulator can be attached. Several wheelchair-mounted manipulators are available to the consumer, but two in particular, MANUS and the Raptor, are more successful. MANUS is the most well known of those successors (Figure 2.2). Raptor manipulator is the first robot assistive manipulator that has gained FDA approval for use in the US [35] (Figure 2.3). Because of its increased size, though, the range of the Raptor is 120 cm compared to the 80 cm of the Manus. It can also lift up to 2.5 kg. Another project that has enjoyed relative success is the Handy 1 [7,11], which was primarily used as a feeding device for children with cerebral palsy. More recently, besides improving eating skills, the aid has been considered for other activities including application of cosmetics leisure activities [26].

Figure 2.3 Raptor Manipulator

In addition, in FRIEND Robot arm system [15], a multimedia user interface was included to enlarge the functionality of existing technical aids. ISAC incorporated Artificial Intelligence (AI) into its controller to reduce the mental load on the user during the performance of manipulative tasks [20]. KARES uses a SPACEBALL 2003 as an input device to teleoperate the robotic arm [21]. In KAREA II, an advanced version of KARES has a visual servo, which allows the robotic arm to operate autonomously through the visual feedback of a binocular camera head [28].

The robot arm workstations or wheelchair-mounted manipulator above compensated for the activity deficiencies of people with disabilities. But because of the high cost, the poor interface between a complex electromechanical system and a person

with limited capabilities, and social stigma attached with a robot, these assistive devices have had limited success as commercial products [1,3,4,7].

Besides assistive robots, another type of rehabilitation robotic system is therapy robot. MIT-MANUS (Figure 2.4 (a)) is the most successful robot-aided therapy platform to undergo intensive clinical testing [85, 86]. This device is a planar, two-revolute-joint, backdriveable 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. The results of clinical trials suggested that exercise therapy improved motor recovery [87-89].

(a)

(b)

Figure 2.4 (a) MIT-MANUS[85], (b) MIME[16]

MIME (Figure 2.4 (b)) is powerful enough to move a patient's arm throughout the three-dimensional workspace against gravity [79]. When the patient moves her/his 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 [16]. The ARM (Assisted Rehabilitation and Measurement) was designed to guide reaching movements across the workspace, and to measure multi-axis force generation and range of motion of the arm [79]. Like MIT-MANUS and MIME, the ARM device can assist or resist movements and can also measure hand movements. The ARM Guide has been used to quantify and understand abnormal coordination, spastic reflexes, and workspace deficits after stroke [90]. The testing results suggested that the constraint force and range of motion measurements during mechanically guided movement may prove useful for precise monitoring of arm impairment and of the effects of treatment techniques targeted at abnormal synergies and workspace deficits [91, 92].

2.2. Telerobotics

Due to the unstructured environment of ADL and varieties of the tasks and the presence of the user, many rehabilitation robots adopt telerobotics systems so that users can issue commands through a human-machine interface [8, 11, 15, 28]. Regarding teleoperation studies, several types of systems and concepts have been defined in the area of remote manipulation technology [39]. The concept developed by Ray Goertz in the

1950's, in which a person's sensing and manipulation capability is extended to a remote location, is referred to as “teleoperation”. His mechanisms were mechanical pantograph devices which allowed radioactive materials to be handled at a safe distance. Later, electrical servos replaced mechanical linkages and cameras replaced direct viewing, so that the operator could be arbitrarily far away. Human operators look at video displays, and operate remotely located slave robot via a hand controller. Usually the term teleoperation refers to systems in which the human operator directly and continuously controls the remote manipulator. In these systems, the kinematic chain which is manipulated by the operator and may provide force feedback is referred to as the “master”, while the remote manipulator is referred to as the “slave”.

From the point of view of autonomy, telerobot is classified into tele-autonomy and tele-collaboration [57]. The former term refers to the combination of teleoperation and autonomous robotic control. In some cases, a unilateral controller is used. In this case, there is no information feedback from slave to master or from master to human. The latter means all operations are controlled by the human-machine collaboration, usually in the form of force reflection. For teleoperation itself, it can be classified into unilateral and bilateral telerobotics according to the data flow. In the former case, the slave robot is operated in free teleoperation, just like an open-loop system. The only feedback is the task execution video of the slave or even no video if the master and slave are in the same room. This case is illustrated in figure 2.5 (upper part). The latter one has force feedback provided to the teleoperator, thus forming a “kinesthetic” or “tele-presence” system [33, 34, 37, 73]. Figure 2.6 shows the architecture of a typical bilateral teleoperation. In this case, strategies in which human decisions are merged with computer-based assistance

have been made possible by more complex forms of automatic control and sensor data fusion. The control system adds computer-generated velocity/force inputs to those from the master in the impedance-controlled formulation to assist controlling the motion of the manipulator, such as moving along a surface without impact and obstacle avoidance. Bilateral impedance control in telerobotic systems provides good teleoperation since force reflection is provided to the operator during operation [33, 36, 39]. Dubey et al proposed variable impedance parameters to adapt to variable circumstances thus overcoming the conflict problem of choosing desired dynamics parameters [34]. This controller is primarily used in tasks requiring contact, such as needle inserting into tissue, object surface exploration.

Teleoperation system design usually takes operation accuracy into account, not the convenience and simplification of operation. With the improvement of the controller architecture and assistance attempt, the task performance of telerobotic system in rehabilitation engineering is still not satisfactory [40, 41, 44]. For a simple "go get a cup and put it on a pad" task, it takes the operator 50 seconds, mostly due to the indexing the master once the master reaches its workspace limit and tuning the gripper to grasp the target [53]. Furthermore, the performance largely depends on the operator's familiarity with the system. In most cases, using a robot as a teleoperated device to complete a task is much harder than using human arm and hand. It can soon become very exhausting, especially if it has to perform repeated tasks such as feeding, even with some assistance. Many researchers tried to improve the operation accuracy, reduce execution time and relieve the operator's mental labor through adding artificial intelligence. Kawamura et al [51] looked at how far rehabilitation robots had come in possessing abilities that relieve

the user from the mental burden of controlling the robot. They had developed modules for fuzzy commands interface, object recognition and task planning. In intelligent telerobot system, vision-based assistance has improved the operation of aligning the end-effector with the target [45, 50].

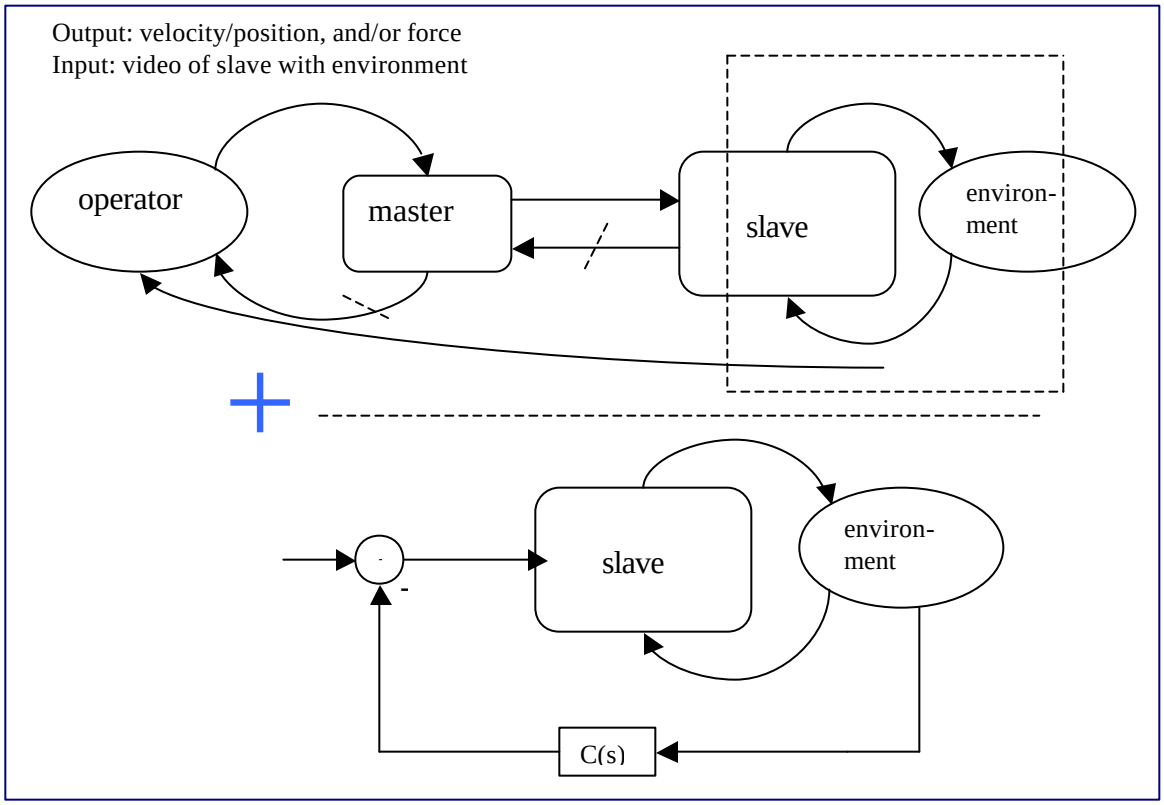


Figure 2.5 Tele-autonomy is the Combination of Teleoperation and Autonomy

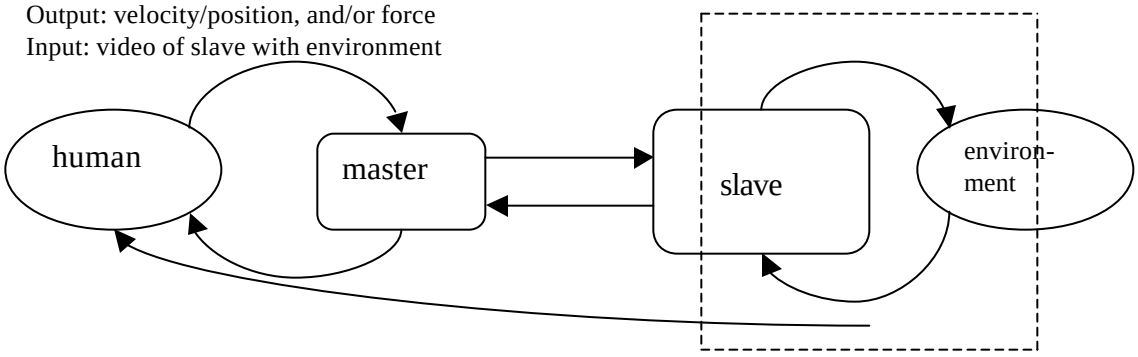


Figure 2.6 Tele-collaboration with Information Feedback

The telerobot emphasized in this dissertation, is the open loop telemanipulation with assistance. The challenge is to make it more functional and more intelligent. This dissertation is an attempt to address the issue of combining human flexibility and machine intelligence into an efficient rehabilitation robotic system.

2.3. Teleoperation Assistance Background

In teleoperation, it is essential to provide as much assistance as possible for the operator. Basically, the assistance algorithm is to map the master commands to the slave in a way that scales up or down depending on the task and environment information. The scaling factors vary according to the tasks and environment. The idea behind the assistance function concept is the generalization of position and velocity mappings between master and slave manipulators of a teleoperation system. This concept was conceived as a general method for introducing computer assistance in task execution without overriding operator's commands to the manipulator (Figure 2.7). The assistance functions can be classified as regulation of position, velocity and contact forces. All of these assistance strategies are accomplished by modification of system parameters. A simple form of position assistance is scaling, in which the slave workspace is enlarged or reduced as compared to the master workspace. The velocity assistance is commonly used in approaching target and in avoidance of obstacles. In both cases, the velocity scaling varies according to whether motion in that particular direction is serving to further the desired effect of the motion.

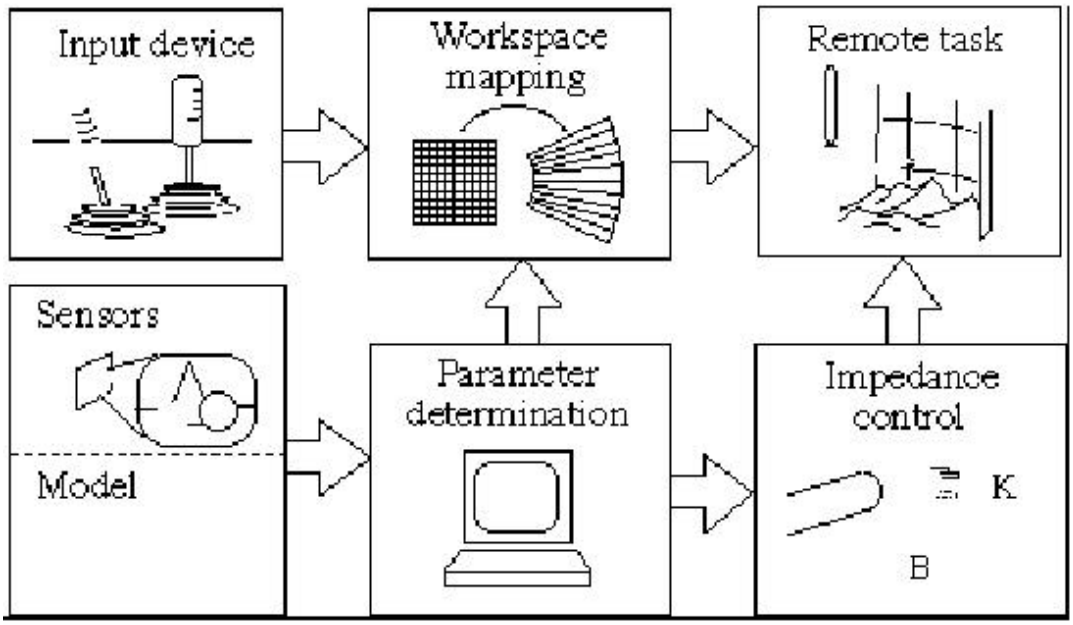


Figure 2.7 Human -Machine Cooperative Teleoperation Concept [29]

2.3.1. Regulation of Positions

In these functions, the motion of the manipulator is constrained to lie along a given line or in a plane. This helps persons with disabilities operate more stably and smoothly. The details of these functions were presented in a different work by the authors [67] (See Figure 2.8).

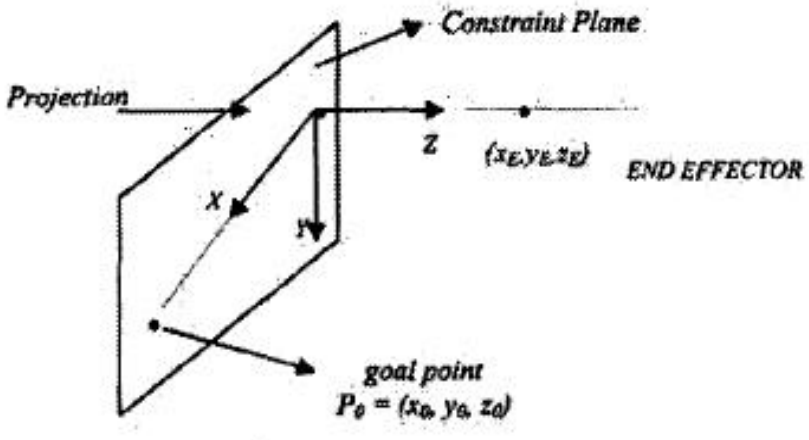


Figure 2.8 Representation of Slave Constraint Frame in the Constraint Plane [67]

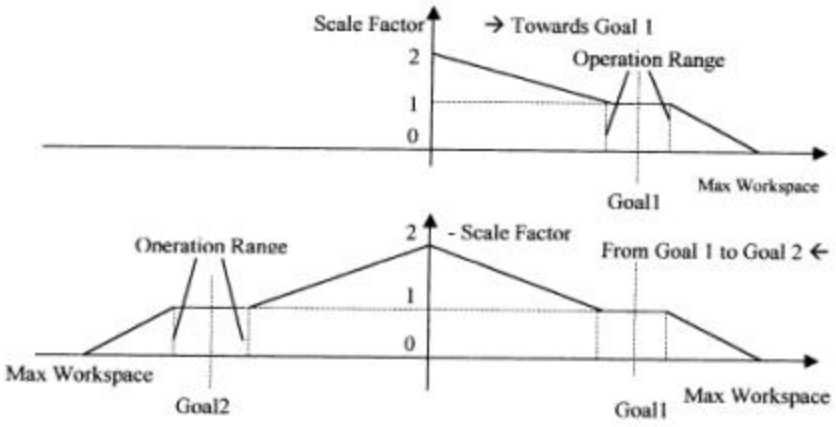


Figure 2.9 Scaling Factor Function [53]

2.3.2. Regulation of Velocities

In this case the mapping between the master and slave is done based on velocities. The velocity scaling used varies according to whether the motion in a particular direction is serving to further the desired effect of the motion. In the approach assistance, the velocity is scaled up if the motion reduces the distance between the current and goal positions of the manipulator. Otherwise, the velocity is scaled down. For velocity regulation, the scaling factor's changing is depicted in Figure 2.9. The scaling factor depends on the subtask being executed and the direction of travel. The relationship between the master/slave velocities is: $V_{slave} = ScaleFactor \cdot V_{maste}$. Figure 2.10 shows a velocity scaling factor varying based on the distance reading when the end-effector is approaching a wall.

Using a vision system, Everett designed a vision-based mapping to align the end-effector of the slave manipulator with a cross object [29, 45]. The velocities that reduce the alignment error are scaled up and the ones that increase the alignment error are scaled down (Figure 2.11).

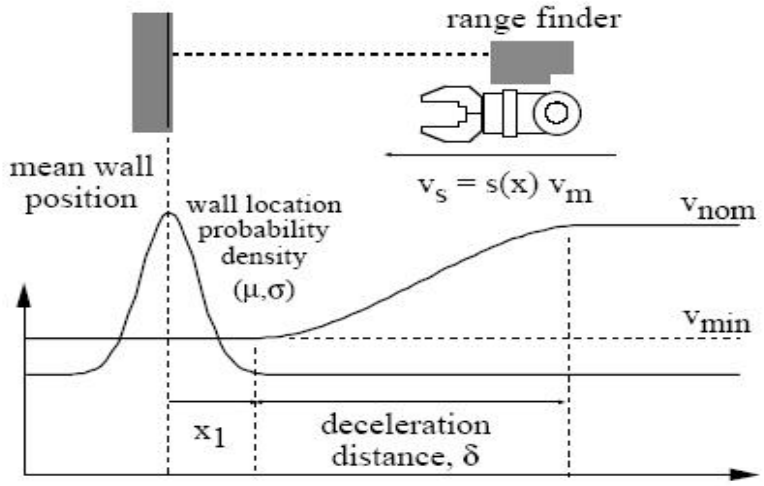


Figure 2.10 Scaling Factor Varying for Approach [29]

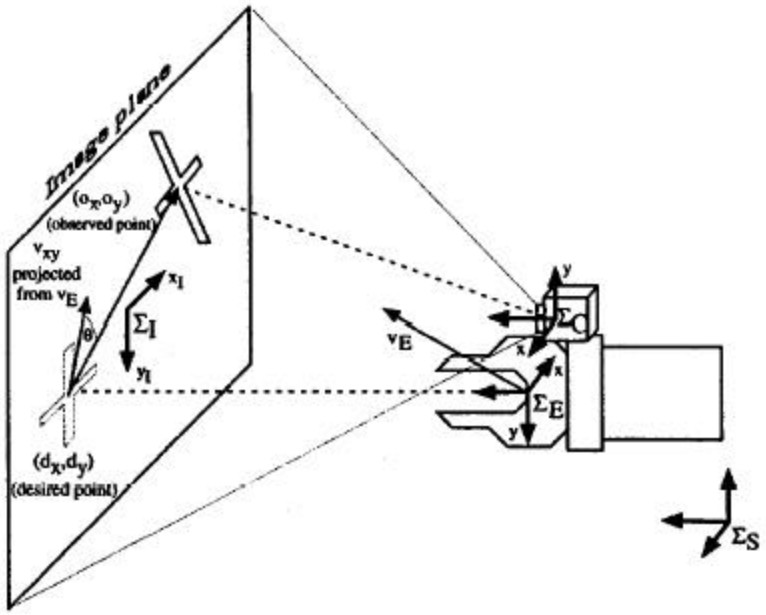


Figure 2.11 Coordinate Frames for Cross Alignment Task [29]

In tele-collaboration, another type of assistance is "virtual fixture". This assistance is functions of spatial parameters, instead of time. But what is virtual fixture? Virtual fixtures are defined, according to [68], as "abstract precepts overlaid on top of the

reflected sensory feedback from a remote environment such that a natural and predictable relation exists between an operator's kinesthetic activities and (efference) the subsequent changes in the sensations presented (afference)". Intuitively, it is very easy to understand this. As a matter of fact, everyone has experience of using a real fixture, for example, drawing a straight line using a ruler. By pressing your pencil against this "fixture", we are able to quickly draw a very straight line. Now imagine if there was no ruler there, but there was a virtual wall you could press against instead of a ruler. Similarly, what if there were invisible forces pulling on your pencil, forcing it to follow a straight path. These are virtual fixtures. Virtual fixtures play the same role in robot motion as they do in our line drawing motion. As a matter of fact, virtual fixture is a computed-generated constraint that displays position or force limitations to a robot manipulator or operator. It can be used to constrain the manually controlled manipulator's motion on a desired surface or to be pulled into alignment with a task [37, 38, 61, 64]. Usually, two stiffness coefficients are defined: stiffness along the desired path and stiffness orthogonal to the path. The ratio between these two stiffness coefficients indicates the softness or hardness. If the ratio is close to zero, it is the hardest fixture, which means that end-effector can only move along the path, not deviating at all. If the ratio is close to 1, it is the softest fixture, where the end-effector can move freely. So this kind of fixture is usually used for path following (Figure 2.12).

Virtual fixture can also be in the form of potential force fields [68, 69]. Potential fields were used to produce velocity commands, which, when added to those generated by the input device, maneuver the manipulator toward the target or away from obstacles [69]. Force field is usually in the magnetic form. The role of this type of fixture is the

same, guiding the end effector into a goal or away from an obstacle. Figure 2.13 shows that extract and insert fixtures restrict the motion of the end-effector when it is close to the tool grasping position. This behavior is implemented in order to avoid a collision of the manipulator with the tool, while allowing the operator to quickly extract/insert the grasping position [69].

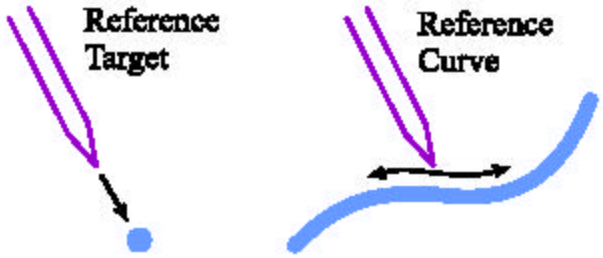


Figure 2.12 Two Types of Reference Direction Fixtures [55]

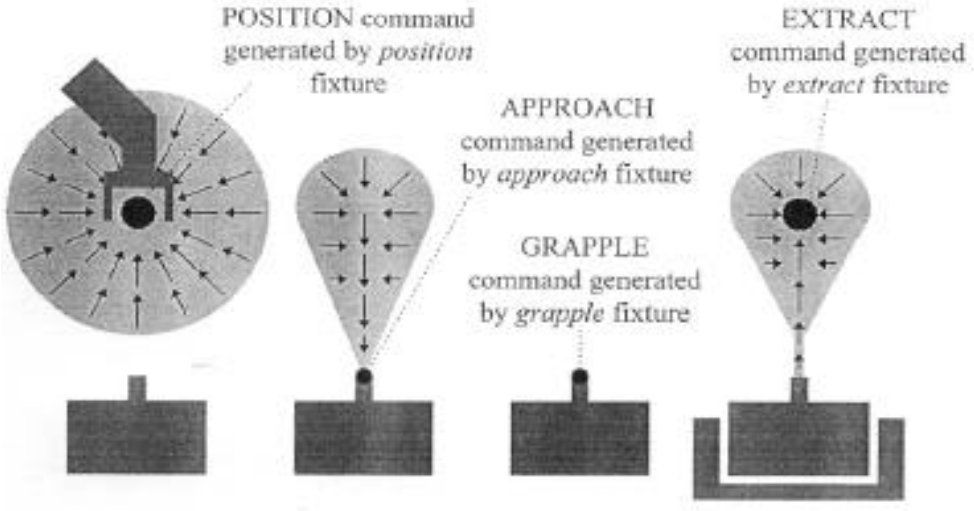


Figure 2.13 Virtual Fixtures to Aid Extract / Insert Motion [69]

Chapter 3: Teleoperation with Assistance Functions

3.1. Introduction

This chapter describes a telemanipulation system to assist persons with disabilities perform dexterous manipulation tasks. This work is expected to enhance the teleoperation performance through the use of scaled mapping from master to slave manipulation based upon sensory data. The concept is that appropriate movement for the task is kept or even enhanced, but the undesirable movements are reduced. This is done using assist functions, that scale the input velocity according to the task. This assistance approach uses assist functions and available sensory data to perform variable velocity mapping between the master and slave, referred to as the Sensor Assist Function(SAF). A common vocational rehabilitation test referred to as Box and Blocks was chosen to test the effectiveness of this sensor-assisted function. A variable scaling scheme was developed using available sensory data. In the simulation mode, a visual environment was created for the Box and Blocks test. This was used to predict if a person with disabilities would be able to perform a task comfortably. The real test was performed using a master and slave manipulator system with a camera and laser range finder. A motion constraint was added to the master to simulate a user with disabilities. The results demonstrated that the sensor assistance not only reduced required input motion, idle time, and execution time, but also increased manipulation accuracy during the Box and Blocks test. This work

prompted the need of building a test-bed that uses available sensory information to adjust parameters during task execution.

3.2. Assistance Functions Concept

Assistance functions were developed to assist the operator by scaling the input velocity according to the task. The assistance includes linear assistance, planar assistance, and velocity assistance.

The linear assist function constrains the input velocity along a line. The input velocity is transformed to a task frame and multiplied by a scaling matrix, and then transformed back to the base frame. A goal line is determined between two points and defined as the *X*-axis of the linear task frame. The *Z*-axis is defined as the perpendicular vector, and the *Y*-axis is defined by the cross product of *Z* cross *X*. A transformation matrix is calculated according to the task frame, and is multiplied by the input velocity.

$$\begin{bmatrix} V_{slaveX} \\ V_{slaveY} \\ V_{slaveZ} \end{bmatrix} = \begin{bmatrix} a_{11} & b_{11} & c_{11} \\ a_{21} & b_{21}. & c_{21} \\ a_{31} & b_{31} & c_{31} \end{bmatrix} \cdot \begin{bmatrix} V_{masterX} \\ V_{masterY} \\ V_{masterZ} \end{bmatrix} \tag{3.1}$$

where V_{slave} is the input velocity in the task frame. Then a scaling matrix is applied to scale down the velocity in the undesired directions along the task frame *Y* and *Z*-axis.

$$\begin{bmatrix} V_{scaledX} \\ V_{scaledY} \\ V_{scaledZ} \end{bmatrix} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} \cdot \begin{bmatrix} V_{slaveX} \\ V_{slaveY} \\ V_{slaveZ} \end{bmatrix} \tag{3.2}$$

where the values of k_x, k_y, k_z depend on a specific task. In the linear assistance case, the values of k_y and k_z are very small. Then, V_{scaled} is transformed back to the base frame

using the transformation matrix, and that becomes the modified velocity that is sent to the robot controller.

The planar assist function constrains the input velocity along a plane. To construct this task frame, three points are used to define a plane. The *X*-axis is defined as the line between points 1 and 2. The *Z*-axis is defined as the normal to the plane, and the *Y*-axis is defined as the cross product of *Z* and *X*. A transformation matrix is determined, and the input velocity is converted to the task frame according to equation (3.1), the same as the linear case. For the planar assistance, however, the value of the scale matrix is different. Since the desired motion lies in the *X*-*Y* plane, only motion along the *Z*-axis will be scaled, so k_z is very small. After the task frame velocity, V_{slave}, is multiplied by the scale matrix, it is converted back to the base frame and sent to the robot controller, according to equation (3.2).

The velocity assist function increases and decreases the velocity according to the distance to the goal object or an obstacle. As the distance to the goal is known, a velocity scale factor can be applied to the velocity in order to increase or decrease the input velocity.

These assistance strategies are integrated together to provide a form of assistance for users with disabilities to perform the Box and Blocks task in this research.

3.3. Box and Blocks Task

The Box and Blocks test measures gross manual dexterity and is frequently used in research on rehabilitation. This test, represented in figure 3.1(simulation mode) and figure 3.2 (real testing), consists of moving one-inch blocks from one side to another in a two-sided box. A wall divides the two sides. This test the use of large motions in all

directions. The goal is to pick up the block from one side, and place it in the other side. In simulation mode (Figure 3.1), force feedback was added to make user feel resistive force and collision. In real test (Figure 3.2), a sphere constraint was applied to simulate the workspace of persons with disabilities. Since the possible input motion has been decreased, the able-bodied user will better represent a person with disabilities. Assistance function algorithm is based on sensory data.

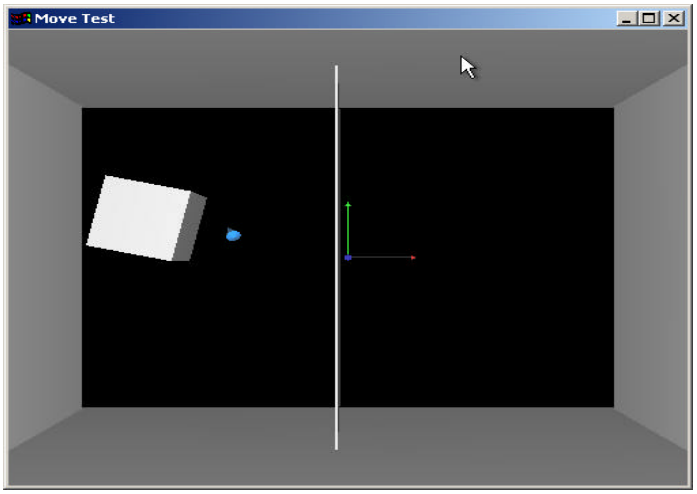


Figure 3.1 Box and Blocks Test Window Interface

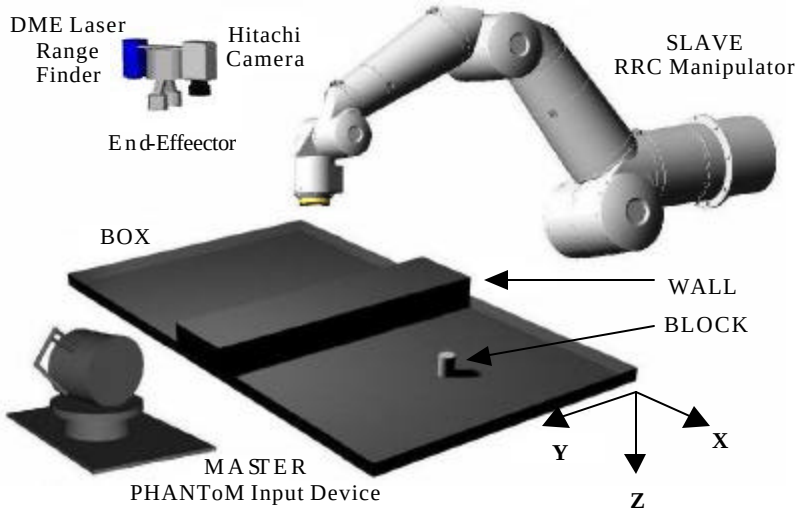


Figure 3.2 Box and Blocks Test, Master and Slave

3.4. Sensor Assist Function

In this research, a combination of the linear, planar and velocity assistance, referred to as the Sensor Assist Function (SAF), was developed for the Vocational Rehabilitation test called Box and Blocks. The SAF essentially uses sensory data to perform variable velocity mapping from master to slave (Figure 3.3).

Figure 3.3 Teleoperation Test-bed

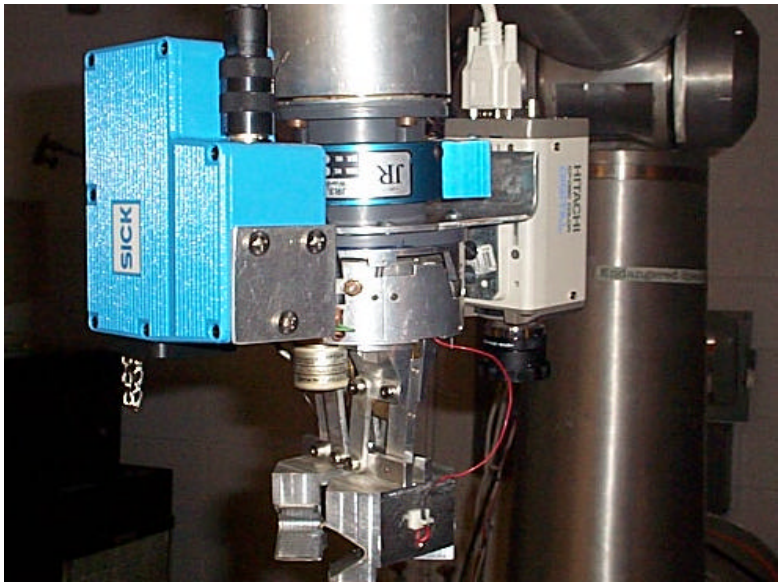


Figure 3.4 Sensors Mounted on End-Effector

The sensors include a DME 2000 Laser Range Finder (LRF), and a vision system using a Hitachi KP-D50. These sensors are mounted on the end-effector according to figure 3.4. The vision system is used to locate the goal object and obstacles. The image processing software, Halcon [77], obtains the center position of the goal object in the image plane. Once the end-effector grasps the object, the software obtains the edge of the wall, which is used to avoid obstacles. The LRF is used in the velocity assistance in the *Z*-direction depending on the depth of the obstacles and the object.

3.4.1. Description

There are seven stages of assistance shown in figure 3.5. At the start of the task, the robot is in the home position and there is no scaling until the object is seen by the vision system.

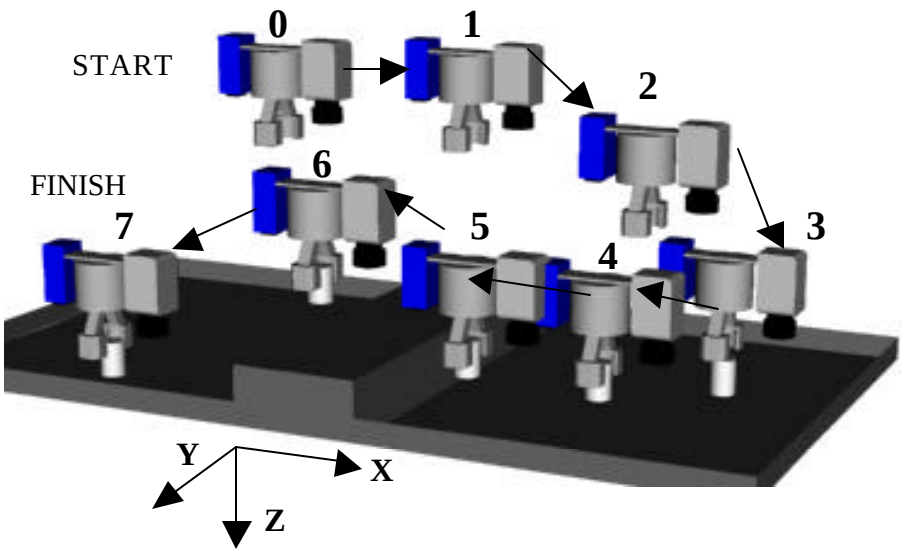


Figure 3.5 The Seven Stages of the Scaling Scheme

The first stage involves minimizing the distance between the end-effector and the object in the X-Y plane. The second stage adds z-direction scaling as the manipulator moves down. The third stage assists the manipulator when the vision system can no

longer see the goal object. Once the object is obtained, the fourth stage assists the operator in avoiding the wall obstacle. The fifth stage is activated when the range data is too close to an object. The sixth stage involves the vision system, and enhances the movement in the horizontal plane to clear the wall horizontally. The seventh stage simply frees the user to place the object down on the correct side of the box.

Since the center of the camera is not the end-effector position, the camera needs to be calibrated with the end-effector. According to figure 3.6, the end-effector position is projected on the image frame, and its pixel position is determined relative to the center position of the goal object.

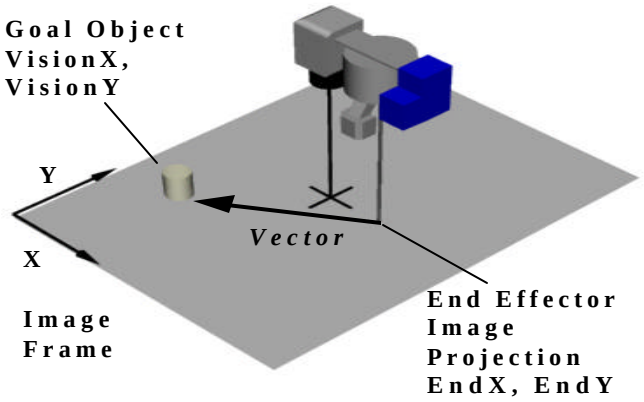


Figure 3.6 Image Frame Showing Vector Determination

3.4.2. Stage One

For stage one, the scaling is based upon the position of the object and the projected end-effector position. A vector is created between these two points, in the *X*-*Y* plane, and the task frame is calculated using this vector and a Z-axis. The x-direction of the image frame is opposite to the x-direction of the slave frame, so the vector calculation is as follows:

$$Vector = (\text{EndX} - \text{VisionX}) \cdot x, (\text{VisionY} - \text{EndY}) \cdot y \tag{3.3}$$

A transformation matrix is determined from the PhanToM frame to the task frame according to the task frame calculations in section 3.2, and the input velocity is scaled according to the following equations:

$$V_{SLAVE} = V_{INPUT} \cdot Transform_{O}^{C} \tag{3.4}$$

$$Scale = \begin{bmatrix} VisionScale & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \tag{3.5}$$

$$V_{SCALED} = Scale \cdot V_{SLAVE} \tag{3.6}$$

$$V_{MODIFIED} = V_{SCALED} \cdot \left(Transform_{O}^{C}\right)^{T} \tag{3.7}$$

where, for stage one, *VisionScale* ranges from 1.5 to 3 maximum. If the dot product of V_{SLAVE} and *Vector* is negative, then *VisionScale* is 0.1. This means that the input velocity is in the opposite direction of the goal object. The modified velocity, $V_{MODIFIED}$ is sent to the low-level controller.

3.4.3. Stage Two

Stage two starts when the magnitude of the *Vector* is less than 75 pixels. This means that the end-effector is close to the correct x, y position over the goal object, and the operator can start moving down towards the object. Stage one exists to help reduce the sensor error by keeping the end-effector in the *X*-*Y* plane for large movements while the operator is approaching the goal. Stage 2 uses the same task frame as stage 1, but the scale matrix reflects increased velocity in the z-direction.

$$Scale = \begin{bmatrix} VisionScale & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & ScaleFactor \end{bmatrix} \quad (3.8)$$

where *VisionScale* ranges between 1 and 1.5, and if the dot product of V_{SLAVE} and *Vector* is negative, then *VisionScale* is 0.1. *ScaleFactor* depends on the value of the LRF, shown in figure 3.7.

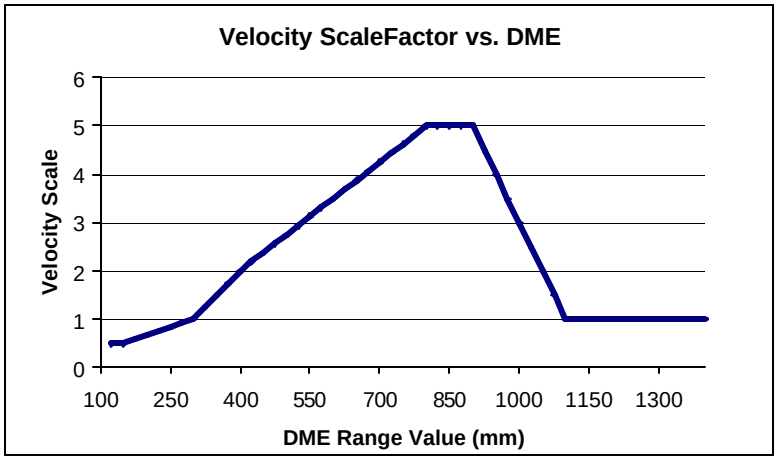


Figure 3.7 ScaleFactor According to LRF Data (DME)

So this scale matrix helps to guide the end-effector down towards the goal object. It increases the scale in the Z-direction, and allows motion in the hVector direction to pull the end-effector to the goal object.

3.4.4. Stage Three

The third stage starts when the vision system can no longer see the object. As the end-effector gets closer to the object, it will eventually move out of the image frame because of the location of the camera on the end-effector. In this stage the task frame will not be calculated since there is no data from the vision system. So the following scale matrix will be directly applied to the input velocity. Since the end-effector is near the object, there will be little motion required in the X and Y direction.

$$Scale = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & ScaleFactor \end{bmatrix} \tag{3.9}$$

$$V_{MODIFIED} = Scale \cdot V_{INPUT} \tag{3.10}$$

Using a scale of $k = 0.25$ in the X and Y direction allows for some error correction, but it scales down large movements from the operator away from the goal object.

3.4.5. Stage Four

The fourth stage begins when the end-effector grabs the object. This stage scales the velocity in order to avoid the center wall obstacle. At first, the velocity is scaled to move the end-effector in the positive z-direction according to *AvoidScale*. *AvoidScale* depends on the LRF value, and ranges from 3 to 1. If the input velocity is in the downward z-direction, then *AvoidScale* is 0.1. The y-direction is scaled down because the desired motion for the task is in the x-direction. The vision system performs edge detection and returns the greatest x-value of that edge in the image frame. The initial scaling equation is:

$$V_{MODIFIED} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & AvoidScale \end{bmatrix} \cdot \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \tag{3.11}$$

3.4.6. Stage Five

A as the end-effector moves to the left to place the object on the other side of the box, the LRF is monitored for obstacles. If the LRF sees an obstacle, then all velocity inputs are scaled down, and the upward z-direction is increased by *AvoidScale*, according to the following equation:

$$V_{MODIFIED} = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & AvoidScale \end{bmatrix} \cdot \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \quad (3.12)$$

As the end-effector moves to the left, the LRF leads, according to figures 3.1, 3.2 and 3.3. Figure 3.3 shows how the LRF can measure the wall without a collision. Therefore, the LRF checks the z-direction to make sure the whole end-effector can clear an obstacle.

3.4.7. Stage Six

Now that the end-effector has enough height to clear the wall vertically, it must clear the wall horizontally. So, once the wall comes into the image frame, the scaling is shown by the following equation:

$$V_{MODIFIED} = \begin{bmatrix} Avoidwall & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \cdot \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \quad (3.13)$$

where Avoidwall increases the negative x-direction, see figures 3.1 and 3.3, to assist in avoiding the seen obstacle. Once the wall obstacle is seen, the z-direction will be scaled down.

3.4.8. Stage Seven

Once the camera can no longer see the wall, the end-effector has avoided the wall obstacle. The scaling returns to regular z-direction velocity assistance according to the following equation.

$$V_{MODIFIED} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & ScaleFactor \end{bmatrix} \cdot \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \quad (3.14)$$

Once the object is near the table on the correct side of the box, the operator is ready to release the object. Now the task is completed, and the completion time is recorded. By returning the end-effector to home position, the operator is now ready to perform another Box and Blocks test.

3.5. Experimental Results

3. 5.1. Telemanipulation System Structure

In this system (figure 3.8), the master robot is a PhanToM with 6 degrees-of-freedom from Sensable Technologies. It can provide tactile feedback for the user. A 7 DOF industrial robot RRC K-2107a is used as a slave manipulator in this application. A Windows 2000 PC is used to control the PhanToM and compute the mapping from master to slave. The slave manipulator controller runs another PC. A third PC handles the sensory data. All PCs are linked together through an Ethernet, and sensory data is sent to the PhanToM PC and the velocity commands are sent to the manipulator PC.

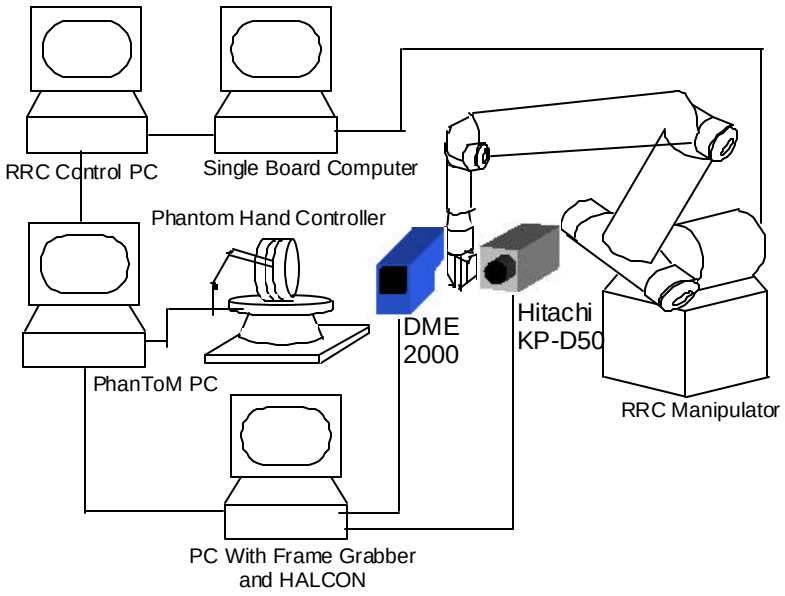


Figure 3.8 The Telemanipulation System

3.5.2. Software Implementation

Two major programs have been developed in this chapter. One is the image processing, which does the Sobel edge detection and region growing to obtain the coordinates of the object in image plane (figure 3.9 and 3.10). This program uses API functions provided by HALCON.

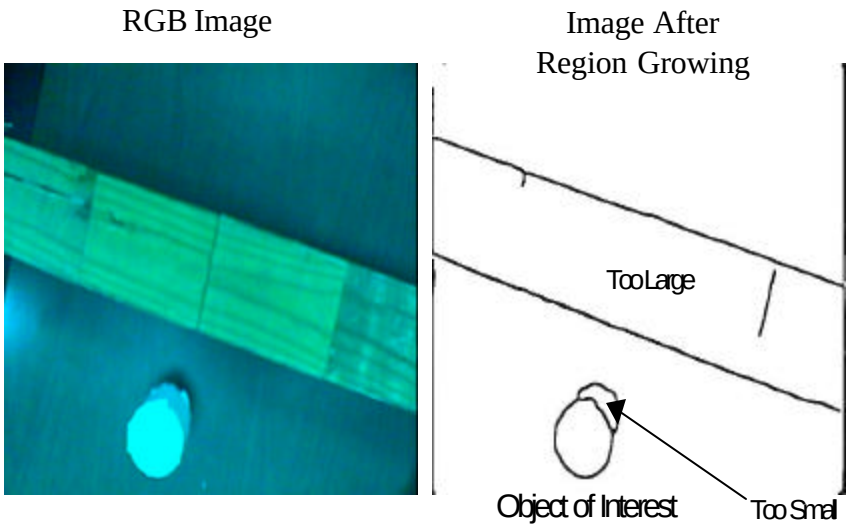


Figure 3.9 Region Growing Image

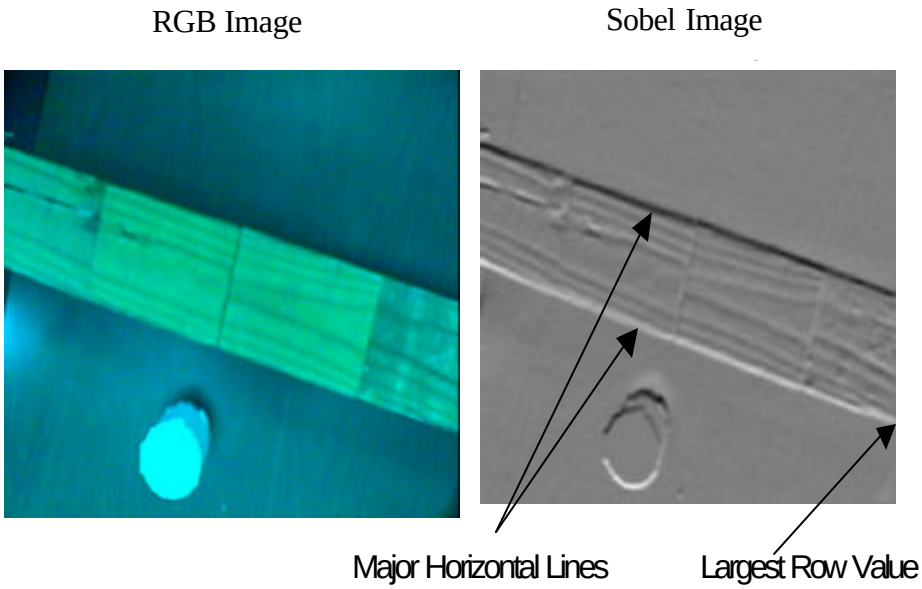


Figure 3.10 Sobel Edge Detection Image

The other is the control program run in the master PC. It was developed using the GHOST SDK from Sensable Technology [48]. This software obtains the accurate position and orientation of the PhanToM. Force reflection is also available with the software. The sample time of the master PC getting position or velocity data from the master device is 0.2s. Once the master velocity is obtained, it is modified according to the SAF. This adjusted velocity command is sent to the slave PC at the same rate as its sample rate.

3.5.3. Results

3.5.3.1. Simulation Mode

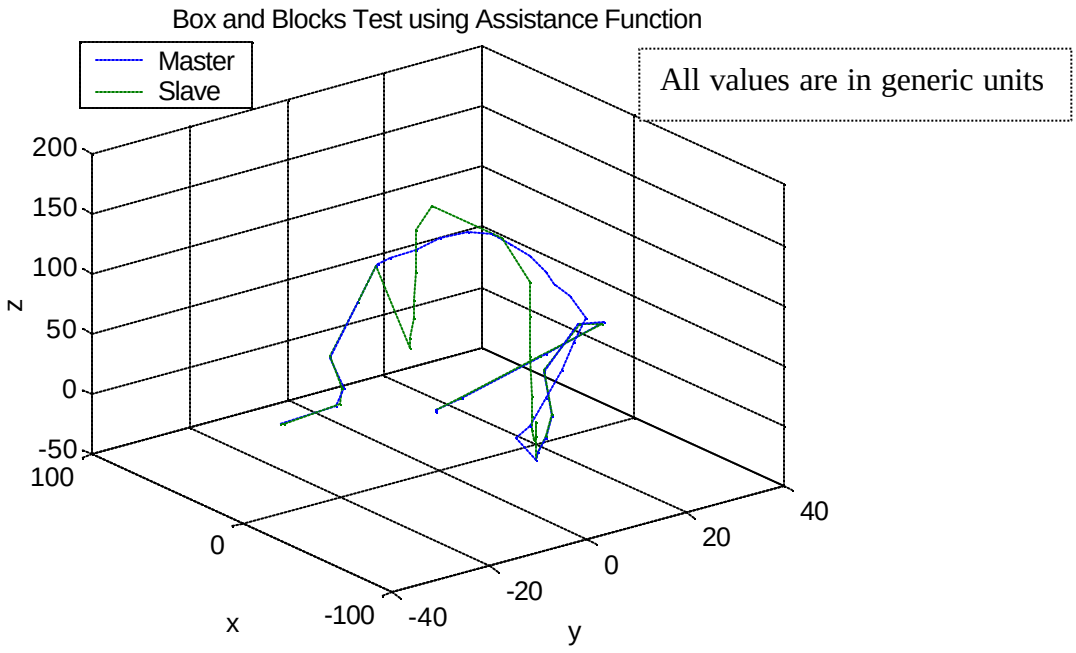


Figure 3.11 Trajectory Comparison of PhanTom and Slave Manipulator

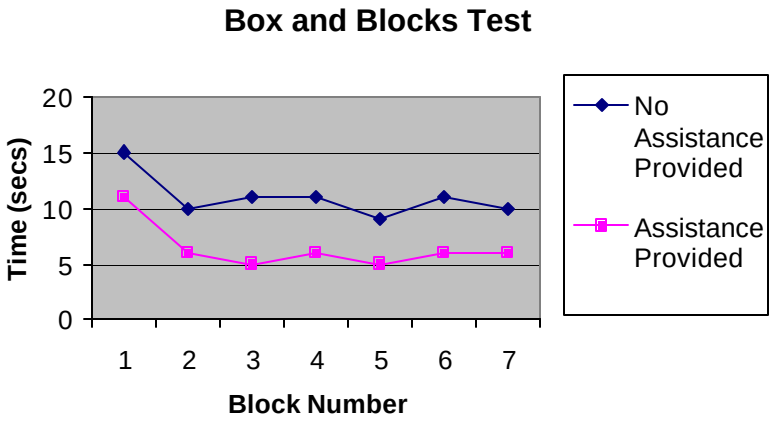


Figure 3.12 Box and Block Time Execution

An able-bodied person performed the Box and Blocks simulation with assistance function to determine the effect of the assistance. When user's movement is away from the desired trajectory, force reflection will be felt by the user that makes the user move back to the desired trajectory. Figure 3.11 is the trajectory comparison of the Phantom and slave manipulator when doing box and blocks test with assistance function. Obviously, though the master has some random movements, the slave manipulator moves along a desired trajectory very well. A sample of time executions of seven tests is shown in figure 3.12. It is noticed that due to the assistance function, the average time was reduced considerably (from 10.33 to 5.66 seconds), and the standard deviation (from 0.81 to 0.50) was smaller as well.

3.5.3.2. Real Test Mode

An able-bodied person performed the Box and Blocks real test with and without the SAF to determine the effect of the assistance with a sphere constraint in his workspace, which simulated the motion of persons with disabilities. The height of the wall in the tests is 10 inch. Figure 3.13 shows the trajectory of the slave manipulator with

no assistance versus the slave with assistance when doing real box-block test. According to this figure, the trajectory with assistance is a smooth curve approaching the object, and then avoiding the wall obstacle. The curve shows how the user was guided toward the object. The trajectory with no assistance shows that the user has a random approach to the object, while showing many uncertain and unnecessary movements. It also shows the effect of each stage of scaling.

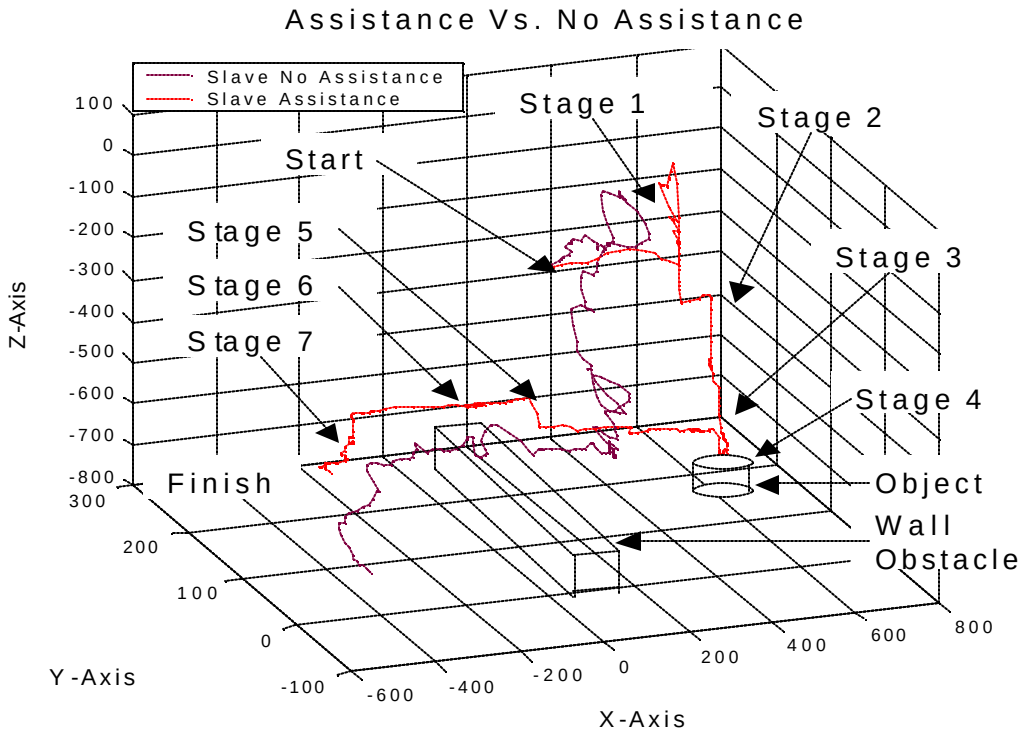


Figure 3.13 Trajectory of Box and Blocks Task

For data analysis, the person performed the test 30 times with assistance and 30 times without assistance. Table 3.1 shows the results of the tests. It includes the decrease of necessary input motion, idle time, and execution time when using the developed computer assistance. Whenever in simulation or real test mode, assistance functions not

only decreased the execution time, but also reduced its standard deviation from 4.512s to 2.086s.

Table 3.1 Comparison of Averages for Box and Blocks Test Using Workspace Constraint

Average Test Data-All Positions	No Assistance	SAF Assistance	% Decrease
Total Distance	11.87	9.89	16.7%
Number of Times Reposition	43.80	23.80	45.7%
Time Spent Repositioning	22.56	9.66	57.2%
Total Completion Time	76.63	50.24	34.4%

3.6. Summary

This work provides a virtual simulation and sensor-assistance approach for a complex teleoperation task to be executed by persons with disabilities. It can be used as a vocational training platform and as an evaluation tool after therapy in rehabilitation engineering. The assistance will increase the safety and dexterity of these users who would not be able to perform the task otherwise. In this dissertation, the Box and Blocks test was explained as well as a suitable combination of assistance that variably scales the input velocity. Able-bodied persons initially performed the test to show the effect of the assistance concept. A constraint was added to the input to simulate a person with disabilities by decreasing the possible movements of the able-bodied user, and more tests were performed. The results show how the desired motion was kept or sometimes augmented, and how the unwanted motion was reduced. Therefore, when applying this assistance, the performance of a person with disabilities will be drastically enhanced.

Chapter 4: Telemanipulation Assistance Based on Motion Intention Recognition

In telemanipulation systems, assistance through variable position/velocity mapping or virtual fixture can improve manipulation capability and dexterity [37, 45, 53, 61, 64]. Conventionally, such assistance is based on the sensory data of the environment and without knowing user's motion intention. In this dissertation, user's motion intention is combined with real-time environment information for applying appropriate assistance. If the current task is following a path, a virtual fixture is applied. If the task is aligning the end-effector with a target, an attractive force field is produced. Similarly, if the task is avoiding obstacles that block the path, a repulsive force field is generated. In order to successfully recognize user's motion intention, a Hidden Markov Model (HMM)-based algorithm is developed to classify human actions, such as following a path, aligning target and avoiding obstacles. The algorithm is tested on a simulation platform. This chapter presents the teleoperation assistance algorithm development based on operator's motion intention recognition through Hidden Markov Model (HMM). The basic theory and the application of HMM are also presented.

4.1. Telemanipulation Assistance

The fundamental purpose of a telerobotic system is to extend operator's sensory-motor facilities and manipulation capabilities in remote environment [70]. This approach is guided by the philosophy that the human operator should remain in direct control of the

slave at all times, with human-independent control parameters altered according to sensor information. However, manipulation tasks such as assembly are still difficult for a telerobotic system. In many cases, the user's physical labor load of completing a task manually is replaced by mental burden of controlling the remote input device mentally. In the field of rehabilitation robotics, this is the main hindering for the wide application of telerobot assistive devices [71]. So assistance for teleoperation has become essential in order to reduce the operation fatigue. The first kind of assistance is the variable position and velocity mapping based on sensory information and force feedback [53]. The other is virtual fixture, which has been used as means of providing direct, physical assistance [37, 61, 64]. Just imagine drawing a straight line without a ruler, it is very difficult. Virtual fixture plays the same role as a ruler to enhance human's drawing a straight line. Both of these assistances can enhance a human's performance accuracy for complex tasks execution and reduce time consumption. But the limitation is that they are related to some specific tasks. Our recent work in telemanipulation systems for rehabilitation engineering motivated us to enhance manipulation accuracy and reduce operator's fatigue [29, 50, 53]. In order to provide general assistance, specific tasks need to be divided into several simple and general subtasks. Our work tries to combine the environment information with user's motion intention before applying appropriate assistance. Human motion intention is classified by movement velocities through Hidden Markov Model: following a path, aligning with a target, avoiding an obstacle and stopping. For each motion, appropriate assistance is provided. For example, if the motion is following a path, a virtual fixture orthogonal to the path is applied, just like a ruler. If the motion is aligning with a target, an attractive force field is applied.

4.2. Classes of Motion in Telemanipulation

With typical telemanipulation, the user enters the control loop, sensing the environment information such as the location and the distance of the target and providing the appropriate control signal through moving the input device. For a common task, such as grasping a cup and putting it on a cup pad, the motion process can be divided into four classes:

1. Following the desired trajectory;
2. Aligning with the target;
3. Avoiding an obstacle; and
4. Stopping

The "following the desired trajectory" motion happens when a desired trajectory is planned. For the “go grasp” task, the desired trajectory is a straight line if there is no obstacle blocking the path. We can decompose the velocity vector v_c into two parts, v_p, velocity component along the desired path tangent direction and v_o, velocity component orthogonal to the desired path tangential (Figure 4.1). While users are following a path, $v_p >> v_o$ (Figure 4.1); While aligning the end effector with the target, both v_p and v_o are relatively small and close to each other (Figure 4.2); while avoiding an obstacle, $v_p << v_o$ (Figure 4.3); and when stopping, both v_p and v_o are close to zero (Figure 4.4). But these features are not true for each sample. We can not classify these four motions for each sample value using a simple threshold. So Hidden Markov Model, a technique of stochastic process is used. Since these two velocity components are orthogonal, they are independent. In order to apply HMM to model these two velocities components, a 2-dimensional HMM is used.

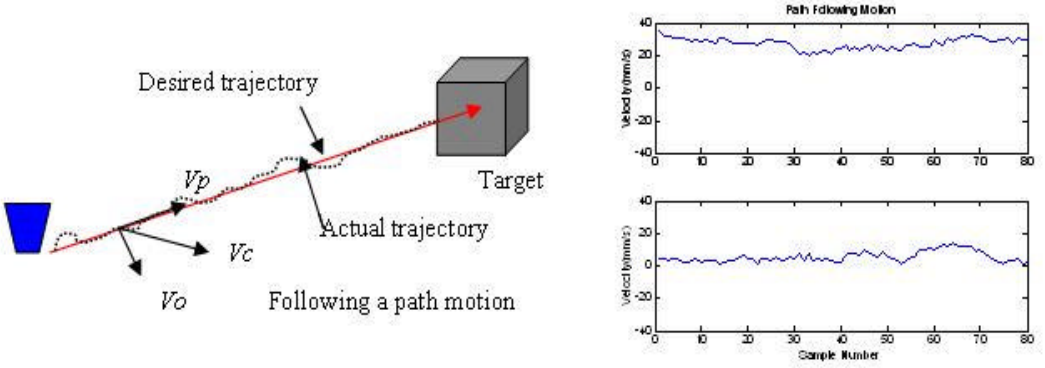


Figure 4.1 Path Following Motion and its Velocity Profile

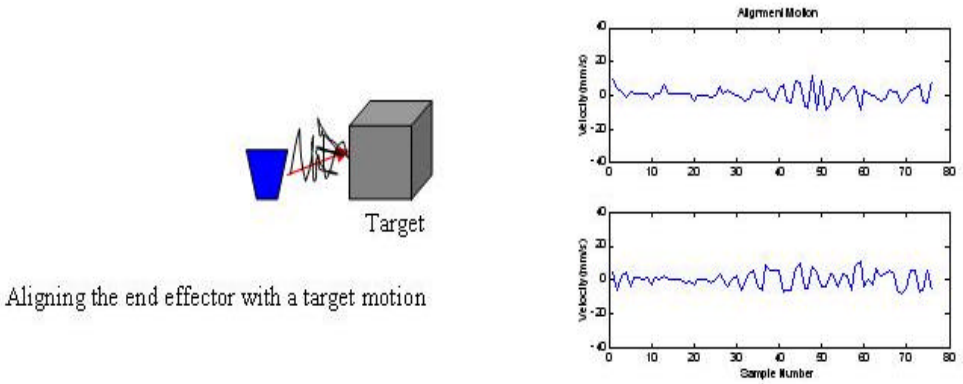


Figure 4.2 Aligning with Target Motion and its Velocity Profile

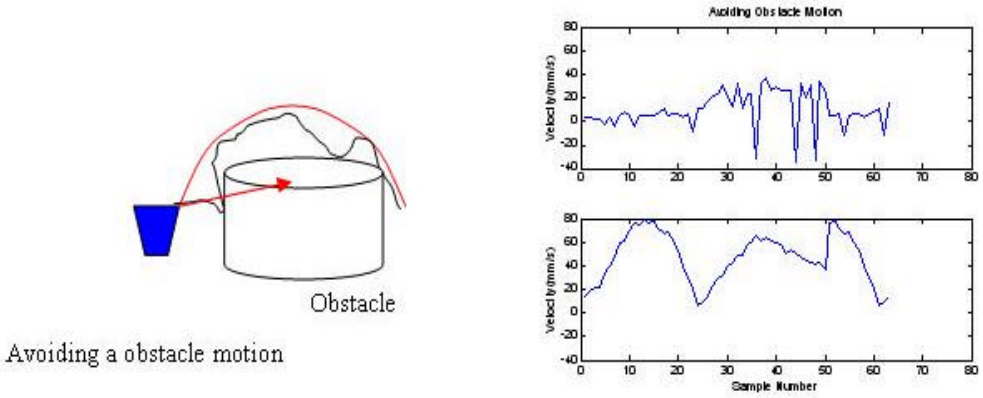


Figure 4.3 Avoiding Obstacle Motion and its Velocity Profile

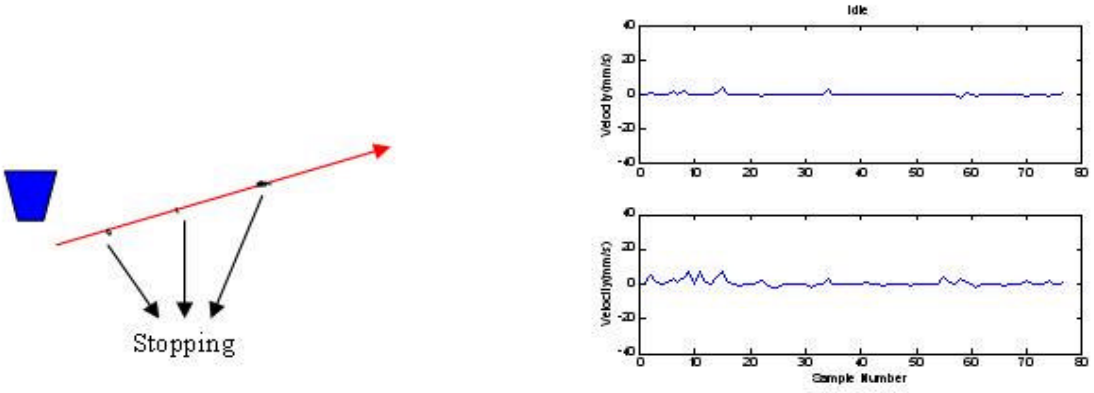


Figure 4.4 Stop Motion and its Velocity Profile

4.3. Hidden Markov Model based Motion Recognition

4.3.1. Data Preprocessing

The velocity of the input device is sampled at 1000Hz rate. The data is denoted as $V = [V_p, V_O]$, *Vp* and *Vo* are the sets of velocity sampling values v_p and v_o.

$$\begin{cases} V_p = [v_{1,p}, v_{2,p}, \ldots\ldots v_{n,p}] \\ V_O = [v_{1,o}, v_{2,o}, \ldots\ldots v_{n,o}] \end{cases} \tag{4.1}$$

where *n* is the sample number. Since V_p and V_o play the same role, we just demonstrate the data processing of one of them, i.e. *Vp*. Since we use discrete HMM, we need to convert this velocity data into a sequence of discrete symbols. We follow two steps in this conversion: (1) data preprocessing and (2) vector quantization, as illustrated in Figure 4.5. The primary purpose of data preprocessing is to extract meaningful feature vectors for the vector quantization. In our case, the preprocessing proceeds in two steps: (1) spectral conversion, and (2) power spectral density (PSD) estimation.

First, a 16-point width window with 50% overlap is used to select data:

$$\overrightarrow{v_p} = [v_{1,p}, v_{2,p}, \ldots\ldots v_{16,p}] \tag{4.2}$$

Prior to spectral conversion, a hamming window is used to filter each frame, thus minimizing spectral leakage. The Hamming transformation $T_H^v(\cdot)$ maps a *k*-length (*k* =16 in this case) real vector to a new k-length real vector.

$$\vec{h} = T_H^v(\overrightarrow{v_p}) = [H_1 v_{1,p}\ H_2 v_{2,p} \ldots H_k v_{k,p}] \quad (\mathrm{k} = 16) \tag{4.3}$$

$$H_i = 0.54 - 0.46\cos[\frac{2p(i-1)}{k-1}], \quad \mathrm{i} \in \{1,2,\ldots,\mathrm{k}\} \tag{4.4}$$

Next, FFT (Fast Fourier Transform) analysis is applied for every Hamming windowed data. The FFT transform $T_F^h(\cdot)$ maps a *k*-length vector $\vec{h} = [h_1, h_2, \ldots h_k]$ to a *k*-length complex vector $\vec{z} = [z_1, z_2, \ldots z_k]$.

$$\vec{z} = T_H^h(\vec{h}) = \left[F_0(\vec{h})\ F_1(\vec{h}) \ldots F_{k-1}(\vec{h})\right] \quad \text{where} \tag{4.5}$$

$$F_p(\vec{h}) = \sum_{q=0}^{k-1} h_{q+1} e^{2p\ \mathrm{ipq/k}}, \quad p \in \{0,1,\ldots k-1\}$$

Now, let us define the power spectral density (PSD) estimates for the hamming-Fourier output $\vec{z}$. The PSD estimates is given by,

$$P(\vec{z}) = [P_0(\vec{z})\ P_1(\vec{z}) \ldots P_{k/2}(\overrightarrow{z})], \text{ where} \tag{4.6}$$

$$P_0(\vec{z}) = \frac{1}{H_{ss}}|z_0|^2,$$

$$P_i(\vec{z}) = \frac{1}{H_{ss}}(|z_i|^2 + |z_{k-i}|^2), \quad i \in \{1, 2, \ldots, k/2-1\},$$

$$P_{k/2}(\vec{z}) = \frac{1}{H_{ss}}|f_{k/2}|^2, \text{ and } H_{ss} = k\sum_{q=1}^{k} H_k^2$$

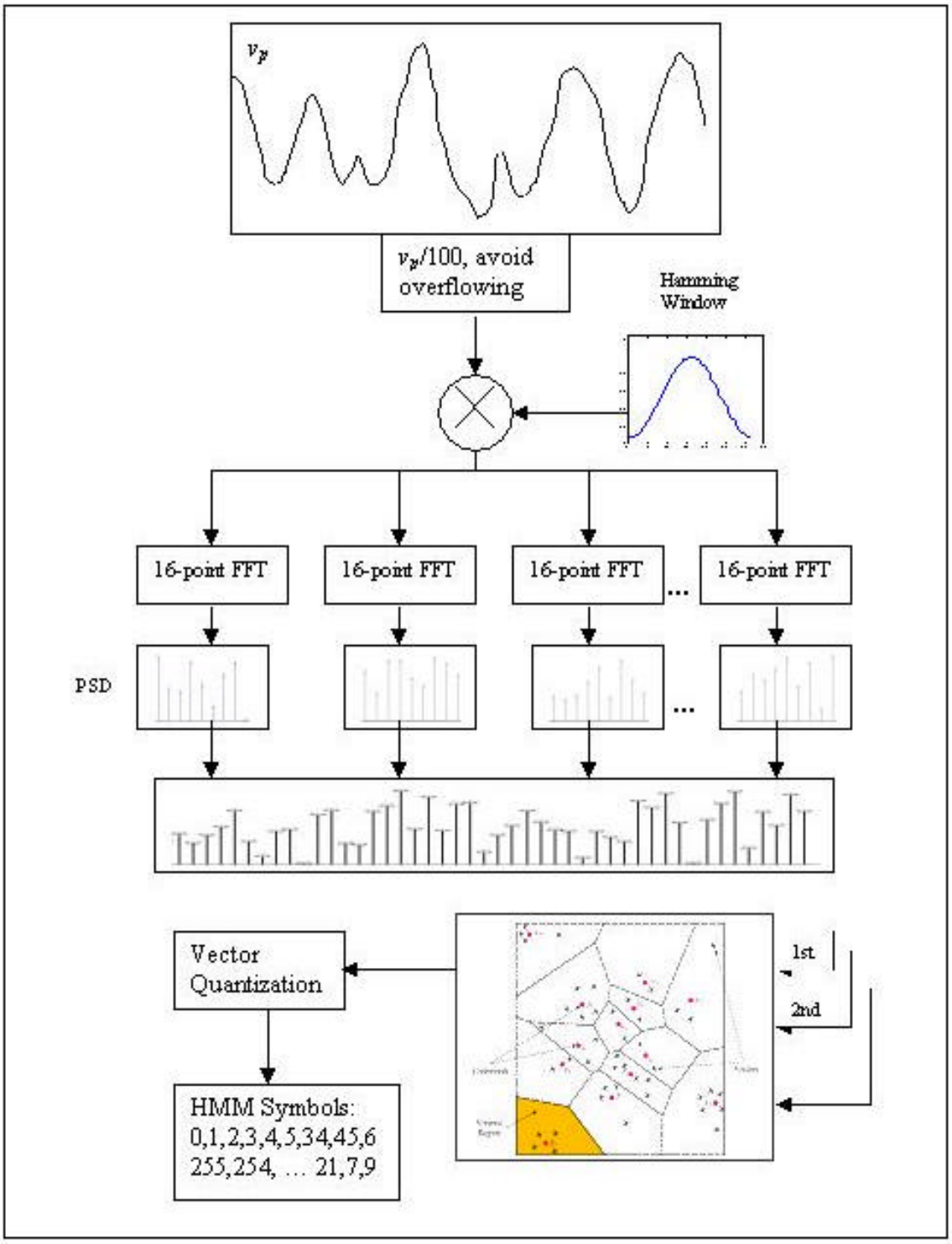


Figure 4.5 Conversion of Continuous Velocity Data into Discrete Symbols

Due to the symmetry structure, the length of PSD estimates output is $k/2 = 8$. As illustrated above, a 16-point velocity samplings window is mapped to an 8-point PSD vector. Let us represent the hamming windowing, Fourier transform and power spectral density by $T^{v}_{(H,F,P)}(\cdot)$. If there are m sampling windows, the PSD estimation vectors form a matrix as shown below,

$$V_P^m = \begin{bmatrix} \{\dot{P}(\dot{z})\}_1 \\ \{\vec{P}(\vec{z})\}_2 \\ \vdots \\ \{\vec{P}(\vec{z})\}_m \end{bmatrix} = \begin{bmatrix} T^{v}_{(H,F,P)}(\overrightarrow{v_{p,1}}) \\ T^{v}_{(H,F,P)}(\overrightarrow{v_{p,2}}) \\ \vdots \\ T^{v}_{(H,F,P)}(\dot{v}_{p,m}) \end{bmatrix} \tag{4.7}$$

In the same way, the second dimensional data, V_o can be converted into a PSD matrix as V_O^m above.

4.3.2. Vector Quantization

In the previous section, we converted raw velocity data into the feature matrix V_P^m and V_O^m. Let $V = \{\dot{v}_t\},\ t \in \{1, 2, \ldots m\}$ denote the set of all feature vectors. In order to apply discrete-output HMMs, we now need to convert the feature vectors V to N discrete symbols, where N is the number of output observables in our HMMs. In other words, we want to replace the many $\overrightarrow{v_t}$ with L prototype vectors $Q_N = \{\vec{x}_n\}, n \in \{1,2,\ldots N\}$, known as the codebook, such that we minimize the total distortion $D(V, Q_N)$

$$D(V, Q_N) = \sum_t \min\, d(\vec{v}_t, \vec{x}_n), \text{ where } d(\vec{v}_t, \vec{x}_n) = (\vec{x}_n - \vec{v}_t) \cdot (\vec{x}_n - \vec{v}_t)^T \tag{4.8}$$

over all feature vectors. We choose the well-known LBG vector quantization (VQ) algorithm [72] to perform this quantization. The illustration of LBG algorithm for

different N is shown in Figure 4.7. For our case, N is determined to be 256. For our data, we set the split offset e = 0.001 and the convergence criterion d_{VQ} = 10.0e^{-15}. With these parameter settings, the centroids usually converge within only a few iterations. Thus, the velocity signal is trained and classified into 256 vectors, denoted by VQ codebook Q_N. Now, given a sequence of feature (velocity for our case) vector V^f, we can convert them into a symbol vector $S_f = \{s_1, s_2, ..., s_f\}$ with length f. Let us use $T_{VQ}(\cdot)$ to represent the conversion from feature vector into symbol, then

$$S_f = T_{VQ}(V^f, Q_N) = \{T_{VQ}(v^1, Q_N), T_{VQ}(v^2, Q_N), ..., T_{VQ}(v^f, Q_N)\} \tag{4.9}$$

$$s_i = T_{VQ}(v^f, Q_N) = index[\min d(v^f, x_n)], n \in \{1, 2, ... N\} \tag{4.10}$$

We train the VQ codebook by these vectors and the codebook is produced by LBG algorithm (see Figure 4.6). The LBG VQ (vector quantization) technique maps these 8-dimensinal vectors into a finite set of vectors $Y = \{y_i: i = 1, 2, ..., L\}$, where L is the length of the codebook(it is determined to be 256 in our case). Each vector y_i *is* called a code vector or a codeword and the set of all the codewords is referred to as a *codebook*. Associated with each codeword, y_i, is the nearest neighbor region called *Voronoi* region, and it is defined by [72]:

$$V_i = \{x \in R^k : \|x - y_i\| \le \|x - y_j\|, \text{ for all } j \ne i\} \tag{4.11}$$

The 256 8-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. The computation procedures of the data preprocessing part are illustrated in Figure 4.5. This method is similar to the continuous-symbol conversion in [62]

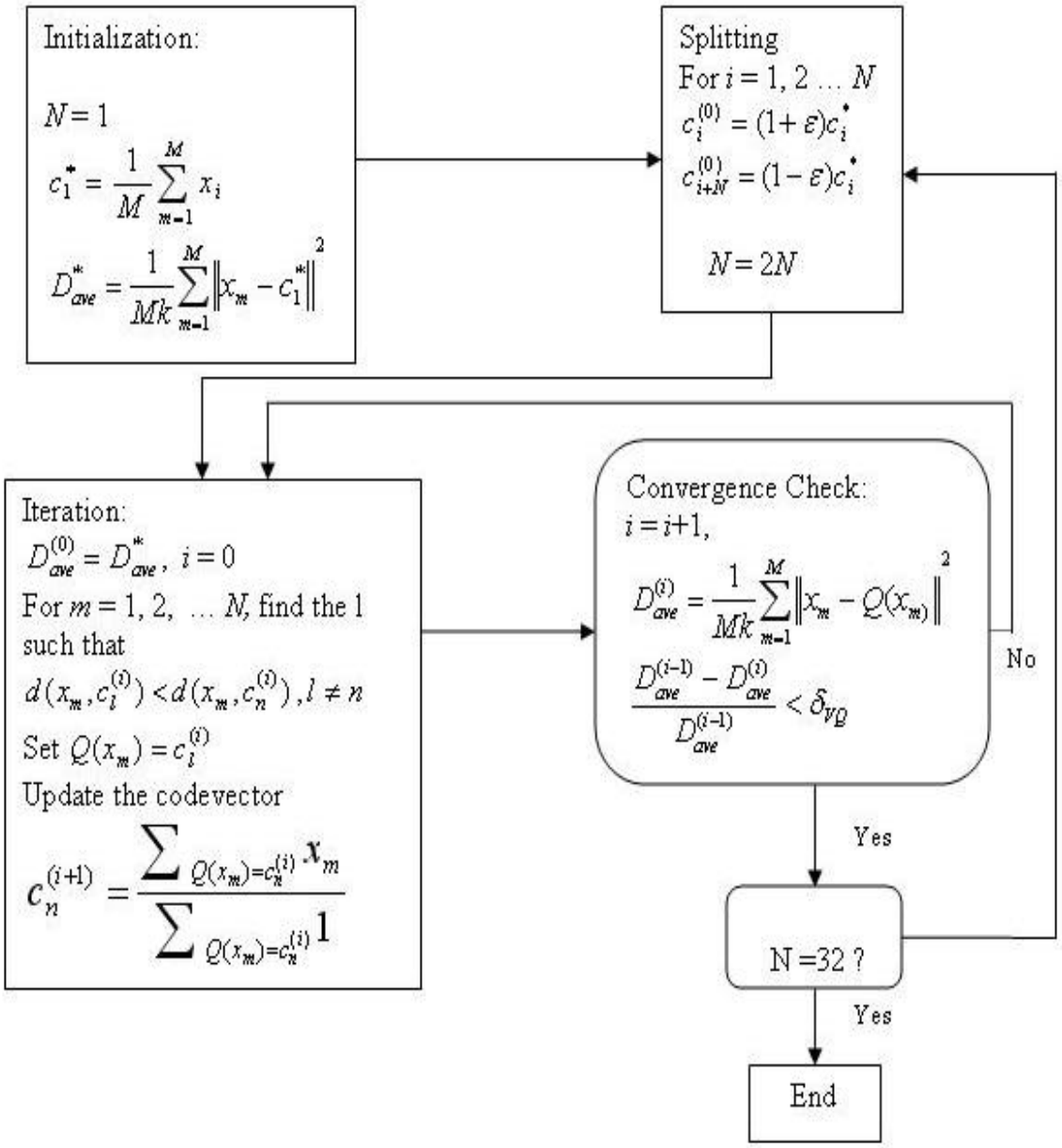


Figure 4.6 LBG Codebook Training

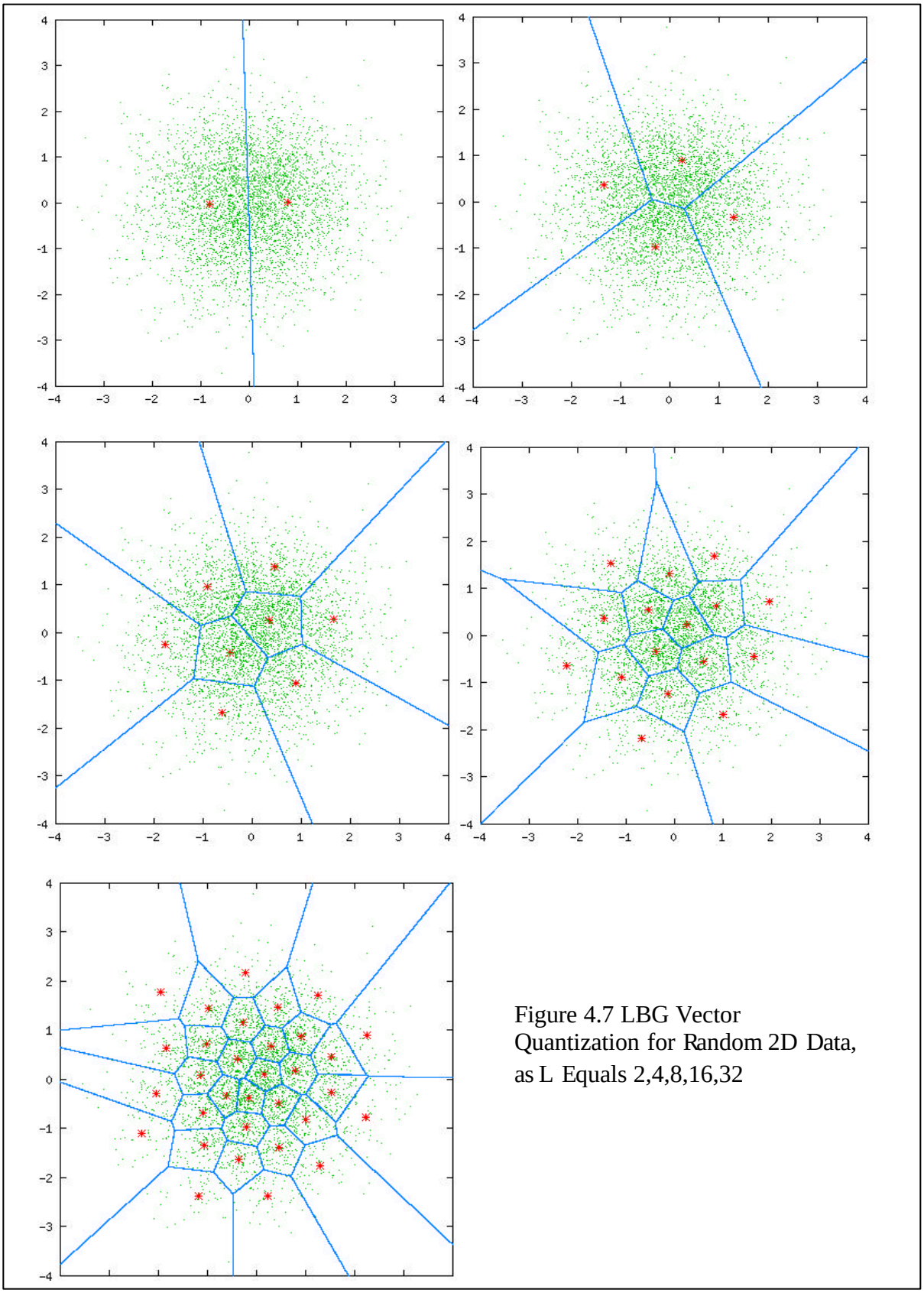


Figure 4.7 LBG Vector Quantization for Random 2D Data, as L Equals 2,4,8,16,32

4.3.3. HMM Training

HMM is usually used in continuous and discrete forms. Relatively, discrete HMM is easier for computation. In this dissertation, discrete HMM is adopted. A discrete HMM can be defined as follows [63]:

1. A set of N states $S=\{S_1, S_2 \ldots S_N\}$
2. A set of M possible observations $V=\{v_1, v_2 \ldots v_M\}$
3. A state transition probability distribution $A=\{a_{ij}\}$, where $a_{ij}=P\,[q_{t+1}=S_j | q_t = S_i]$, $1<=i, j<=N$
4. Observation probability distribution in each state j, $B=\{b_j(k)\}$ where $b_j(k)=P\,[v_k \text{ at } t | q_t = S_j]$, $1<=j<=N$, $1<=k<=M$
5. Initial State distribution $\pi = \{p_i\}$, where $p_i = P\,[q_i=S_i]$ $1<= i <=N$
6. Let $\lambda = (A, B, \pi)$ be the complete parameter set.

Figure 4.8 represents a 5 state HMM, where each state emits one of 256 discrete symbols in two dimensions.

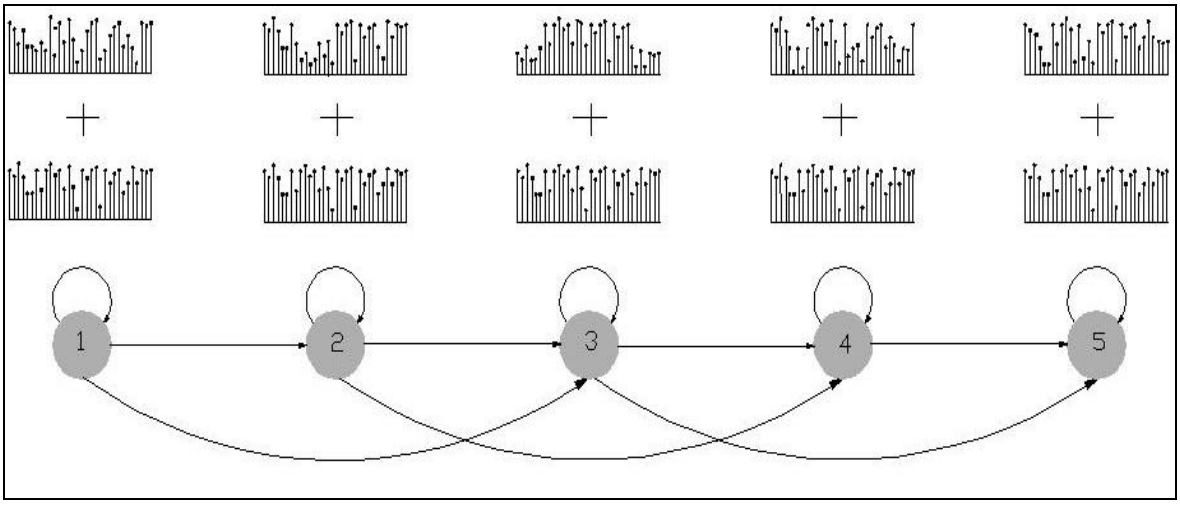


Figure 4.8 5-states Left-Right Hidden Markov Model, with 32 Observable Symbols in Each State

In order to train an HMM model and use it to do recognition, the following three basic problems for HMM need to be solved [63]:

1. Given the observation sequence $O = o_1\ o_2 \ldots o_T$, and a model $\lambda = (A, B, \pi)$, how to determine $P(O|\lambda)$, the probability of the observation sequence, given the model? This can be viewed as scoring a model in terms of how well it matches the observation.
2. Given the observation sequence $O = o_1\ o_2 \ldots o_T$, and a model $\lambda = (A, B, \pi)$, what is the best corresponding state sequence $Q = q_1\ q_2 \ldots q_T$, that best explains the observation (e.g. the most probable sequence).
3. How do we set or adjust the parameters of a model $\lambda = (A, B, p)$ to maximize $P(O|\lambda)$. This is the training or learning problem of adjusting the model's parameters to best fit a set of training data.

In order to classify four different motions, we need to design a separate HMM for each motion. The observations are a sequence of coded spectral vectors where each spectral vector is mapped to one of several code words which is the closet match. Also the observations are sequences of codes representing the motion executed repeatedly by one or more operators. The solution to problem 3 is to set the parameters of the model for each motion. The solution to problem 2 is to segment each of the motion training sequences into states and thereby gain information about how to adjust the number of states or the codebook. Once the four models are built; we can use the solution to problem 1 to score each motion model's match to a given observation sequence and select the best model. The computation of the three problems will be explained in this section.

Problem 1 is to determine $P(O|\lambda)$. Examine every state sequence length T, $Q = q_1,q_2,\ldots,q_T$, how likely this state sequence is and how likely it is to generate the observation sequence. First, we assume that individual observations are independent, and then the probability of observing O given Q is [63]:

$$P(O \mid Q, l) = \prod_{t=1}^{T} P(O_t \mid q_t, l) = b_{q1}(o_1) \cdot b_{q2}(o_2) \cdots b_{q_T}(o_T) \tag{4.12}$$

The probability of a given state sequence is simply:

$$P(O, Q \mid l) = P(O \mid Q, l) P(Q \mid l) \tag{4.13}$$

So the joint probability of an observation and a state sequence is:

$$P(O \mid l) = \sum_{all\ Q} P(O \mid Q, l) P(Q \mid l) \tag{4.14}$$

The computation of Eq.(4.14) requires summing over N^T possible sequences. Instead, a forward-backward procedure is used to do this. The detailed algorithms is described by L. Rabiner[63].

Problem 2 is to find the state sequence, Q, which is the most probable given a sequence of observations, i.e want to maximize $P(Q|O,l)$, or equivalently maximize $P(Q,O|\lambda)$. The Viterbi algorithm [63] finds this state sequence by defining

$$d_t(i) = \max_{q1,q2,..q_t} (P[q_1, q_2, ...q_t = i, O \mid l]) \tag{4.15}$$

i.e. the probability of the best subsequence that accounts for the first t observations and ends in state S_i. The induction

$$d_{t+1}(j) = \max_i (d_t(i) a_{ij}) \cdot b_j(O_{t+1}) \tag{4.16}$$

computation is used. Also it is necessary to store the state argument i that maximizes this function for each t and j, this will be kept in the vector $\psi_t(j)$.

Problem 3 is about training. So far there is no known way to analytically calculate the parameters of a model that maximizes the probability of an observation. However, the parameters can be locally maximized using an iterative hill-climbing method called Baum-Welch or EM(expectation modification)[63]. Let us explain Baum-Welch method. Define $x_t(i,j)$ as the probability of being in state S_i at time t and state S_j at time t+1.

$$x_t(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid O, l) \tag{4.17}$$

This can be calculated as [63]

$$x_t(i,j) = \frac{P(q_t = S_i, q_{t+1} = S_j, O \mid l)}{P(O \mid l)} = \frac{a_t(i) a_{ij} b_j(O_{t+1}) b_{t+1}(j)}{P(O \mid l)} \tag{4.18}$$

Let $g_t(i)$ be the probability of being in state S_i at time t given the sequence and the model.

$$g_t(i) = \sum_{j=1}^{N} x_t(i,j) \tag{4.19}$$

The expected number of transitions from S_i is then $\sum_{t=1}^{T-1} g_t(i)$. Then the expected number of transition from S_i to S_j is then $\sum_{t=1}^{T-1} x_t(i)$. Now we can estimate new values of the parameters given the observation as [63]:

$\overline{p_i}$ = Expected probability of being in state i at t=1 = $\gamma_1(i)$

$$\overline{a_{ij}} = \frac{\text{exp ected number of transitio ns from state i to state j}}{\text{expected number of transitio ns from state i}}$$

$$= \frac{\sum_{t=1}^{T-1} x_t(i,j)}{\sum_{t=1}^{T-1} g_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{exp ected number of times in state j observatin g v}_k}{\text{expected number of times in state j}}$$

$$= \frac{\sum_{t=1}^{T} g_t(j), O_t = v_k}{\sum_{t=1}^{T} g_t(i)}$$

It can be proven that the updated model, or say a new model $\bar{l}$ is then either [63]

- $\bar{l}$ =λ (We are at local maximum. This is also the stopping criterion for training) or
- $\bar{l}$ is better than λ regarding given observation, i.e. $P(O \mid \bar{l}) > P(O \mid l)$

Overall, the training step is to obtain a "maximum likelihood estimate" of an HMM for an observation. The flow of this algorithm can be described as follows [63]:

- Initialize λ= $\bar{l}$ =(A, B, π) to random estimates that satisfy the probabilistic constraints (see below)
- Repeat
  - Set λ: = $\bar{l}$
  - Calculate $\bar{A}, \bar{B}, \bar{p}$ based on O and λ and set $\bar{l}$: = $\bar{A}, \bar{B}, \bar{p}$.

  Until $\bar{l}$ = λ
- Always maintains probabilistic constraints:

$$\sum_{i=1}^{N} \bar{p}_i = 1, \quad \sum_{j=1}^{N} \bar{a}_{ij} = 1 (1 \le i \le N), \quad \sum_{k=1}^{L} \bar{b}_j(k) = 1 (1 \le j \le N)$$

In practice, it is impossible that $\bar{l}$ =λ. But they could be very close. Let $l^{(k-1)}$ denote the HMM l after k-1 iterations of Baum-Welch algorithm, and let $l^{(k)}$ denote the current iteration of Baum-Welch. Then, the training computation stops if

$$\frac{P(O \mid l^{(k)}) - P(O \mid l^{(k-1)})}{[P(O \mid l^{(k)}) - P(O \mid l^{(k-1)})]/2} < e_{HMM} \tag{4.20}$$

where e_{HMM} = 0.00001. In addition, in order to avoid computation overflow due to the multiplication of very small probability numbers, scaling up for too small probability values are applied if necessary. This scaling up does not affect the training of the HMM since the only useful information is the ratio of different probabilities and not their real values. As explained in the previous section, a model corresponds to a motion. So we need to train four separate HMMs. Obviously, problem 3(training) is the most difficult one of the HMM's three problems. Suppose the HMM for "path following" is initialized as follows: λ = (A, B,π).

$$p = [0.2, 0.2, 0.2, 0.2, 0.2]$$

$$A = \begin{bmatrix} 0.8 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.1 & 0.6 & 0.1 & 0.1 & 0.1 \\ 0.05 & 0.1 & 0.7 & 0.1 & 0.05 \\ 0.025 & 0.025 & 0.025 & 0.9 & 0.025 \\ 0.1 & 0.15 & 0.05 & 0.1 & 0.6 \end{bmatrix}$$

$$B = \begin{bmatrix} 1.0/256 & 1.0/256 & 1.0/256 & \cdots & 1.0/256 \\ 1.0/256 & 1.0/256 & 1.0/256 & \cdots & 1.0/256 \\ 1.0/256 & 1.0/256 & 1.0/256 & \cdots & 1.0/256 \\ 1.0/256 & 1.0/256 & 1.0/256 & \cdots & 1.0/256 \\ 1.0/256 & 1.0/256 & 1.0/256 & \cdots & 1.0/256 \end{bmatrix}$$

From these, we can see the probability constraints: the sum of the probability distribution from the current state to other states is "1"; at each state, the sum of the probability distribution of all possible observations is also "1". Using the observation sequences of "path following", the HMM is trained, that is, the probability parameters are adjusted. The trained HMM is expressed by the updated values until convergence occurs.

$$p = [0.08, 0.15, 0.28, 0.19, 0.29]$$

$$A = \begin{bmatrix} 0.87 & 0.06 & 0.0 & 0.06 & 0.01 \\ 0.0 & 0.70 & 0.05 & 0.21 & 0.04 \\ 0.0 & 0.0 & 0.26 & 0.74 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.87 & 0.13 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.0023 & 0.0 & 0.016 & \cdots & 0.0 \\ 0.013 & 0.0156 & 0.0 & \cdots & 0.006 \\ 0.0 & 0.0 & 0.003 & \cdots & 0.0025 \\ 0.0 & 0.0046 & 0.001 & \cdots & 0.0 \\ 0.009 & 0.0 & 0.001 & \cdots & 0.018 \end{bmatrix}$$

4.3.4. Motion Recognition

Once the four HMMs are trained by their corresponding training set, they can classify motions. The classification criterion is the forward score of a sequence of observations for a given model. This forward calculation is the same as the forward part of the Forward-Backward procedure used in solving problem 1.

$$a_1(i) = p_i b_i(o_1), \quad 1 \le i \le N \tag{4.21}$$

$$a_{t+1}(j) = \left[\sum_{i=1}^{N} a_t(i-1) a_{ij} \right] b_j(o_{t+1}), \quad \begin{cases} 1 \le t \le T-1 \\ 1 \le j \le N \end{cases} \tag{4.22}$$

$$P(o_1, o_2, \ldots o_t \mid l) = \sum_{j=1}^{N} a_T(j) \tag{4.23}$$

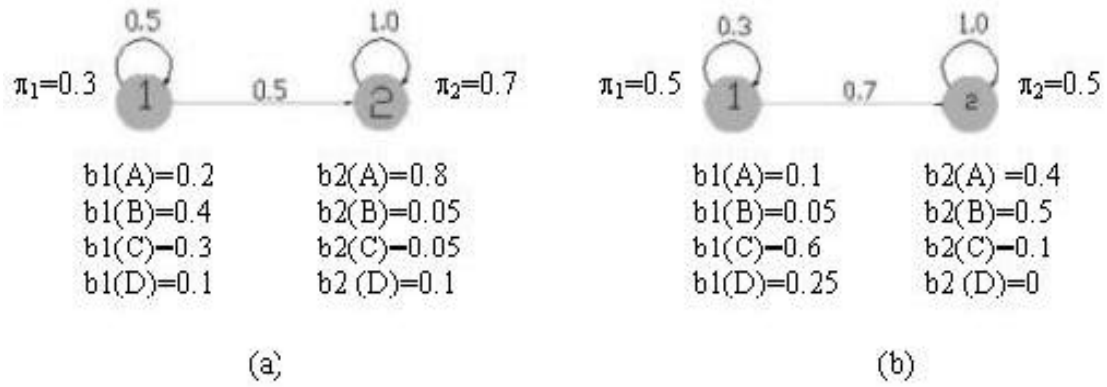


Figure 4.9 Forward Computation Illustration

Let us illustrate this computation by two one-dimensional, two-state, left-right HMMs as an example. Figure 4.9 shows two HMMs representing two classes. The length of the observation vector is 4. Therefore, at each time t, one of the four symbols, A, B, C or D will be observed for each state. From the structure of the first HMM (Figure 4.9 (a)), it's parameters are:

$$p = [0.3 \quad 0.7], \; A = \begin{bmatrix} 0.5 & 0.5 \\ 0.0 & 1.0 \end{bmatrix}, \; B = \begin{bmatrix} 0.2 & 0.8 \\ 0.4 & 0.05 \\ 0.3 & 0.05 \\ 0.1 & 0.1 \end{bmatrix}$$

For the given observation sequence ABA, its forward score is computed as follows:

$$a_1(1) = p_1 b_1(A) = 0.3 \times 0.2 = 0.06$$
$$a_1(2) = p_2 b_2(A) = 0.7 \times 0.8 = 0.56$$

$$a_2(1) = [a_1(1)a_{11} + a_1(2)a_{22}]b_1(B) = [0.06 \times 0.5 + 0.56 \times 0] \times 0.4 = 0.012$$
$$a_2(2) = [a_1(1)a_{12} + a_1(2)a_{22}]b_2(B) = [0.06 \times 0.5 + 0.56 \times 1.0] \times 0.05 = 0.0295$$
$$a_3(1) = [a_2(1)a_{11} + a_2(2)a_{21}]b_1(A) = [0.012 \times 0.5 + 0.0295 \times 0] \times 0.2 = 0.0012$$
$$a_3(2) = [a_2(1)a_{12} + a_2(2)a_{22}]b_2(A) = [0.012 \times 0.5 + 0.0295 \times 1] \times 0.8 = 0.0284$$

$$P(O = ABA \mid l_1) = a_3(1) + a_3(2) = 0.0296$$

This is the probability of the first HMM for the given observation sequence ABA. For the second HMM (Figure 4.9 (b)), it's parameters are:

$$p = [0.5 \quad 0.5], \; A = \begin{bmatrix} 0.3 & 0.7 \\ 0.0 & 1.0 \end{bmatrix}, \; B = \begin{bmatrix} 0.1 & 0.4 \\ 0.05 & 0.5 \\ 0.6 & 0.1 \\ 0.25 & 0.0 \end{bmatrix}$$

The forward score for the given observation sequence ABA is computed in exactly the same way:

$$a_1(1) = p_1 b_1(A) = 0.5 \times 0.1 = 0.05$$
$$a_1(2) = p_2 b_2(A) = 0.5 \times 0.4 = 0.2$$

$$a_2(1) = [a_1(1)a_{11} + a_1(2)a_{22}]b_1(B) = [0.05 \times 0.3 + 0.2 \times 0] \times 0.05 = 7.5e^{-4}$$
$$a_2(2) = [a_1(1)a_{12} + a_1(2)a_{22}]b_2(B) = [0.05 \times 0.7 + 0.2 \times 1] \times 0.5 = 0.1175$$
$$a_3(1) = [a_2(1)a_{11} + a_2(2)a_{21}]b_1(A) = [7.5e^{-4} \times 0.3 + 0.1175 \times 0] \times 0.1 = 2.25e^{-5}$$
$$a_3(2) = [a_2(1)a_{12} + a_2(2)a_{22}]b_2(A) = [7.5e^{-4} \times 0.7 + 0.1175 \times 1] \times 0.4 = 0.0472$$

$$P(O = ABA \mid l_2) = a_3(1) + a_3(2) = 0.0472$$

Since $P(O = ABA \mid l_2) = 0.0472 > P(O = ABA \mid l_1) = 0.0296$, it can be concluded that λ_2 is more likely to generate the observation sequence ABA. In other words, if we get the observation sequence ABA, the underlying process represented by HMM_2 has

been recognized. In our case, the HMMs have two dimensions and the length of each dimension of observation vector is 256. The successive four symbols obtained by data preprocessing are used for the partial observation sequence. It could be, for example, {20, 255, 120, 19}. This vector is used to compute the forward likelihood of the four HMMs as shown in the illustration above. Then for the given observation vector, we choose the model that has the largest likelihood as our recognized model at time t.

4.4. Design of Fixture Assistance

Once user's motion intentions are recognized, appropriate assistance can be designed for each motion. We define the path curve as p(s) and denote the target position by t. When the goal during task execution is to move to a target, we assume that the desired trajectory is a straight line that connects the current Cartesian position of the end-effector and the target. A preferred reference direction d can be defined for each point of the end-effector x_c as:

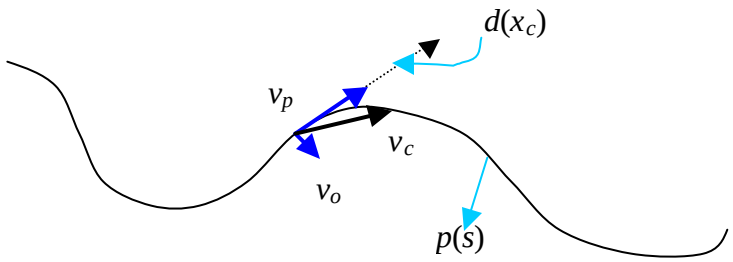


Figure 4.10 Virtual Fixture Definition

$$d(x_c) = \frac{x_t - x_c}{\|x_t - x_c\|} \tag{4.24}$$

Where x_t and x_c are the target position and the current position of the end-effector respectively. We decompose v_c, the current velocity, into two orthogonal components:

$$v_p = (v_c \cdot d)d \tag{4.25}$$

$$v_o = v_c - (v_c \cdot d)d \tag{4.26}$$

where v_p is the velocity component along the path curve tangent and v_o is the velocity component orthogonal to the curve tangent. The desired path following is such that the velocity tangent to the curve is large and velocity components in orthogonal direction are relatively small. If the desired trajectory of a sub-task is a straight line, a virtual fixture can provide the same assistance as a ruler helps in drawing a line.

4.4.1 Fixture Assistance

Fixture assistance is always applied for path following except when the user is trying to align an object or avoid an obstacle. So the stiffness coefficient k_d along the curve tangent is set to be zero. The stiffness orthogonal to the curve tangent is defined as:

$$k_o = \begin{cases} k_c & d > r \\ k_c(1 - d / r) & d \le r \end{cases} \tag{4.27}$$

where k_c is the fixture coefficient (it is determined to be 0.5N/mm for this experiment), d is the distance between the end-effector and the center position of the force fields, and r is the force fields radius. This means that once the end-effector goes inside force field, path following fixture is removed (See Figure 4.11 for fixture coefficient).

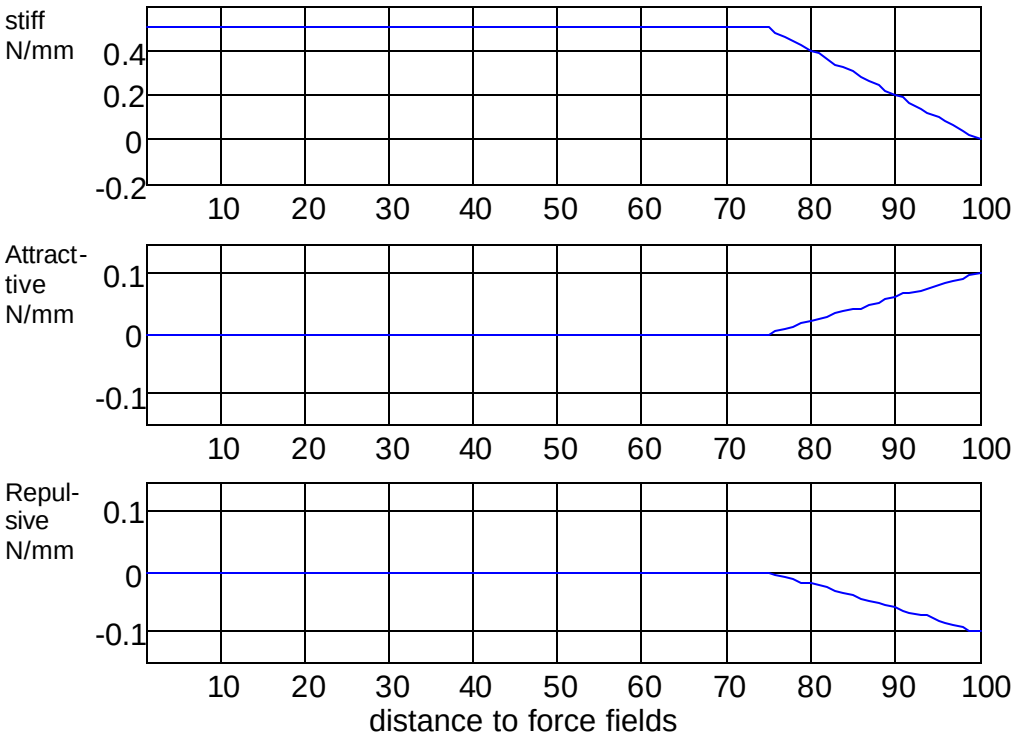


Figure 4.11 Stiffness Coefficients of Different Fixtures

4.4.2. Force Field Design for Targets and Obstacles

In general, aligning the end effector with a target and avoiding obstacles are not easy to execute, especially for persons with disabilities on the upper-limb. Potential fields generated from the center position of the target or the obstacle can provide some assistance. Based on this concept, force fields are designed around targets and obstacles. We define the radius of force field to be r. In this dissertation, the force field is defined using spring force. For approaching a target, the force is defined as:

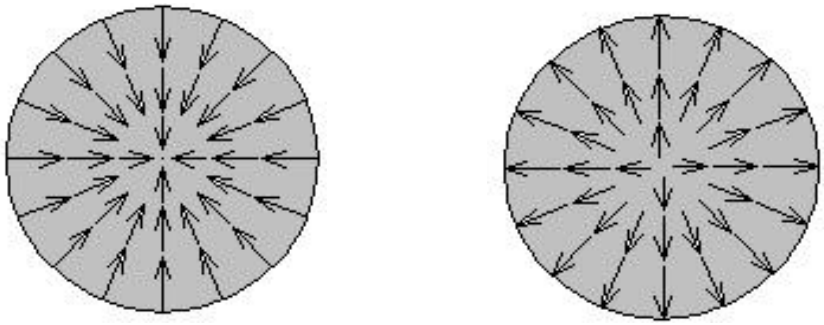
$$f = \begin{cases} 0 & d > r \\ k_f (r - d) & d \le r \end{cases} \tag{4.28}$$

where k_f is 0.1N/mm.

For obstacle avoidance, the force is defined as:

$$f = \begin{cases} 0 & d > r \\ -k_f (r - d) & d \le r \end{cases} \tag{4.29}$$

where k_f is 0.1N/mm. Once the end-effector goes within the radius r for aligning with the target, the attractive force originated from the object center position can provide assistance. The force vectors generated by position and approach fixtures are shown in Figure 4.12. Payandeh et al used such virtual fixture as a task-dependent telemanipulation aid [5, 14]. However, the origin of the force fields needs to be determined from the sensory data. In addition, r should be larger than the size of the target or the obstacle.



(a) (b)

Figure 4.12 Force Fields Illustration (a: Attractive force, b: Repulsive force)

4.5. Experiments

We have implemented the algorithm described above, and conducted experiments to determine the system's performance without and with the assistance.

4.5.1. Experimental Test Bed

Our telemanipulation simulation system is composed of a visualization scene and a haptic device. The visualization component, simulation scene, is realized through the PhanToM and GHOST [48]. In this experiment, the task is to move the end-effector from

the origin (0,0,0) to (-80,50,0), referred to as target "Grasp" (this means the end-effector must reside in the object sphere for a short time) and then avoid the obstacle (0,45,0) and then put the target at the "target destination" (80,50,0) and go back to the origin. The target "grasp" and the target "destination" are simulated as 8mm radius spheres. The obstacle and the end-effector are simulated as 15mm and 5mm radius spheres; respectively. User is asked to move the end-effector as fast and as smoothly as possible (Figure 4.13). In order to avoid confusion, the operator is allowed to move on a planar surface and a planar constraint is added to the haptic device. In this experiment, we are concerned about the straight-line path since it is relatively easy to obtain from the environment information. This algorithm can be extended to a complex trajectory application if we can define the trajectory using visual information for the unstructured environment.

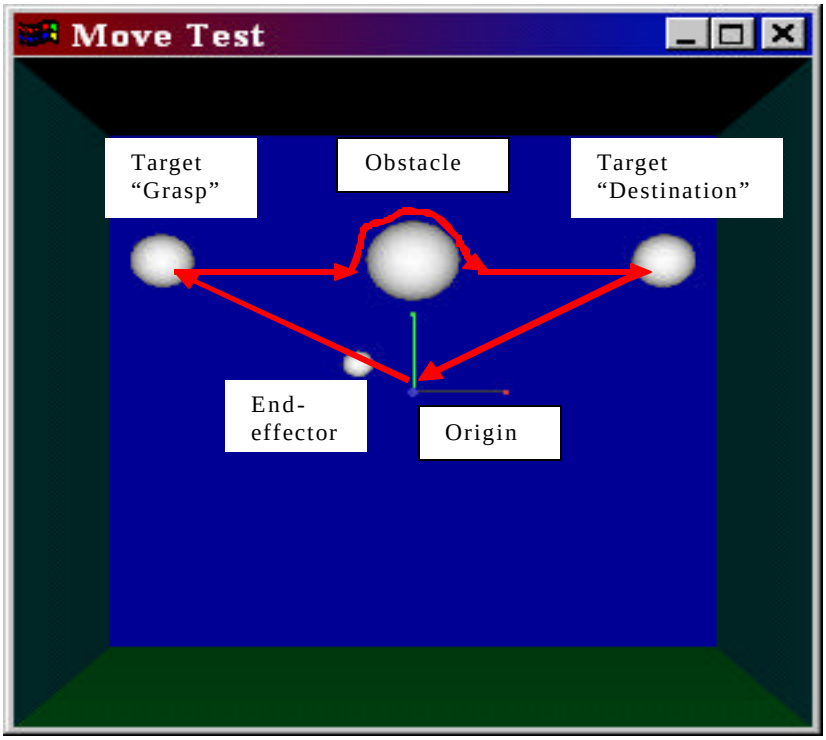


Figure 4.13 Simulation of the Task Execution

4.5.2. Experimental Results Without Assistance

First, an expert user completed the task several times without assistance. During the first several tests, the common performance of the system is shown in Figure 4.14 and 4.15. As expected, the free motion has much difficulty in aligning with the target and following the path. The velocity components orthogonal to the path are not small compared to the useful velocity components tangential to the path. Table 4.1 summarizes the results, including path following error (mm) and execution time(s).

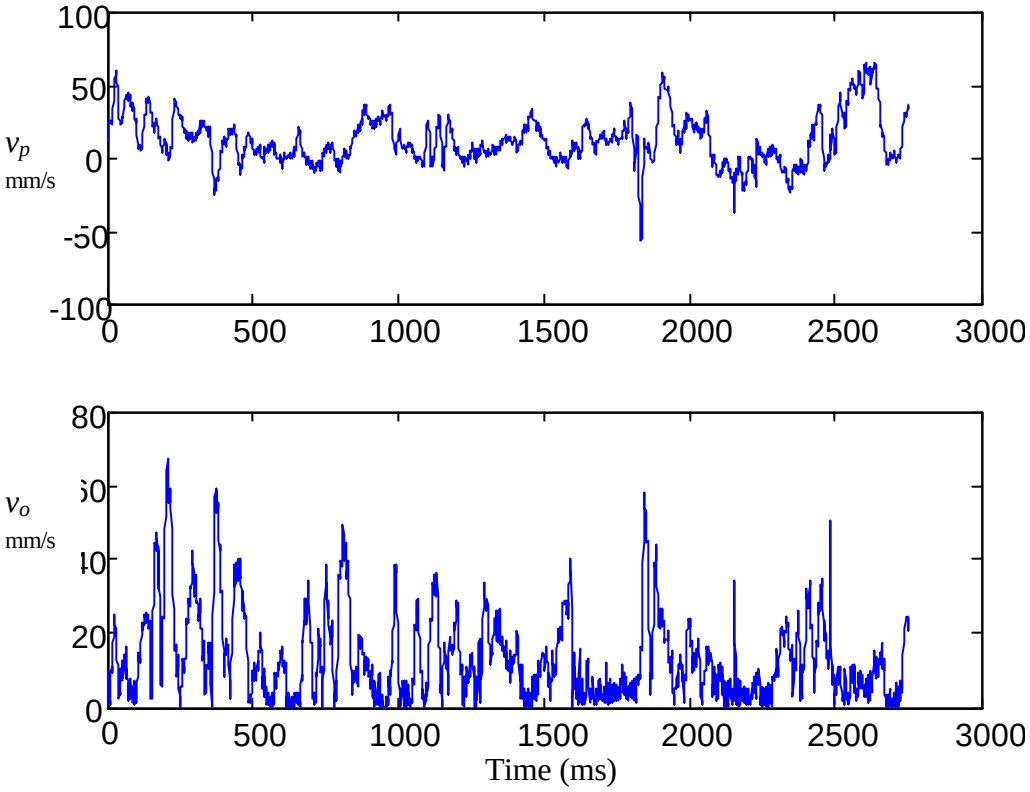


Figure 4.14 Velocity Components Without Assistance

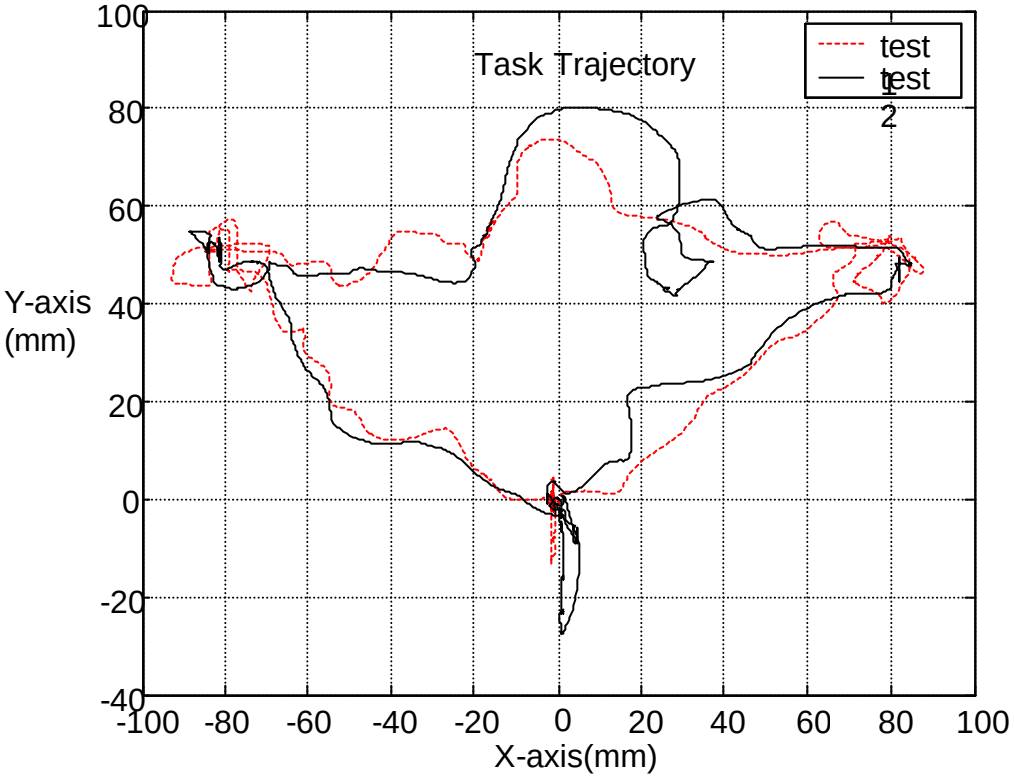


Figure 4.15 Trajectories without Assistance

Table 4.1 Performance Summary without Assistance

Subject	Path Error (mm)		Execution Time(s)	
	Mean	Stdev	Mean	Stdev
1	10.1	2.4	21.5	1.9
2	8.9	1.5	20.2	3.3
3	11.8	2.6	22.1	3.4
4	10.3	2.5	20.4	2.8

4.5.3. Motion Recognition

For the task used in this dissertation, four users in the lab completed the task for 10 times each. We collected 250 samples of data for each motion, the first 200 for training and the rest of the samples are for testing. For a total 50 testing samples of four motions, the system successfully recognized 43 samples. The accuracy is 86%. Definitely, the size of the training set influences the recognition accuracy. After we included 500 samples into the training set, the system recognized 92 samples from 100 testing samples. The motion recognition performance is shown in Table 4.2.

Table 4.2 Motion Recognition Rate

Motion	Correct rate	Incorrect rate			
		to 1	to 2	to 3	to 4
1: Path following	90.5%	----	4.0%	2.3%	3.2%
2: Target aligning	89.1%	6.4%	----	2.3%	2.2%
3: Obstacle avoidance	88.3%	7.7%	2.0%	----	2.0%
4: Stopping	98.7%	0.0%	1.3%	0.0%	----

4.5.4. Experimental Results with Assistance Based on Motion Intention Recognition

As mentioned before, the resultant assistance is applied to each motion of a task. If the motion at a certain stage is path following, a hard fixture is applied so that the end-effector can move along the path. Once the motion has been changed into "aligning with a target" motion, hard fixture is replaced by an attractive force field. For avoiding an obstacle, a repulsive force field is applied. If the motion is classified as stopping, no assistance is applied. In general, the shape of an obstacle is difficult to determine from the sensory information of the environment. So creating a desired path for obstacle

avoidance is not feasible. This repulsive force field provides assistance for the operator to go around the obstacle. With these assistances, four users executed the same tasks for multiple times. Every time, the system performance was consistent and had very little variation. Two random trajectories from different subjects are shown below. The fixture helped significantly for path following, primarily due to the fact that the constraints applied to the PhanToM tool tip could force it to back up once there was some deviation from the path. Most of the time, the velocity component was much smaller compared to the velocity component tangential to the path. The large orthogonal velocity occurs when the user is aligning with a target or avoiding an obstacle.

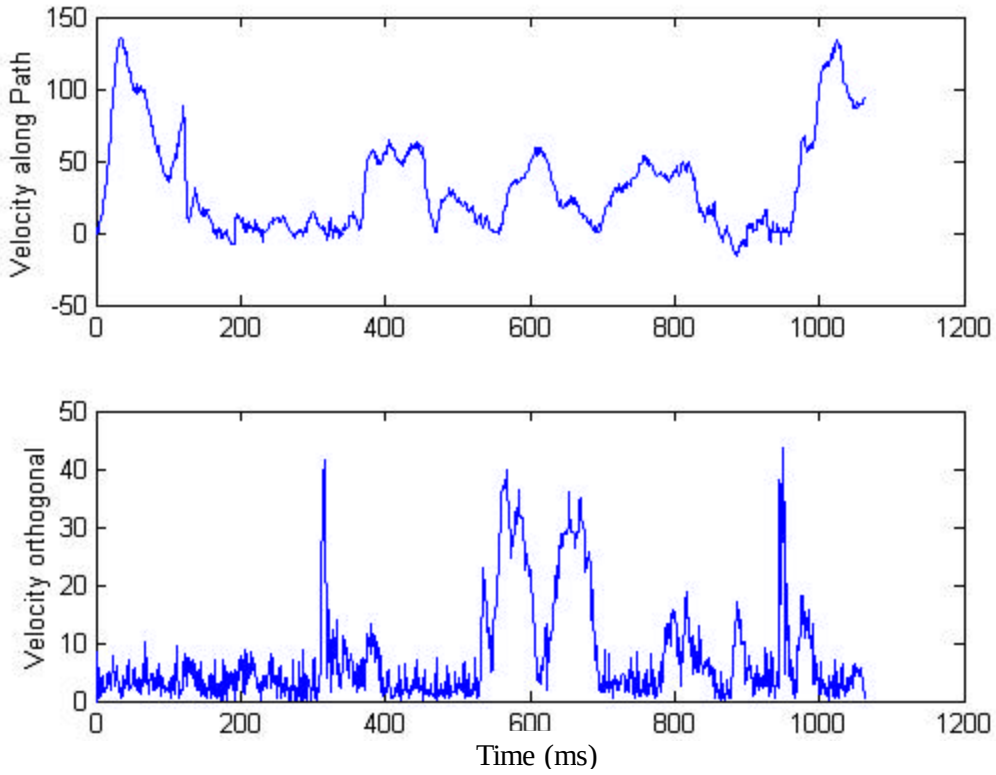


Figure 4.16 Velocity Components with Assistance

Table 4.3 Performance Summaries with Assistance

Subject	Path Error (mm)		Execution Time(s)	
	Mean	Stdev	Mean	Stdev
1	5.1	0.5	12.6	0.7
2	4.6	0.8	11.8	0.6
3	5.3	0.9	12.9	1.2
4	4.8	1.1	13.4	1.2

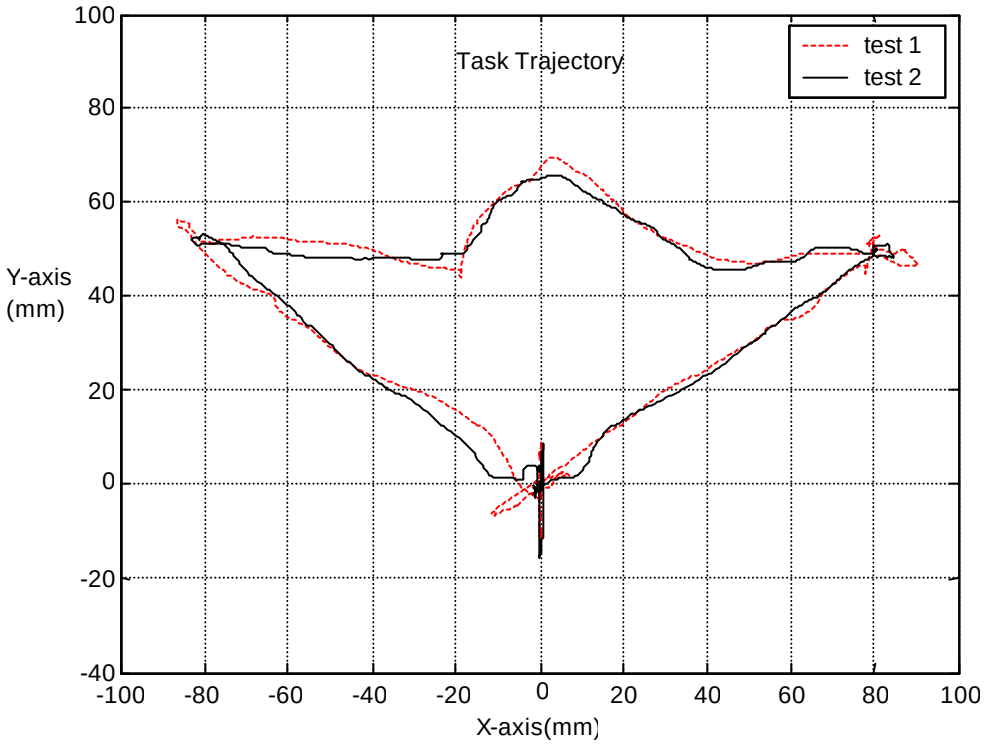


Figure 4.17 Trajectories with Assistance

4.6. Summary

Hidden Markov Model is effective for the classification of random processes such as human's motion intention in a teleoperation task. As long as the training set is sufficiently large, the motion recognition accuracy is close to 100%. The selected assistance based on the recognized motion is appropriate for each type of motion. The

experimental results without assistance have shown that the operator always has random errors that result in difficulty in following a path and aligning with a target. The experimental results with assistance showed that the undesired random errors were removed or reduced. The HMM based assistance is useful for improving performance accuracy and decreasing execution time. These results indicate that the appropriate assistance approach selection based on motion intention is possible. Based on the operator's motion intention, it is possible to determine if an object is a target or an obstacle. In order to improve the recognition accuracy, the dimension number of the Hidden Markov Model can be expanded. As long as they all are independent, the added dimensions will only linearly increase the computational requirements.

Chapter 5: Robotic Therapy for Persons with Disabilities Using Skill Learning

This chapter describes the Hidden Markov Model (HMM)-based skill learning and its application in a motion therapy system using a haptic interface. A relatively complex task, moving along a labyrinth, is used. A normal subject executes this task for a number of times and the labyrinth skill is learned by Hidden Markov Model. The learned skill is considered as a virtual therapist who can train persons with disabilities to complete the task. Two persons with disabilities on upper limb (cerebral palsy) were trained by the virtual therapist. The performance before and after therapy training, including the smoothness of the trajectory, distance ratio, time taken, tremor and impact forces are presented in this chapter. This labyrinth can be used as a therapy platform for upper limb coordination, tremor reduction and motion control improving.

5.1. Motion Therapy

Much evidence suggests that intensive therapy improves movement recovery [78, 79]. But such therapy is expensive, because it requires therapists on a person-to-person basis. Recently, there has been an increased interest in restoring functions through robot-aided therapy. This approach is to design therapy platform, such as force fields and moving constraints, to substitute therapist's work. In this chapter, the role of the therapist is replaced by the learned skill. When humans execute a task, their actions reflect the skill associated with that task. When one does a particular task many times, each time the

performance is different even though it represents the same skill. For example, when one draws 50 circles of the same radius by hand, each circle will be different from the others although they may look close. But any one of the 50 circles is the result from operator's circle-drawing skill. The different looking of these circles is due to the random control commands from brain and the random movements of hand. Since Hidden Markov Model is feasible to model a stochastic process, such as speech signal, it is possible to characterize the skill of the upper-limb motion for a specific task. In this dissertation, we have modeled the human movement along a labyrinth so that the underlying nature of it is revealed and can be used to transfer the skill to people with disabilities. It is desired that persons with disabilities can be trained for manipulation capabilities, which are incrementally improved through learning practice. Learning from observation is a paradigm where one observes other persons' performance and learns from it. This is also like physical therapy for a specific disability.

5.2. Hidden Markov Model Based Skill Learning

In this dissertation, we model the motion of moving along a labyrinth task skill using HMM. In order for the user to visualize the virtual therapist more effectively, the trajectory of the movement is chosen as the skill for learning. Since we only consider the movement in *X*-*Y* plane, position coordinates, *Px* and *Py* are used to represent the movement. In chapter 4, it has been explained how to convert continuous velocity data into discrete symbols. Similar procedures are used in this chapter to convert continuous position data into discrete symbols.

5.2.1. Raw-data Conversion

The raw data used by HMM for motion intention in chapter 4 is the user's velocity. In this chapter, the raw data is the translation trajectory, P_x, P_y. In order to use discrete HMM, we still need to convert raw data into symbols. The procedures will be explained in this section.

First of all, the translation trajectory is sampled by 1000Hz rate. Since P_x and P_y are independent vectors and processed in the same way, we just demonstrate the preprocessing procedures of P_x. For simplicity, we use an example with less data. Let us assume that the position samples for a specific task result in the following 3 vectors.

V1 = [45.8066 36.9727 19.1504 16.2247 19.1068 29.9084 40.7183 17.3202 46.9558 31.8121 20.7432 39.3534 30.6080 24.9133 38.8958 34.7934];

V2 = [63.5857 76.5475 41.8072 70.4114 13.8365 78.3798 21.7158 20.1863 70.0594 58.9845 10.9215 0.9405 71.5118 15.9310 23.8978 52.9154];

V3 = [13.6516 22.5228 3.1095 47.4401 27.9740 20.3278 24.7446 16.0297 20.7795 10.8456 27.8307 36.4975 25.4315 30.7453 10.0353 18.2313];

The vector length is 16 points. In other words, we cut every 16 points and form a vector. These vectors are so called raw data. Their waveforms are shown in Figure 5.1. They do not have much useful information, just like our voice signal waveform in time domain. So we need to do some transformation. As illustrated in chapter 4, each raw data vector is multiplied by a Hamming window and then transformed by 16-point FFT. It is well known that the result of FFT is a symmetrical vector. So in order to reduce computation complexity, only half of the FFT result is used in PSD computation. The 3 vectors shown previously are transformed into the following 3 vectors with 8-point length.

P1 = 10^4 *[1.0271 0.2550 0.0111 0.0010 0.0049 0.0241 0.0320 0.0154];

P2 = 10^4 *[1.8429 0.3981 0.0195 0.0518 0.1750 0.1967 0.0299 0.0931];

P3 = 10^3 *[5.8727 0.9417 0.1443 0.0256 0.0841 0.0292 0.0769].

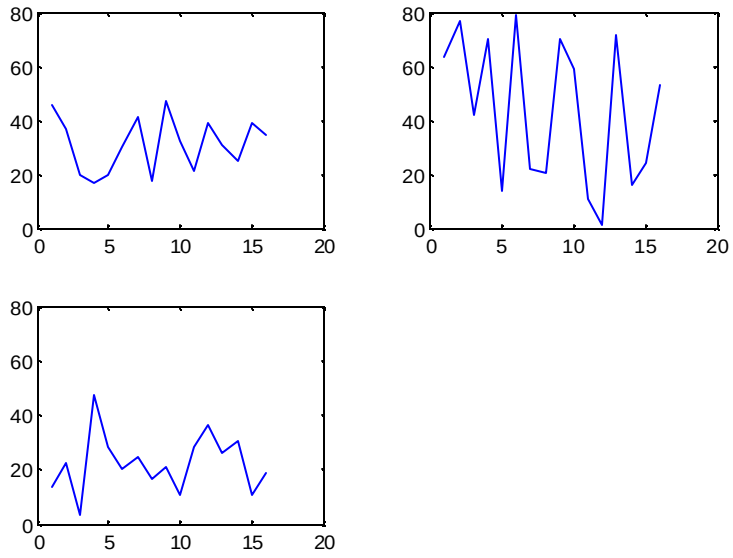


Figure 5.1 Raw-data Vectors

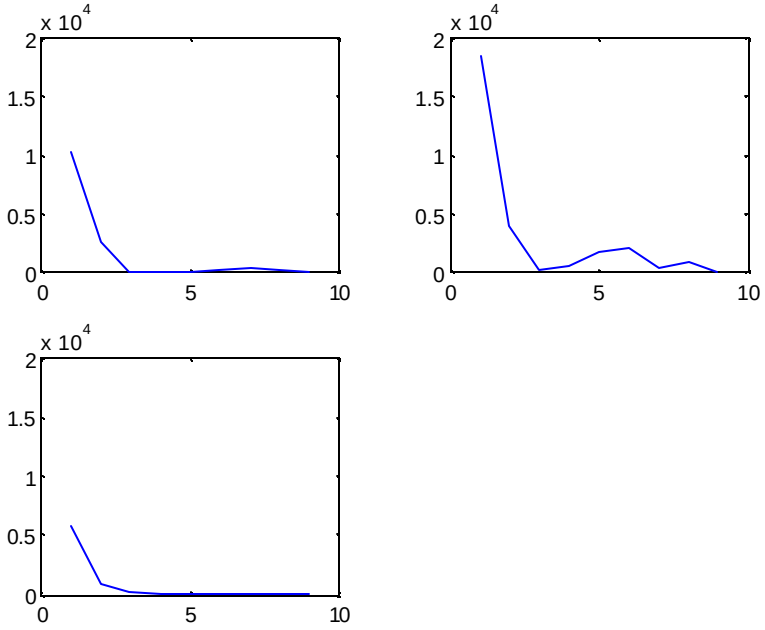


Figure 5.2 PSD Vectors

If this task is executed 10 times, we will have 30 PSD vectors. For a simple task, we can use all these 30 vectors in the computations. But for general applications in real life, this number could be very huge. It is impossible to do the computations using all of these vectors. This is why we need to do vector quantization. As for as vector quantization, it is an algorithm to group vectors into different clusters according to the vector distance criteria. The number of the clusters is determined based on the application and accuracy. For some simple applications, usually 32 or 64 will be enough. The set and the number of the clusters are called codebook and codebook length. The clusters are called codewords of the codebook. The larger is the codebook length, the higher accuracy is the grouping. For this simple example, the length of the codebook for vector quantization is determined to be 4. Figure 5.3 shows the illustration of vector quantization when the codebook length is 4. In other words, the vector quantization is to divide the whole vectors set into 4 clusters according to how the vectors are close to each other. There are many available vector quantization algorithms in literature. The well known one is LBG [72].

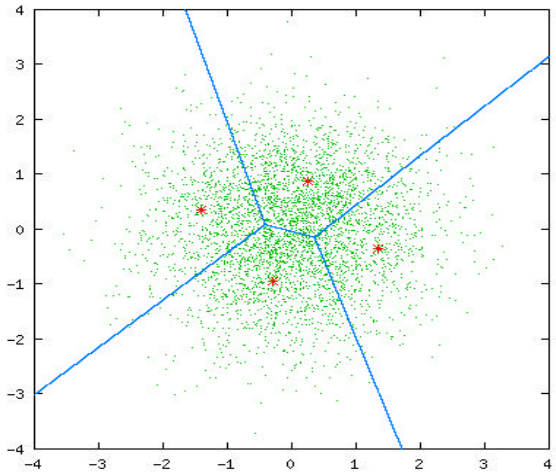


Figure 5.3 Vector Quantization When Codebook Length is 4

Once the codebook is obtained, we can use it as a template to convert any vector into discrete symbols. As a matter of fact, the 30 vectors used in vector quantization can also be represented by symbols. If we represent each cluster by a symbol, (for example, "A" or "1" represents cluster 1, "B" or "2" represents cluster 2 and so on), we may express the 30 PSD vectors as "ABACDCDBACDCDABACDCAABDACDABDB" or "12134342134341213431124134124 2". This is the result of data preprocessing. When new vectors come in, they will be compared with codewords and placed into the corresponding clusters with which the vectors are closest, thus converting raw position data into discrete symbols. We did this so that we can use discrete Hidden Markov Model to do all computation.

5.2.2. Hidden Markov Model Computation

Let us assume that we executed this task 3 times to do skill learning. We need to determine which one of the three task executions represents our skill. For each task execution, their raw position data is preprocessed and converted into 3 discrete symbols. Let us assume that the symbols from the first task execution are "ABA", the second one "CBD", and the third one "BDB". All symbols from these three task executions will be used as the training set. So the training set for HMM is "ABACBDBDB". In order to explain the computation clearly, we use a two states left-right HMM as shown below.

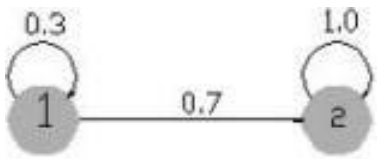


Figure 5.4 Two-state Left-right Hidden Markov Model

Before training, all parameters of HMM are initialized by randomly generated probability values, as shown below.

$$p = [0.35 \quad 0.65], \; A = \begin{bmatrix} 0.25 & 0.75 \\ 0.56 & 0.44 \end{bmatrix}, \; B = \begin{bmatrix} 0.21 & 0.78 \\ 0.44 & 0.05 \\ 0.13 & 0.15 \\ 0.22 & 0.02 \end{bmatrix}$$

Using the training set "ABACBDBDB", these HMM parameters are updated using the same algorithm explained in chapter 4. After training, the HMM parameters are:

$$p = [0.5 \quad 0.5], \; A = \begin{bmatrix} 0.3 & 0.7 \\ 0.0 & 1.0 \end{bmatrix}, \; B = \begin{bmatrix} 0.1 & 0.4 \\ 0.05 & 0.5 \\ 0.6 & 0.1 \\ 0.25 & 0.0 \end{bmatrix}$$

The HMM with the adjusted parameters is shown in Figure 5.5:

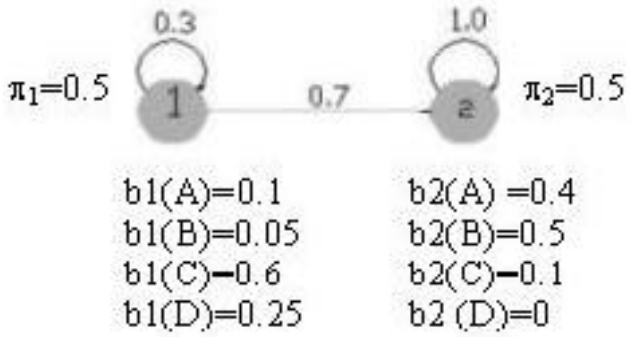


Figure 5.5 Hidden Markov Model with the Adjusted Parameters

Once the HMM is trained by the training set, they can be used to evaluate any given observation sequence. The evaluation criterion is the forward score of a sequence

of observations given in a model. The forward score of a given observation sequence is computed as follows:

$$a_1(i) = p_i b_i(o_1), \quad 1 \le i \le N \tag{5.1}$$

$$a_{t+1}(j) = \left[\sum_{i=1}^{N} a_t(i-1) a_{ij} \right] b_j(o_{t+1}), \quad \begin{cases} 1 \le t \le T-1 \\ 1 \le j \le N \end{cases} \tag{5.2}$$

$$P(o_1, o_2, \ldots o_t \mid l) = \sum_{j=1}^{N} a_T(j) \tag{5.3}$$

where N is the number of states (2 in this case), T is the time corresponding to a symbol. For the HMM trained by the combination of the three-time execution data, we might as well evaluate the forward score of each task execution.

For the first task execution, the observation sequence is “ABA”. The forward score of this observation set is computed as following:

$$a_1(1) = p_1 b_1(A) = 0.5 \times 0.1 = 0.05$$
$$a_1(2) = p_2 b_2(A) = 0.5 \times 0.4 = 0.2$$

$$a_2(1) = [a_1(1)a_{11} + a_1(2)a_{22}] b_1(B) = [0.05 \times 0.3 + 0.2 \times 0] \times 0.05 = 7.5e^{-4}$$
$$a_2(2) = [a_1(1)a_{12} + a_1(2)a_{22}] b_2(B) = [0.05 \times 0.7 + 0.2 \times 1] \times 0.5 = 0.1175$$

$$a_3(1) = [a_2(1)a_{11} + a_2(2)a_{21}] b_1(A) = [7.5e^{-4} \times 0.3 + 0.1175 \times 0] \times 0.1 = 2.25e^{-5}$$
$$a_3(2) = [a_2(1)a_{12} + a_2(2)a_{22}] b_2(A) = [7.5e^{-4} \times 0.7 + 0.1175 \times 1] \times 0.4 = 0.0472$$

$$P(O = ABA \mid l_2) = a_3(1) + a_3(2) = 0.0472$$

So the forward score of the observation sequence “ABA” is 0.0472.

For the second task execution, the observation sequence is "CBD". The forward score of this observation sequence is computed in the same way:

$$a_1(1) = p_1 b_1(C) = 0.5 \times 0.6 = 0.3$$
$$a_1(2) = p_2 b_2(C) = 0.5 \times 0.1 = 0.05$$

$$a_2(1) = [a_1(1)a_{11} + a_1(2)a_{22}]b_1(B) = [0.3 \times 0.3 + 0.05 \times 0] \times 0.05 = 4.5e^{-3}$$
$$a_2(2) = [a_1(1)a_{12} + a_1(2)a_{22}]b_2(B) = [0.3 \times 0.7 + 0.05 \times 1] \times 0.5 = 0.13$$

$$a_3(1) = [a_2(1)a_{11} + a_2(2)a_{21}]b_1(D) = [4.5e^{-3} \times 0.3 + 0.13 \times 0] \times 0.25 = 3.375e^{-4}$$
$$a_3(2) = [a_2(1)a_{12} + a_2(2)a_{22}]b_2(D) = [4.5e^{-4} \times 0.7 + 0.13 \times 1] \times 0 = 0.0$$

$$P(O = CBD \mid l) = a_3(1) + a_3(2) = 3.375e^{-4}$$

For the third task execution, the observation sequence is "BDB". The same way, its forward score is:

$$P(O = BDB \mid l) = a_3(1) + a_3(2) = 6.844e^{-4}$$

Since

$$P(O = ABA \mid l) = 0.0472 > P(O = BDB \mid l) = 6.844e^{-4} > P(O = CBD \mid l) = 3.375e^{-4}$$

It can be concluded that the task execution with "ABA" observation represent the task skill more closely than the other two observation sequences. In other words, the task execution whose observation sequence has the highest forward score represents the task skill.

5.3. Experiments in Virtual Environment

5.3.1. Tasks and Experimental Test-Bed

To evaluate the validity and effectiveness of the HMM for skill learning and its application for therapy, we designed a haptic interactive simulation test bed (Figure 5.6).

It is composed of a visualization scene and a PhanToM Premium 1.5 [48]. The PhanToM is an impedance haptic device that can provide force reflection to operators if collision happens. The simulation scene is realized through API functions of GHOST [48]. The end-effector is simulated as a sphere whose radius is 5mm. The width of the labyrinth is 18 mm. In this experiment, the task is defined to move the end-effector from the origin (0, 0, 0) to get out of the labyrinth as quickly and smoothly as possible, and with as few collisions as possible. In order to avoid the depth perception problem, operators are only allowed to move in the *X-Y* plane by adding a planar constraint to the haptic device. Bardorfer et al used this haptic interface to do motion analysis of upper-limb for patients with neurological diseases (ND), but they did not try to improve the manipulation performance [80].

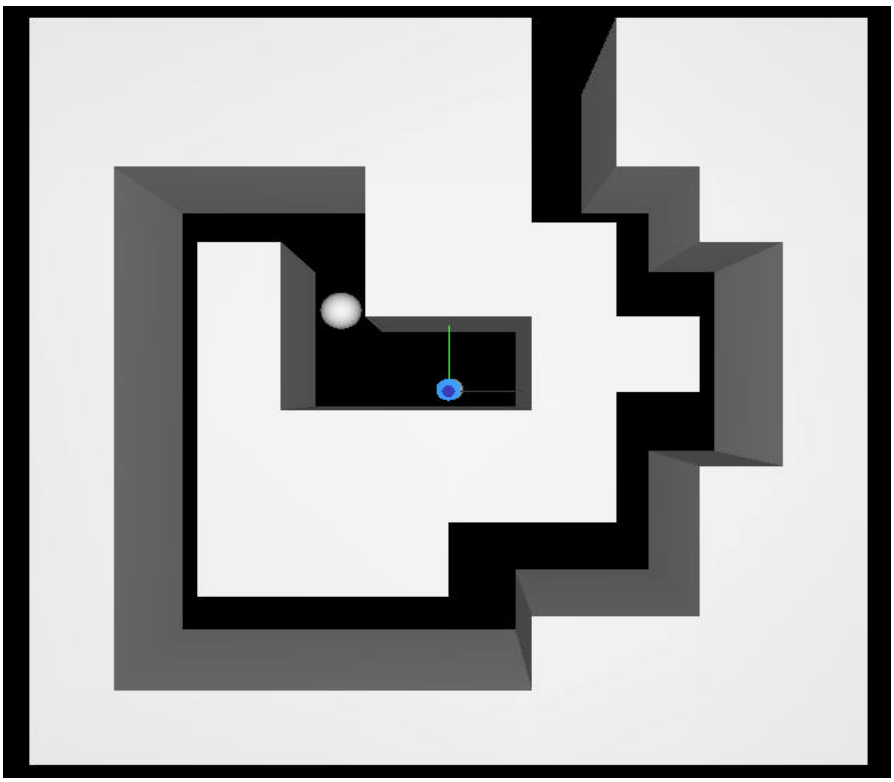


Figure 5.6 Virtual Environment for Simulation Test-bed

5.3.2. Skill Learning and Transferring

HMM is used to model the translation skill of moving along the labyrinth. The learned skill is later used as a virtual therapist in motion training. This task was executed twelve times to produce the training set for HMM by a normal subject. The translation data of the end-effector is recorded and converted into discrete symbols using the preprocessing approach as illustrated in section 5.2.1. The discrete symbols of these task executions are used to train HMM. Once the HMM has been trained, it can be used to evaluate each task execution. The set of symbols that produce the largest forward likelihood $P(O|M)$ correspond to the motion that is most likely executed by the normal subject. In other words, it represents the skill needed by that specific task. We use a 5-state, left-right, two dimensional HMM for skill learning. So the prior matrix π is a 1×5 matrix, the transition matrix A is a 5×5 matrix with each row representing the transition probability from a certain state to other states. It is necessary to note that we have two observability matrices B, each of which is 256×5. π, A and B matrices are initialized by the uniformly distributed random number as usual. Starting with these initial parameters, the HMM is trained by the training test. The forward algorithm was used to score each trajectory (Figure 5.7). It can be seen that No. 7 is the highest and No. 6 is the lowest in the probability values. It is important to note that the best (highest) or worst (lowest) scores do not refer to the performance, but to the accuracy of representing the skill of doing the task. For example, if we are asked to draw many line segments with the same direction and length, it would be likely that we would draw a couple of close to perfect ones and a couple of very bad ones. But these extreme cases do not represent our line-drawing skill. The lines that we are most likely to draw represent our line-drawing skill.

The trajectory with the highest score represents the translation skill of the subject most likely to do this task.

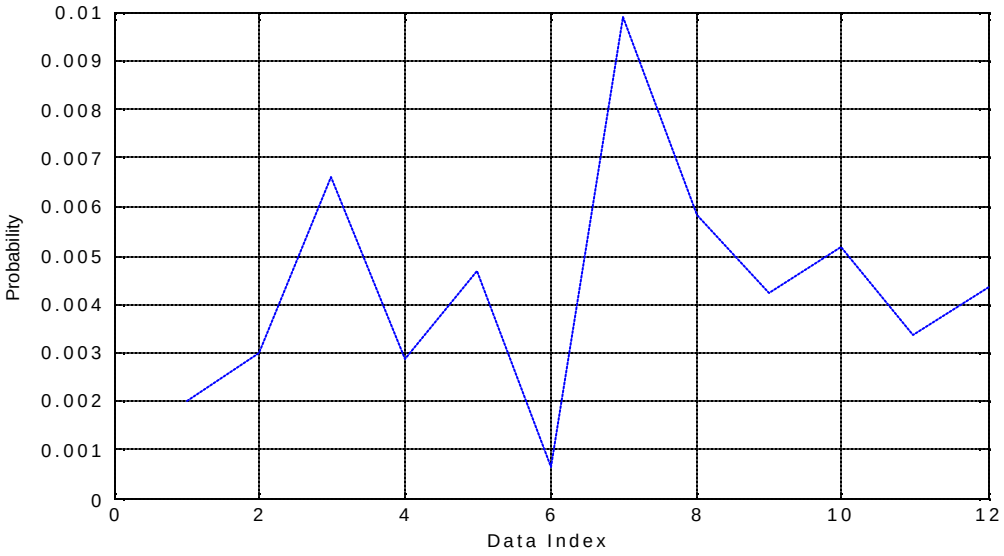


Figure 5.7 Forward Scores for all 12 Times of Task Execution

5.4. Motion Therapy Experiments

Since the skill of this task has been learned, the trajectory of the learned skill is displayed on the screen acting as a therapist. During the therapy training session, operators try to follow it as accurately as possible (Figure 5.6). Two subjects: one is female, cerebral palsy with right hemiparesis and spasticity, persistent low back pain; the other is male, 19, cerebral palsy, partial paralysis of his upper and lower extremities, executed this task seven times each before and after training. Before collecting data, they practiced this movement for several times until they felt comfortable about it. Their data, including translation, velocity and reaction forces, were sampled at 1000Hz. The evaluation indexes include:

- Distance ratio R_d. Its value reflects the trajectory optimization capabilities. The smaller, the better. The ideal value is a little greater than 1.

$$R_d = \frac{d_{actual}}{d_{skill}} \tag{5.4}$$

where d_{actual} is the actual distance traveled, d_{skill} is the distance traveled by the learned skill.

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

$$C(n) = \begin{cases} 1, & Collide \quad with\ wall \\ 0, & elsewhere \end{cases} \tag{5.5}$$

$$N_c = \sum_j (C(j+1) - C(j) = 1) \tag{5.6}$$

- Time duration of the collisions T_i. It reflects reaction capabilities.

$$T_i = t_{j, C(j+1)-C(j)=-1} - t_{j, C(j+1)-C(j)=1} \tag{5.7}$$

- Impact force of the collisions with walls F_i;

$$F_i = \sqrt{F_{i,x}^2 + F_{i,y}^2} \tag{5.8}$$

where $F_{i,x}$ and $F_{i,y}$ are impact forces when the end-effector collides with the X-direction wall and Y-direction wall respectively.

- Tremor magnitude M_t and frequency F_t.

For motion analysis, operator's collisions with X-directional wall and Y-direction wall do not make much difference. So only the magnitude of impact force is analyzed. The direction of impact force is not meaningful. Tremor information is extracted by

applying a high pass filter, which has a cut-off frequency *fc = fmax/10 (fmax* is maximum tremor frequency*)*. Tremor magnitude is available in time domain. The tremor frequency can be obtained through discrete Fourier transform (DFT). The collision forces along *X*- and *Y*-axes are combined and the magnitude of the combined force was analyzed. *C(n)* indicates the case when collisions occur. *Nc* is the number of collisions occurring during task execution. *Ti* is the time duration of each collision. *Nc* and *Ti* are obtained through checking the transition of *C(n)* between 0 and 1.

5.4.1. Motion Performance before Therapy Training

Two persons with disabilities performed the task before and after therapy training. Figures 5.8, 5.9, and 5.10 present the performance of subject 1 before training. Figure 5.8 shows an actual trajectory and the skilled trajectory. Figure 59 shows the translation tremor along X and Y-axes, including tremor magnitude and frequency. Figure 5.10 presents the collision information, including the impact force and the time duration for each collision occurring.

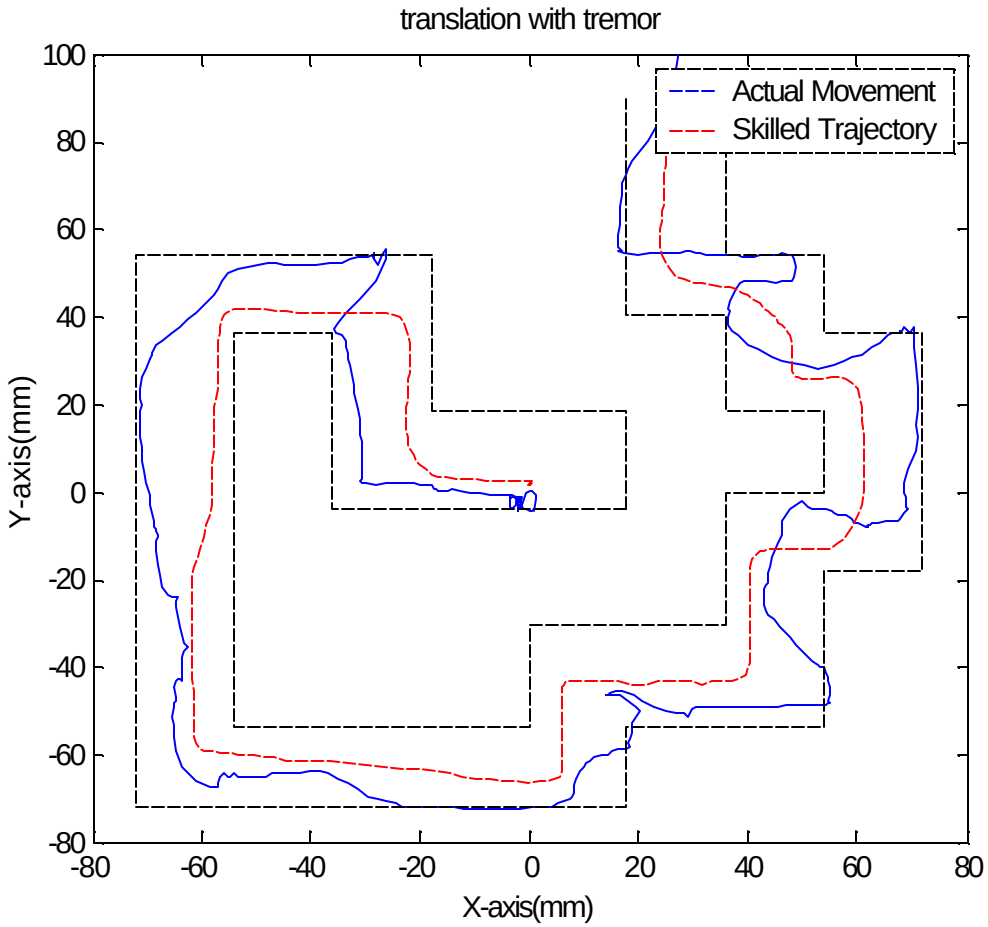


Figure 5.8 Actual Moving Distance is 716.8mm, Skill Moving Distance is 495.2mm, and Distance Ratio is 1.44

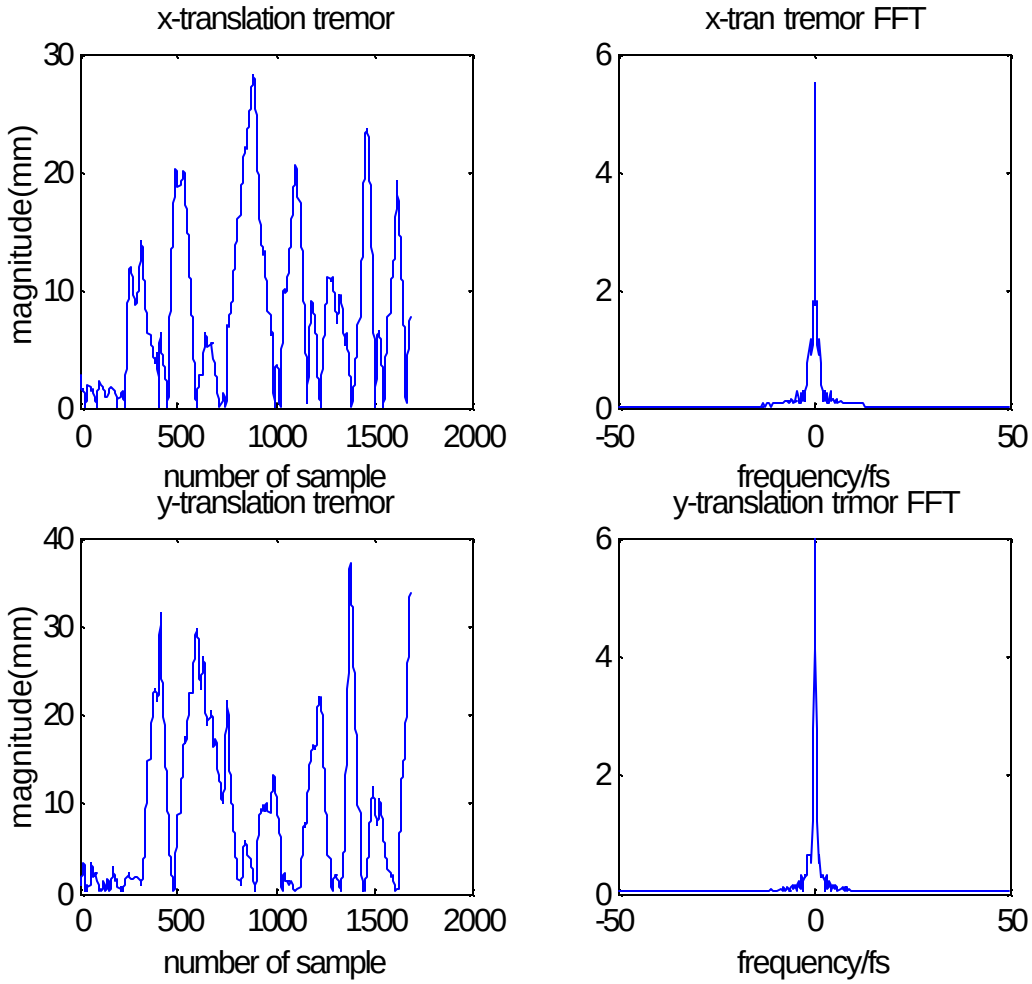


Figure 5.9 Tremor Measurements. X tremor magnitude mean is 8.4mm and STD is 6.9mm. Y tremor magnitude mean is 9.3mm and STD is 8.8mm

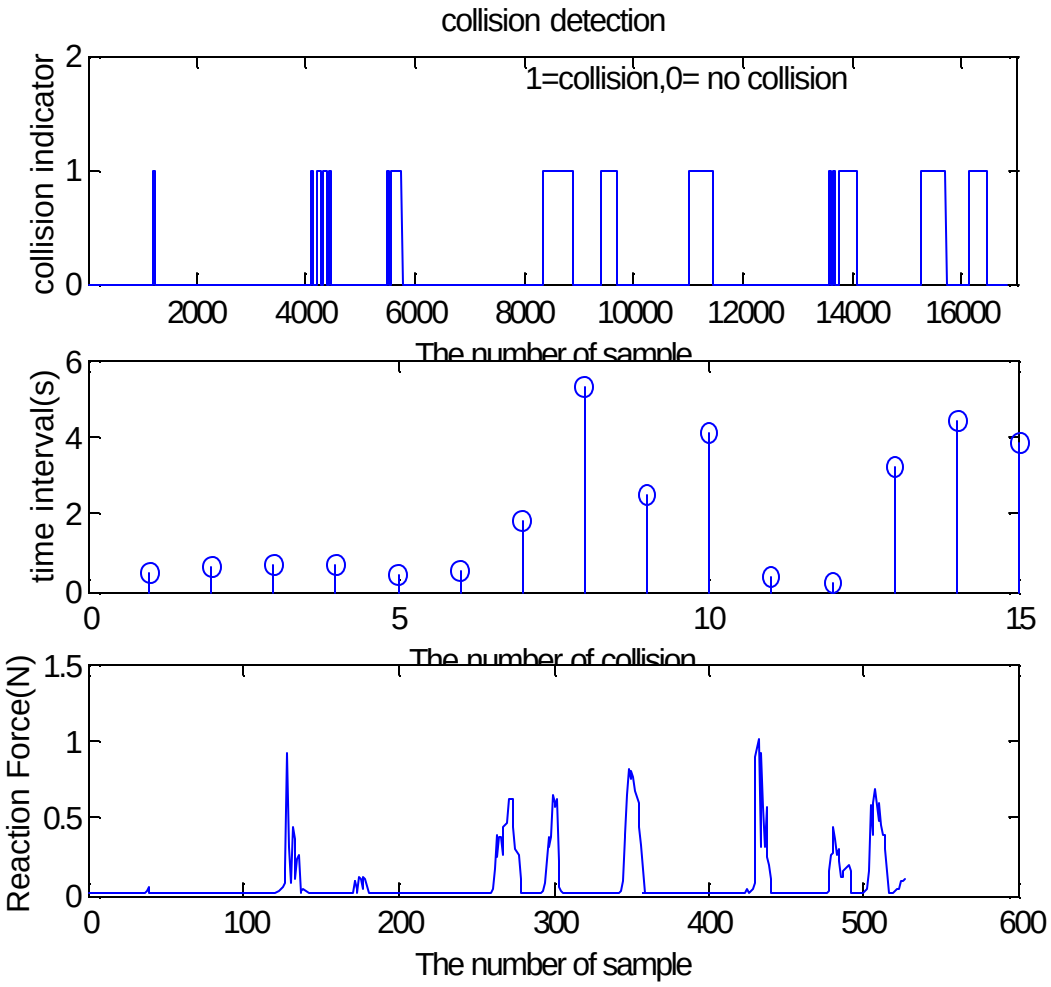


Figure 5.10 Collisions: 15 Collisions Occurred. The max time duration is 5.87s and the minimum is 0.14s. The max impact force is 1.01N and the minimum is 0.15N

5.4.2. Motion Performance after Therapy Training

After therapy training, the data for each subject was collected. The analysis for subject 1 is presented in Figures 5.11, 5.12, and 5.13.

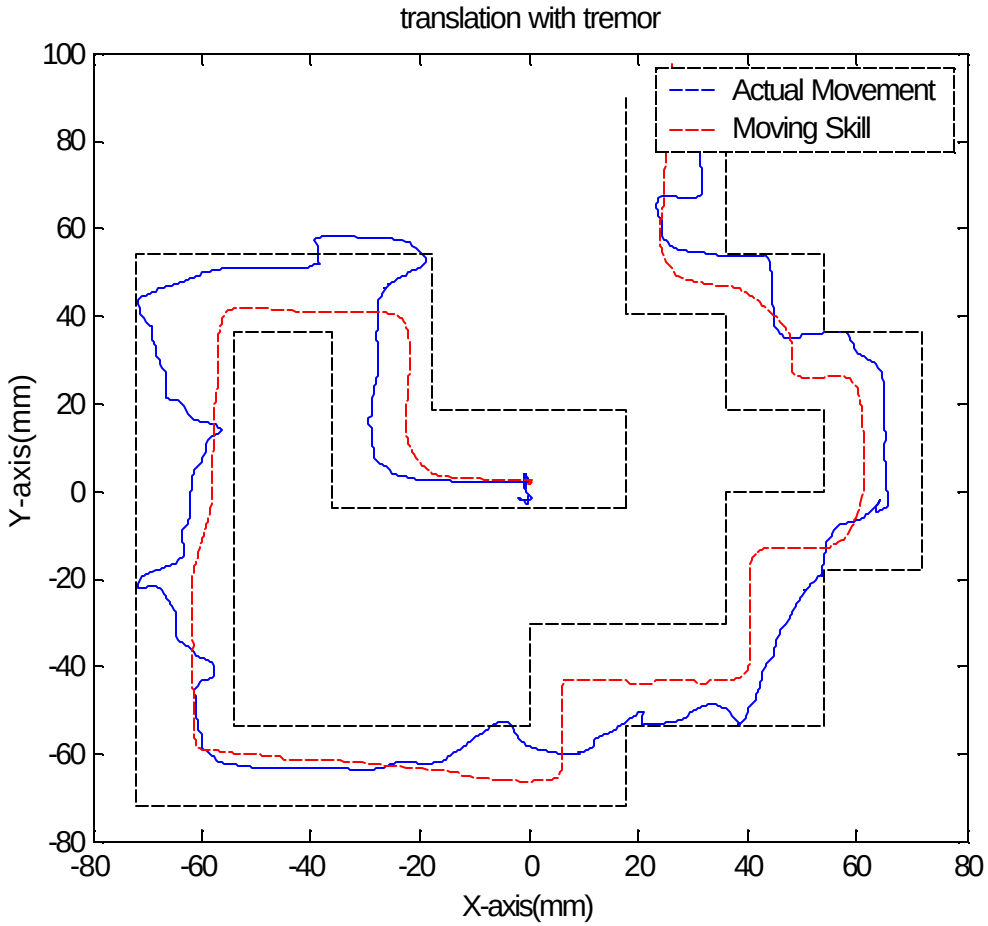


Figure 5.11 Trajectories after Therapy Training

Actual Moving Distance is 619.3 and Distance Ratio is 1.25

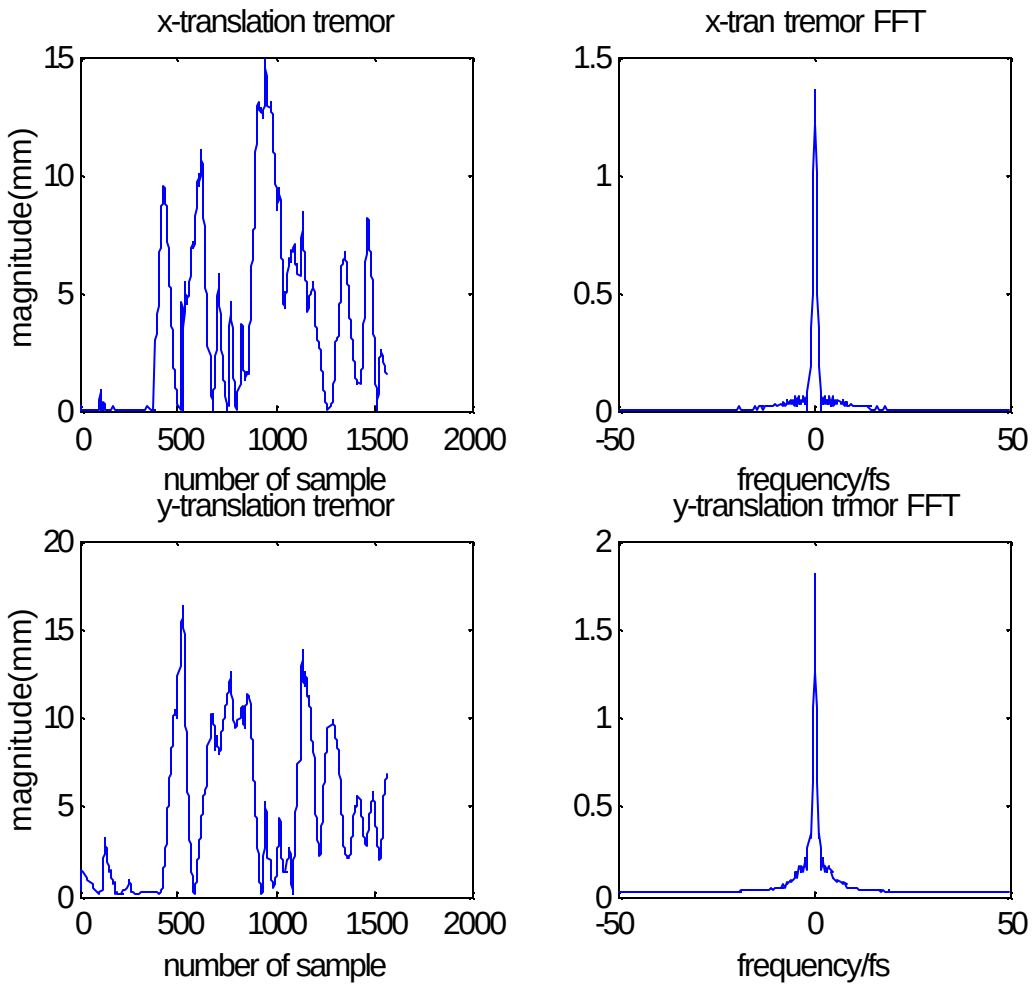


Figure 5.12 Translation Tremors After Therapy. X-axis Tremor Magnitude Mean Is 5.27 mm and STD Is 3.93. Y-axis Tremor Magnitude Mean Is 6.71 and STD Is 4.41

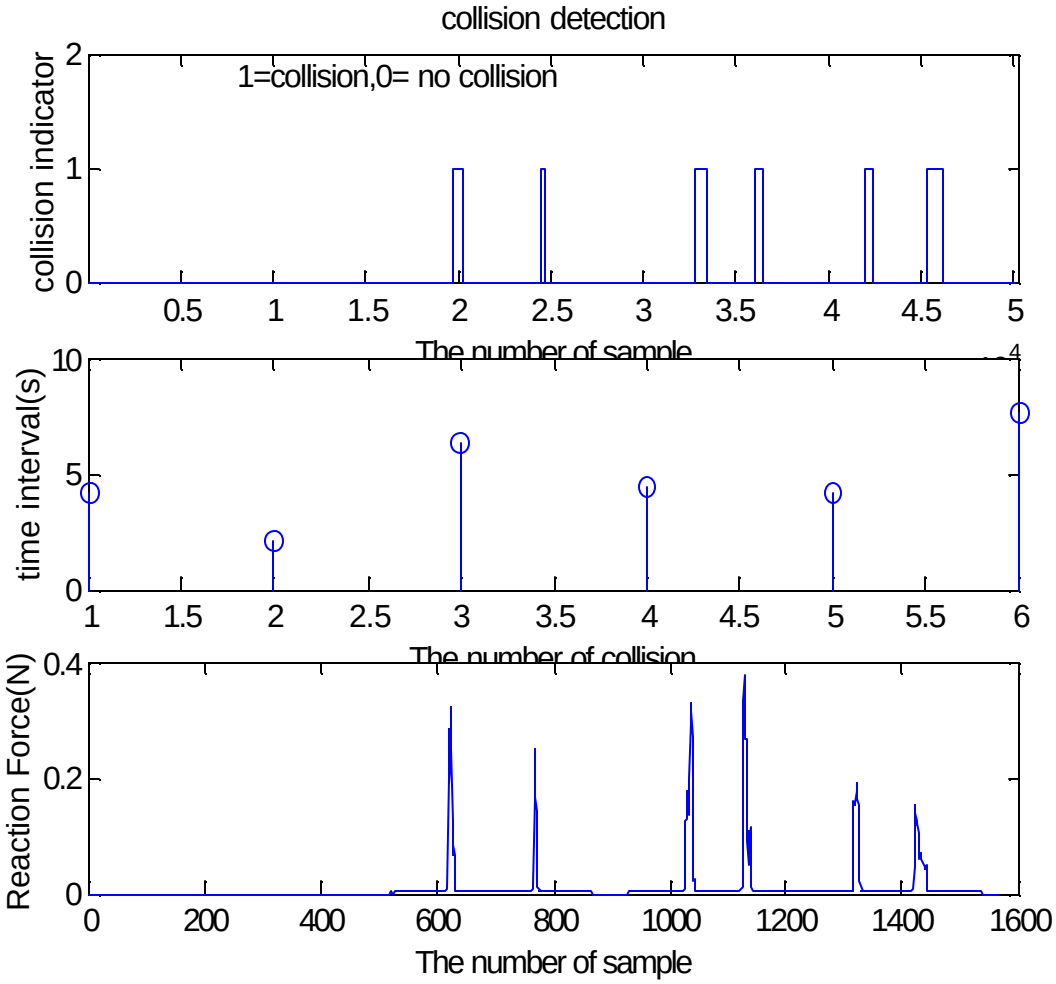


Figure 5.13 Collisions After Therapy. 6 Collisions Occurred. The maximum impact force is 0.39N and the minimum is 0.18N

The *X-Y* plane trajectory presents movement quality. The smoothness reflects the capability of controlling the end-effector during movement. The tremor information plots present tremor magnitude, without considering its direction since magnitude is more meaningful than direction. The tremor frequency was always low, 2-3 Hz for the two subjects. Impact force occurs when there is a collision. Generally, the impact force is related to the smoothness of the trajectory. The smoother the trajectory is, the smaller the tremor magnitude is. The time duration of each collision indicates the reaction to collision. From figures 5.8 and 5.11, it can be seen that the trajectory was improved significantly. Figures 5.10 and 5.13 show the collision information before and after the therapy training, respectively. As we can see, the numbers of collisions, the collision durations and the impact forces were decreased. Though the tremor magnitude was reduced considerably, the tremor frequency was about the same. This is due to the fact that tremor frequency is not observable to the user. Before and after therapy training, seven trials of execution data were collected for each subject. The performance summary is presented in Table 5.1.

Table 5.1 Movement Performance Summary

	Subject 1(Femal, cerebral palsy with right hemiparesis and spasticity (Mean / Std)		Subject 2(male, 19, cerebral palsy, partial paralysis) (Mean / Std)	
	Before Training	After raining	Before Training	After Training
Length Ratio R	1.68/0.35	1.16/0.27	1.46/0.24	1.12/0.17
Time taken(s)	25.35/3.78	16.99/2.08	18.03/2.80	12.04/1.58
Collision Numbers	17.57/4.70	10.43/3.87	13.42/3.05	8.77/2.49
X tremor Mag-(mm)	10.47/4.86	4.26/2.33	7.77/3.61	5.13/2.03
Y-Tremor Mag-(mm)	10.21/6.72	6.43/2.15	8.42/4.34	5.43/1.93
Tremor freq(max)	3.5Hz/--	3.4Hz/--	2.8Hz/--	2.5Hz/--
MaxTime Duration(s)	4.87/1.59	2.15/0.86	3.04/1.33	1.96/0.65
Impact force(max,N)	1.02/0.53	0.71/0.33	0.89/0.35	0.65/0.20

5.5. Summary

In this chapter a HMM based approach for labyrinth moving skill learning and transferring of the learned skill to persons with disabilities is presented. The two dimensional model is built for the XY plane translation. The learned skill is not the best or the worst one of the numerous task executions, but the one that the operator is most likely to do. That is, the most natural one. The learned skill was used as a virtual therapist for persons with disabilities. Persons with disabilities were asked to follow the "virtual therapist" as closely as possible. The difference between the subject and the "virtual therapist" provides visual feedback which helps the eye-hand coordination control capability. After several times of therapy training, operators could control the end-effector better, and hence reducing collisions and making the trajectory smoother.

Chapter 6: Conclusions and Recommendations

6.1. Dissertation Overview

An intelligent teleoperation system using assistance functions was developed to improve task execution efficiently and to decrease the execution time. The approach was guided by the philosophy that the human operator should remain in the control loop of the slave manipulator, thus using human intelligence for the telerobotics system control. A common rehabilitation evaluation task, "Box and Blocks" was tested using teleoperation assistance functions. The results showed how the desired motion was kept or sometimes augmented and how the unwanted motion was reduced. Complex telemanipulation tasks were decomposed into general and relatively simple subtasks: following a path, aligning with a target, avoiding an obstacle and stopping. Hidden Markov Model was used to classify human motion intention into one of the four classes. For different subtasks, appropriate assistance was applied to enhance the input from master device. Another rehabilitation-robotics application is motion therapy. Using HMM, a labyrinth movement skill was learned by the robot. The learned skill then acted as a virtual therapist and two persons with disabilities on upper limb were trained using this approach. The skill learning based robot therapy and its effectiveness were discussed.

6.2. Virtual Fixture Assistance Based on Motion Intention

In telemanipulation systems, assistance through variable position and velocity mapping or virtual fixture can improve manipulation capability and dexterity. This

assistance is useful not only for path following, but also for aligning with targets and avoiding obstacles. Conventionally, such assistance is based on the environmental information and without knowing the user's motion intention. In this dissertation, user's motion intention is combined with real-time environmental information to apply appropriate assistance. If the current task requires following a path, a hard virtual fixture orthogonal to the path is applied. Similarly, if the task is to position a target, an attractive force field is produced to provide a guide for approaching.

Hidden Markov Model is effective for motion classification. As long as the training set is sufficiently large, the motion recognition accuracy is close to 100%. The assistance is appropriately selected based on the recognized motion. The experimental results without assistance showed that the operator always had random errors that resulted in difficulty in following a path and positioning a target. The experimental results with assistance showed that all those undesired random errors were removed or reduced. The HMM based assistance is useful for improving performance accuracy and decreasing execution time. In order to improve the recognition accuracy, the Hidden Markov Model can be expanded. As long as they are independent, the added dimensions linearly increase the computation complexity.

6.3. Robot Therapy and its Effectiveness

A HMM based approach for labyrinth moving skill learning and transferring the learned skill to persons with disabilities on their upper limb was presented. The multidimensional model is built for the learning *X*-*Y* plane translation skill. The learned skill was used as a therapist for persons with disabilities. They need to follow the "virtual

therapist" as close as possible. The difference between the subject and the "virtual therapist" provides visual feedback that helps the eye-hand coordination control capability. During the training process, the trajectory smoothness did not improve significantly even though the user had less collisions and shorter execution time. This could be due to the fact that operators tend to quickly withdrawn the ball after the collision to follow the continuously updated trajectory. After many repetitions of therapy, operators were able to control the end effector to avoid collisions and make the trajectory smooth. They displayed some movements to avoid unnecessary body arrangements and postured themselves accordingly. The purpose of therapy is to restore some of the lost functions of persons with disabilities. This robot-aided therapy emphasizes the movement control through eye-hand coordination training learned from normal subject's performance. This compensation allows persons with disabilities to improve upper limb coordination; tremor reduction and motion control capabilities.

6.4. General Discussion

Overall, when applying teleoperation assistance, the performance of subjects with disabilities can be enhanced. The results of the various experimental results were promising, and indicated that the proposed assistances techniques have real potential in speeding up the execution of a variety of tasks, improving operation accuracy and reducing operator's fatigue. The Hidden Markov Model based skill learning proposed a new approach for motion therapy. While physical therapy directed by a therapist restores the lost motion through physical exercise, robot therapy supervised by a 'virtual therapist" improves eye-hand coordination by learning from a demonstrator.

6.5. Recommendations

The assistance algorithms were tested by using simulation platforms. It is recommended to use a robot manipulator to test for a variety of real rehabilitation tasks. These tests could be implemented on the workstation-based teleoperation system, which consists of a PhanToM Premium 1.5 and PUMA or RRC manipulator, both of which will be available in our laboratory. Although teleoperation assistance provides very valuable assistance for complex task execution, autonomous execution for some repetitive tasks requiring accurate fine tuning movement is recommended. For the robot system in our lab, computer vision can be configured to implement visual servoing for target grasping.

References

1. 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.

2. 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.

3. Gelderblom, G.J., Cremers G., and Soede,M. "Review of Rehabilitation Robotics Application", International Journal of Human-Friendly Welfare Robotics System", Vol.4, No.1 2003.

4. Axel Gräser, Christan Martens, " Rehabilitation Robusts Transfer of Development and Research Results to Disabled Users", International Journal of Human-Friendly Welfare Robotics System", Vol.3, No.1 2003.

5. W. Seamone and G.Schmeisser, "Early clinical evaluation of a robot arm/worktable system for spinal-cord-injured persons." J.Rehab. Res. Dev., pp. 38-57, Jan. 1985.

6. H. Roesler, H.J.Kuppers, and E.Schmalenbach, "The medical manipulator and its adapted environment: A system for the rehabilitation of severely handicapped," in IRIA Proc. Int. Conf. Telemanipulators for the Physically Handicapped, pp.73-77, 1978.

7. 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.

8. J. Guittet, H. H. Kwee, N.Quetin, and J.Yclon, "The Spartacus telethesis: Manipulator control and experimentation," in IRIA Proc. Int. Conf. Telemanipulators for the physically Handicapped, pp.79-100.1978.

9. H. H. Kwee, "Spartacus and Manus: Telethesis developments in France and the Netherlands," in Interactive Robotic Aids-One Option for Independent living. World Rehabilitation Fund, 1986, pp.7-17.

10. H. H. Kwee, J.J. Duimel, J.J. Smits, A.A. Tuinhofde Moed, and J.A. van Woerden, " The Manus wheelchair-borne manipulator: System review and first results," in IARP Proc. 2nd Wkshp. Medical and Healthcare Robotics, pp. 385-395,1989.

11. M. Topping, "'Handy 1', a robotic aid to independence for severely disabled people," in Proc. 3rd Cambridge Workshop Rehabilitation Robotics, pp.13-16, 1994.

12. J.L. Dallaway and R.D. Jackson, "RAID a Vocational robotic workstation," in ICORR 92-conference proceedings, 1992.

13. Michael Hillman*, Karen Hagan, Sean Hagan, Jill Jepson and Roger Orpwood, " A Wheelchair Mounted Assistive Robot", in International Conference on Rehabilitation Robotics, Stanford, CA, U.S.A.

14. Philippe Hoppenot and Etienne Colle, "Location and Control of a Rehabilitation Mobile Robot by Close Human-machine Cooperation", in IEEE Transaction of Neural System and Rehabilitation Engineering, Vol .9, No.2, June 2001.

15. Christian Martens, Oleg Ivlev and Axel Gräser. Burgar, "Interactive Controlled Robotic System FRIEND to Assist Disabled People," in 7th International Conference on Rehabilitation Robotics, France, May 2001.

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. G.E.Birch, M. Fengler, R.G.gosine, K. Schroeder, M. Schroeder, M. Schroeder and D.L. Johnson, "An assessment methodology and its application to a robotic vocation assistive device", in Technology and Disability, 5(2):151-166, 1996.

19. S. J. Sheredos, B. Taylor, C. B. Cobb and E. E. Dann, Preliminary evaluation of the helping hand electro-mechanical arm. *Technology and Disability*, 5(2): 229-232, 1996.

20. Kazuhiko Kawamura, Sugato Bagchi, Moenes Iskarous, and Magured Bishay, "Intelligent Robotic Systems in Service of Disabled," IEEE Transactions on Rehabilitation Engineering, VOL. 3, NO.1, March 1995.

21. Jin-Woo Jung, Won-Kyung, Heyoung Lee and Jong-Sung Kim, "A Study on the Enhancement of Manipulation Performance of Wheelchair-Mounted Rehabilitation Service Robot," in International Conference on Rehabilitation Robotics, Stanford, CA.

22. Kelly McClenathan and Tariq Rahman, "Power Augmentation in Rehabilitation Robots", in International Conference on Rehabilitation Robotics, Stanford, CA, U.S.A.

23. Christine Wright-Ott, "The GOBOT: A Transitional Powered Mobility Aid For Young Children With Physical Disabilities", in International Conference on Rehabilitation Robotics, Stanford, CA, U.S.A.

24. N. Didi, M.Mokhtari, A.Roby-Brami, "Preprogrammed Gestures for Robotic Manipulatos: An Alternative Speed up Task Execution Using MANUS", in International Conference on Rehabilitation Robotics, Stanford, CA, U.S.A.

25. Richard M. Mahoney, "The Raptor Wheelchair Robot System", in 7th International Conference on Rehabilitation robotics, France, May 2001.

26. Mike Topping, "Handy 1, A Robotic Aid to Independence for Severely Disabled People", in 7th International Conference on Rehabilitation robotics, France, May 2001.

27. O. Ait Aider, P. Hoppenot, E.Colle, " Localization by Camera of a Rehabilitation robot", in 7th International Conference on Rehabilitation robotics, France, May 2001.

28. Zeungnam Bien, Won-Kyung Song, Dae-Jin Kim, Jeong-Su Han, "Vision-based Control with Emergency Stop through EMG of the Wheelchair-based Rehabilitation Robotic Arm, KARES II", in 7th International Conference on Rehabilitation robotics, France, May 2001.

29. Steven Edward Everett, "Human-Machine Cooperative Telerobotics Using Uncertain Sensor and Model Data". Ph.D. Dissertation, The University of Tennessee, Knoxville, 1998.

30. P.G. Backes, "Multi-sensor based impedance control for task execution", In Proceedings of the 1992 IEEE International Conference on Robotics and Automation, pages 1245-1250, Nice, France, May, 1992.

31. Schuyler, R. Mahoney. "Job Identification and Analysis for Vocational Robotics Applications". Proceedings RESNA 1995.

32. L. Leifer, "Rehabilitation Robots", Robotics Age, pp 4-15, May/June 1981.

33. T. F. Chan and R. V. Dubey, "Generalized Bilateral Controller for a Teleoperator System with a Six DoF Master and a Seven DoF Slave," Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, California, May 8-13, 1994, pp. 2612-2619.

34. R. V. Dubey, T. F. Chan and S. E. Everett, "Variable Damping Impedance Control of a Bilateral Telerobotic System," IEEE Control System Magazine, February 1997.

35. http://www.appliedresource.com/RTD/Products/Raptor/index.htm.

36. K. Sato, M. Kimura, and A. Abe, "Intelligent Manipulator System with Nonsymmetric and Redundant Master-Slave," Journal of Robotic System, 9(2):281-290,1992.

37. Luc D. Joly and Claude Andriot, "Imposing motion constraints to a force reflecting telerobot through real-time simulation of a virtual mechanism," in Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, May 1994, pp. 357-362.

38. Kazuhiro Kosuge, Koji Takeo, and Toshio Pukuda, "Unified approach for teleoperation of virtual and real environment manipulation based on reference dynamics " in Proc. of the 1995 IEEE International Conference on Robotics and Automation, Nagoya,Japan,May1995.

39. Thomas B. Sheridan. Telerobotics, Automation, and Human Supervisory Control. The MIT Press, London, England, 1992.

40. Gunnar Bolmsjo, Hakan Neveryd, and Hakan Eftring. "Robotics in Rehabilitation". IEEE Transactions on Rehabilitation Engineering, Vol. 3, No. 1, March 1995.

41. Norali Pernalete. 'Development Of A Robotic Haptic Interface To Perform Vocational Tasks By People With Disabilities." Ph.D. Dessertation, Department of Electrical Engineering, University of South Florida, December 2001.

42. Koivo, A.J., Houshangi, N. "Real-time vision feedback for servoing robotic manipulator with self-tuning controller**",** Systems, Man and Cybernetics, IEEE Transactions on, Volume: 21 , Issue: 1 , Jan.-Feb. 1991, Pages:134 – 142.

43. Jia Li; Najmi, A.; Gray, R.M.; "Image classification by a two-dimensional hidden Markov model". Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] , Volume: 48 , Issue: 2 , Feb. 2000 Pages:517 – 533.

44. 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.

45. Steven E. Everett, Rajiv V. Dubey, Y. Isoda, and C. Dumont. "Vision-Based End-Effector Alignment Assistance for Teleoperation". Proceedings of the IEEE International Conference on Robotics and Automation, Detroit, MI.. May 1999, pp. 543-549.

46. Jie Yang, Yangshen Xu, Chiou S. Chen," Hidden Markov Model Approach to Skill learning and Its Application to Telerobotics", IEEE Transactions on Robotics and Automation, Volume: 10 No.5,Oct 1994.

47. B. Hannaford and P. Lee, Hidden Markov Model Analysis of Force/Torque Information in Telemanipulation. The International Journal of Robotics Research, Oct.1991, Vol. 10, No.5, pp.528-539.

48. SensAble Technologies. http://www.sensable.com/products/phantom.htm.

49. Robotics Research Corporation, P.O. Box 206, Amelia, OH 45102. K-Series Robot Arms Users Manual, 1986. Publication UM-100-87.

50. Wentao Yu, Benjamin Fritz, Norali Pernalete, Michael Jurczyk ,Rajiv V. Dubey "Sensors Assisted Telemanipulation for Maximizing Manipulation Capabilities of Persons With Disabilities", Haptics Symposium 2003, Los Angles, CA, 2003.

51. Kazuhiko Kawamura, Sugato Bagchi, Moenes Iskarous, and Magured Bishay, "Intelligent Robotic Systems in Service of Disabled," IEEE Transactions on Rehabilitation Engineering, VOL. 3, NO.1, March 1995.

52. S. Hayati and S. T. Venkatarman, " Design and Implementation of a Robot Control System with Traded and Shared Control Capabilities", In Proceedings of the 1989 IEEE International Conference on Robotics & Automation, pages 1310-1315, Scottsdale, AZ. May 1989.

53. N. Pernalete, Wentao Yu, R. V. Dubey, W.A. Moreno, "Development of an Intelligent Mapping Based Telerobotic Manipulation System To Assist Persons With Disabilities". In Proceedings of the 2002 IEEE International Conference on Robotics & Automation, Washington, DC U.S.A., May 2002.

54. J.Yang, Y.Xu and C.S. Chen, "Hidden markov Model Approach to Skill Learning and its Application to Telerobotics," IEEE Trans. On Robotics and Automation, vol.10, no.5, pp.621-31,1994.

55. A. Bettini, S. Lang, A. Okamura and G. Hager, "Vision Assisted Control for Manipulation Using Virtual Fixtures," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001, pp. 1171-1176.

56. Gregory D. Hager, A Modular System for Robust Positioning Using Feedback from Stereo Vision", IEEE Transactions on Robotics and Automation, vol.13, No.4, August 1997.

57. Young S. Park, Hyosig Kang, Tomas F. Ewing, Eric L. Faulring, J. Edward Colgate, Michael A. Peshkin, " Enhanced Teleoperation for D & D", in the IEEE International Conference on Robotics and Automation, New Orleans, 2004.

58. Jiang Wang, William J. Wilson, " 3D Relative Position And Orientation Estimation Using Kalman Filter For Robot Control", in Proceedings of the 1992 IEEE International Conference on Robotics and Automation , Nice, France-May 1992.

59. Seth Hutchinson, Gregory D. Hager and Peter I. Corke, "A Tutorial on Visual Servo Contro," IEEE Transactions on Robotics and Automation, vol.12, No.5, October 1996.

60. Bernard Espiau, Francois Chaumette, and Patrick Rives , "A new Approach to Visual Servoing in Robotics", IEEE Transactions on Robotics and Automation, vol.8, No.3, June 1992.

61. P. Marayong, A. Bettini and A. Okamura, "Effect of Virtual Ficture Compliance on Human-Machine Cooperative Manipulation", in IEEE/RSJ proceedings, Lausanne, Switzerland, Oct 2002.

62. Jie Yang, Yangshen Xu, Chiou S. Chen, " Human Action Learning via Hidden Markov Model"," IEEE Transactions on Systems,Man, And Cybernetics-Part A: Systems and Humans, Volume: 27 No.1 , January 1997.

63. L. R. Rabiner, "A Tutorial on Hidden Markov Model and Selected Applications in Speech Recognition", Proceedings of the IEEE, Volume: 77 Issue: 2, Feb 1989.

64. Z. Stanisic, S. Payandeh, and E. Jackson. Virtual fixture as an aid for teleoperation., in 9th Canadian Aeronautic and Space Inst. Conference., 1996.

65. Christophe Collewet, FranÇois Chaumette, and Philippe Loisel, "Image-based visual servoing on planar objects of unknown shape", in Proceedings of IEEE International Conference on Robotics and Automation, Seoul, Korea, May 2001.

66. W. Yu, N. Pernalete, R. V. Dubey "Telemanipulation Enhancement through User's Motion Intention Recognition and Virtual Fixture", submitted to IEEE International Conference on Intelligent Robots and Systems 2004.

67. N. Pernalete, W. Yu, R. Dubey. "Augmentation of manipulation Capabilities of Persons with Disabilities Using Scaled Telemanipulation", IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, October 2002.

68. Rosenberg, L.B.; "Virtual fixtures: Perceptual tools for telerobotic manipulation", Virtual Reality Annual International Symposium, IEEE , 18-22 Sep 1993 Page(s): 76 -82.

69. Payandeh, S.; Stanisic, Z. "On application of virtual fixtures as an aid for telemanipulation and training", Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. 10th Symposium.

70. Pepper, R.L.and Kaomea, P.K. "Research Issues in Teleoperator Systems", 28th annual Human Factors Society Meeting, San Antonio, Tx, 1984.

71. Bolmsjo, G.; Neveryd, H.; Eftring, H.; "Robotics in rehabilitation", Rehabilitation Engineering, IEEE Transactions, Volume: 3 Issue: 1, Mar 1995 Page(s): 77 -83.

72 R.M. Gray, "Vector quantization", IEEE ASSP Mag., vol.1,No.2. pp.4-29,1984.

73. N. Turro, O.Khatib,E.Coste-Maniere, "Haptically Augmented Teleoperation", International Conference on Robotics and Automation. Seoul, Korea, May 2001.

74. Roger Y. Tasi, Reimar K. Lenz, " A new Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration", IEEE Transactions on Robotics and Automation, vol.5, No.3, June 1989.

75. Y.Shirai, H. Inoue, "Guiding a robot by visual feedback in assembly tasks", Patter Recognition., vol.5, pp.99-108,1973.

76. G. Puskorius and I. Feldkamp, "Calibration of robot vision," in International Conference on Robotics and Automation. Raleigh, NC 1987.

77. http://www.mvtec.com/halcon/, "The Software Solution for Machine Vision Application".

78. C. Butefisch, H. Hummelsheim, P. Denzler, and K. Mauritz, "Repetitive training of isolated movement improves the outcome of motor rehabilitation of the centrally paretic hand," *Journal of the Neurological Sciences*, vol. 130, pp. 59-68, 1995.

79. Myer Kutz, McGraw-Hill, “Biomedical Engineer’s Handbook”, 2002.

80. Aleš Bardorfer, Marko Munih, Anton Zupan, Alenka Primožic, “Upper Limb Motion Analysis Using Haptic Interface,” IEEE/ASME Transactions on Mechatronics, VOL.6, NO.3, September 2001.

81. Allen, P.K; Timcenko, A.; Yoshimi, B.; Michelman. P; “Trajectory filtering and prediction for automated tracking and grasping of a moving object”, IEEE International Conference on Robotics and Automation, May 12-14, 1992.

82. Hager, G.D.; Grunwald, G.; Hirzinger, G.; “Feature-based visual servoing and its application to telerobotics”, IEEE/RSJ International Conference, Sept. 1994.

83. Robotics Research R2 Controller Manual V1.4, http://www.robotics-research.com.

84. Brian P. DeJong, J. Edward Colgate, Michael A. Peshkin, “Improving Teleoperation : Reducing Mental Rotations and Translations”, IEEE International Conference on Robotics and Automation, April 26-May 1, 2004.

85. B. Volpe, H. Krebs, N. Hogan, L. Edelstein OTR, C. Diels, and M. Aisen, "A novel approach to stroke rehabilitation: robot-aided sensory motor stimulation," *Neurology*, vol. 54, pp. 1938-44, 2000.

86. 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.

87. H. I. Krebs, N. Hogan, B.T. Volpe, M.L.Aisen, L. Edelstein, and C. Diels, “Robot-Aided Neuro-Rehabilitation in Stroke: Three-Year Follow-Up,” in International Conference on Rehabilitation Robotics, Stanford, CA, U.S.A.

88. H. I. Krebs, B.T. Volpe, B.Rohrer, M. Ferraro, S. Fasoli, L. Edelstein, and N. Hogan, "Robot-Aided Neuro-Rehabilitation in Stroke: Interim Results on the Follow-Up of 76 Patients and on Movement Performance Indices," in 7th International Conference on Rehabilitation robotics, France, May 2001.

88. H. I. Krebs, N. Hogan, M. L. Aisen, and B. T. Volpe, "Robot-aided neurorehabilitation," *IEEE Trans. Rehab. Eng.*, vol. 6, pp. 75-87, 1998.

89. D. J. Reinkensmeyer, L. E. Kahn, M. Averbuch, A. N. McKenna-Cole, B. D. Schmit, and W. Z. Rymer, "Understanding and treating arm movement impairment after chronic brain injury: Progress with the ARM Guide," *Journal of Rehabilitation Research and Development*, vol. 37, pp. 653-662, 2000.

90. Leonard E. Kahn, Michele Averbuch, W. Zev Rymer, David J. Reinkensmeyer, “Comparison of Robot-Assisted Reaching in Promoting recovery From Chronic Stroke”, in 7th International Conference on Rehabilitation Robotics, France, May 2001.

91. D. J. Reinkensmeyer, J. P. A. Dewald, and W. Z. Rymer, "Guidance based quantification of arm impairment following brain injury: A pilot study," *IEEE Transactions on Rehabilitation Engineering*, vol. 7, pp. 1-11, 1999.

Appendices

Appendix A: System Testbed and Experiment Design

This chapter presents the system test bed at rehabilitation robotics lab which is used by this project. The hardware and software used in the project will be introduced.

A.1. Introduction

The previously outlined concept was implemented on the hardware and software in this laboratory. This chapter describes the hardware used to test the new assistance strategy and the software we used in the testbed.

A.2. Hardware

During the course of this project, it was necessary to reconfigure the previously constructed telerobotic system used by students at University of Tennessee at Knoxville [29]. The Kraft Master Hand Controller has been replaced by a PHANTOM premium 1.5. The currently used hardware and corresponding schematic are described in this section.

A.2.1. Robotics Research Corporation Manipulator

The Rehabilitation Robotics and Telemanipulation Laboratory in the Mechanical Engineering Department at the University of South Florida uses a seven-degree of freedom robot manipulator from Robotics Research Corporation (RRC), model k-2107, as the remote manipulator. The manipulator has seven revolute joints boasting a redundant joint for obstacle avoidance.

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, such as Figure A.1,

reaches 2.1 meters, about seven feet. Figure A.1 also shows the schematic of the robot manipulator's seven joints including and location of each joint and their respective travel limits. The travel limits are displayed in table A.1. The motions of the seven revolute joints and an end-effector are displayed in figure A.2. Figure A.3 shows a picture of the complete telerobotic system including the actual mounting of the robot manipulator on the horizontal plane that is not reflected in the previous figure.

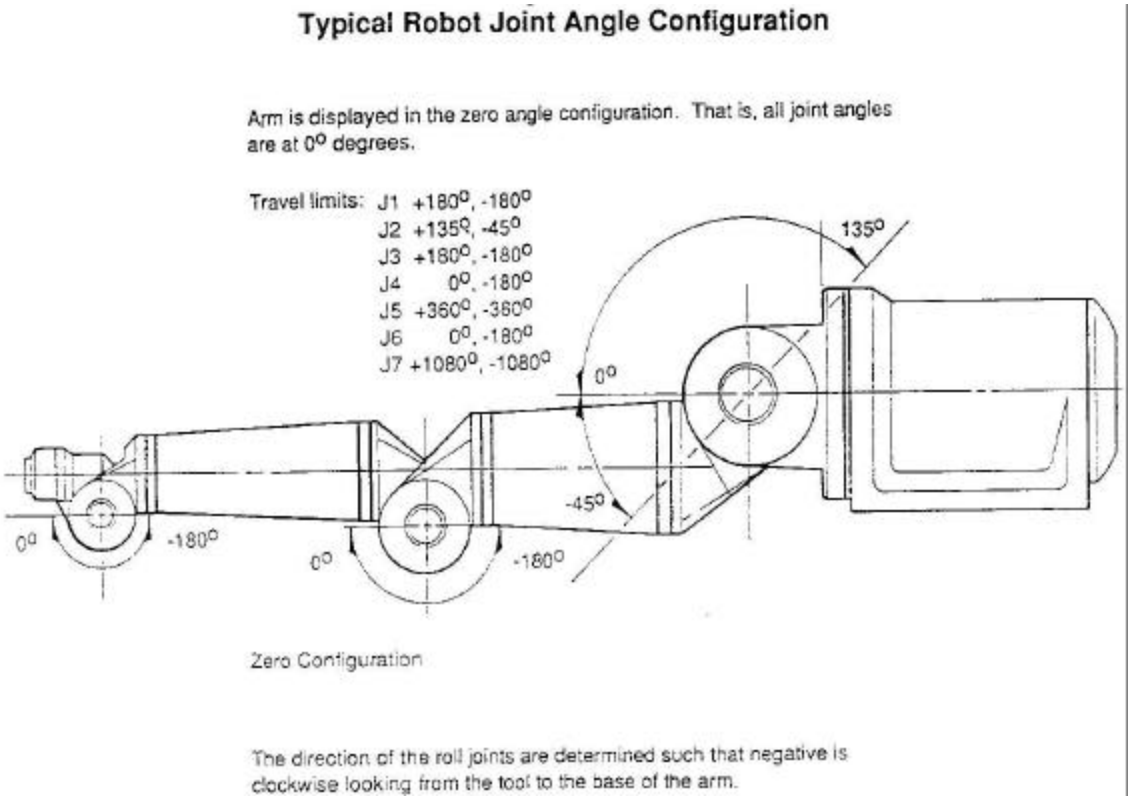


Figure A.1 RRC Manipulator Joints and Limits

Appendix A (Continued)

Table A.1 Joint Limits for the RRC Manipulator

Joint Number	Lower Limit	Upper Limit
1	+180	-180
2	+135	-45
3	+180	-180
4	0	-180
5	+360	-360
6	0	-180
7	+1080	-1080

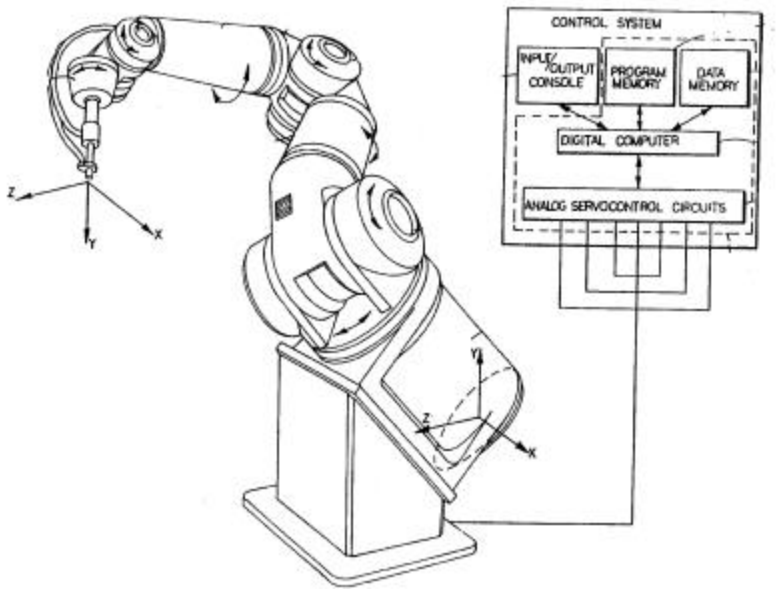


Figure A.2 RRC Manipulator

The manipulator uses a PC based controller. The controller uses inputs from the computer's graphical user interface (GUI) or the teach pendent as the reference position for each of the seven joints. From these positions, the inverse kinematics is 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.

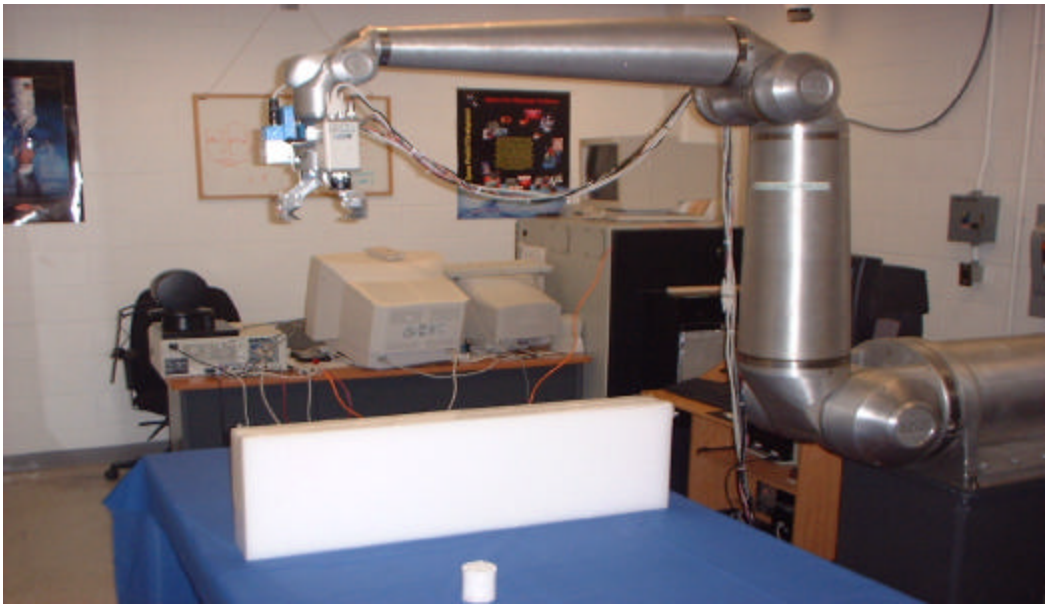


Figure A.3 RRC Manipulator with Sensors and End-Effector

A.2.2. PHANTOM Premium 1.5

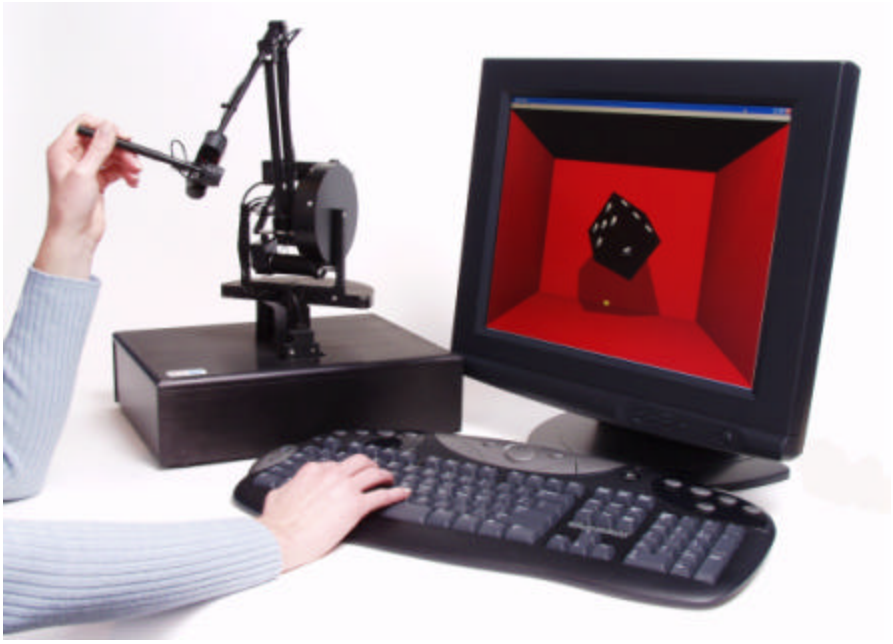


Figure A.4 PHANTOM Premium 1.5

Appendix A (Continued)

Developed by SensAble Technologies [48], the PHANTOM device represents a resolution in human computer interface technology. Prior to its invention, computer users only had the capability to interact through the sense of sight, and more recently, sound. The sense of touch, the most important sense in many tasks, has been conspicuously absent. The PHANTOM device changes all of this. Just as the monitor enables users to see computer-generated images, and audio speakers allow them to hear synthesized sounds, the PHANTOM device makes it possible for users to touch and manipulate virtual objects. The PHANTOM haptic interface is distinguished from other touch interfaces by what it is not. It is not a bulky exoskeleton device, a buzzing tactile stimulator nor a vibrating joystick. PHANTOM application areas include medical and surgical simulation, geophysics and nanomanipulation. The device used in this project is a premium 1.5, whose spec is as follows:

Table A.2 PHANTOM Premium 1.5 Specifications

Workspace	7.5 x 10.5 x 15 inches/19.5 x 27 x 37.5 cm
Range of motion	Lower arm movement pivoting at elbow
Nominal position resolution	860 dpi / 0.03 mm
Back drive friction	0.15 oz / 0.04 N
Maximum Exertable Force	1.9 lbf / 8.5 N
Continuous Exertable Force	0.3 lbf/ 1.4 N
Stiffness	20 lbs./in / 3.5 N/mm
Inertia	< 0.17 lbm < 75 g
Footprint	10 x 13 inches / 25 x 33 cm
Force feedback	x, y, z(3DOF)
Position sensing	x, y, z translation and rotation (6DOF optional)
Interface	Via Parallel Port
Supported platforms	Intel-based PCs

Appendix A (Continued)

A.3. Software

Several independently running programs on various computers make up the software which acts to simulate telemanipulation and control this telerobotics system. The code includes that supplied by RRC manipulator manufacturer, purchased as general purpose software, and written in the lab.

A.3.1. R2 Controller Program

The R2 controller is developed on the basis of real-time motion controller, supporting virtually any robotic mechanism with minimum software changes. It is completely configurable through the use of text configuration files with respect to manipulator and control hardware [83]. The R2 controller provides a server-client TCP/IP protocol interface, which indirectly utilizes the Dynamic Host Configuration Protocol (DHCP) service and the Windows Internet Name Service (WINS) for dynamic mapping of network names and address. A third party application can interface to the R2 server and the R2 real-time controller via the R2 Server API server-client protocol. All the motion controller commands are supported in the R2 Server so that the manipulator can be directed from either a client remotely via an Ethernet communication or an inter-process communication protocol. This API decouples the higher-level control development from the lower level motion controller.

A.3.2. HALCON Computer Vision Software

HALCON is commercial software for machine vision application, which has flexible architecture for rapid development of image analysis and machine vision

Appendix A (Continued)

applications. HALCON provides a library of more than 1100 image processing operators with outstanding performance for blob analysis, morphology, pattern matching, metrology, 3D calibration, and binocular stereo, to name just a few [77]. For example, if we need to get image edge, we can choose “Sobel”, or “Canny” edge detector to do that. Also Halcon supports most of the currently used frame grabbers. We can just call “open_framegrabber” and “grab_image” functions to get real-time image. Components in Halcon are independent objects in the C++ object and VB modules which can be used by users for application development. The image acquisition and processing program can be developed in the integrated development environment (shown in Figure A.5). But usually, in order to implement some complex computation, the program edited in Halcon operators is converted into C++ or VB in which user’s algorithm can be done easily. In this project, the image processing program and the data communication are developed using VC++.

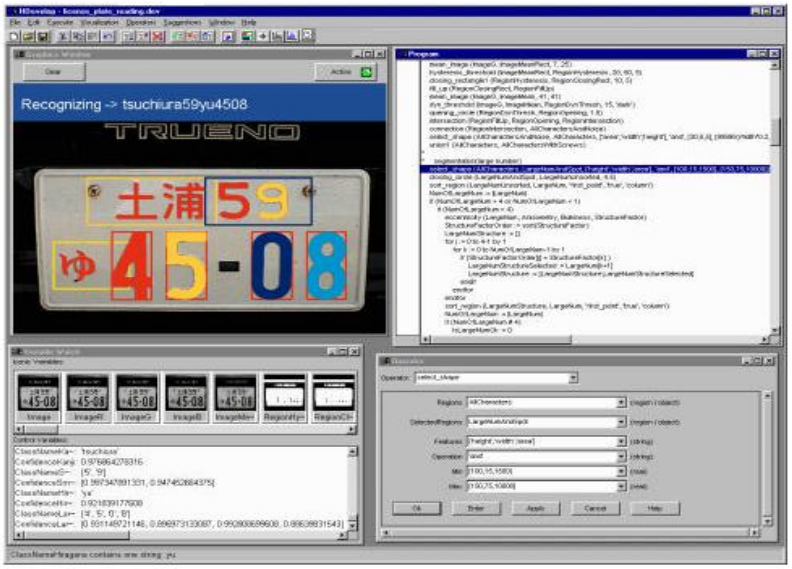


Figure A.5 Integrated Development Environment of Halcon

Appendix A (Continued)

A.3.3.Telerobot Control Interface

This is the main control program to implement telemanipulation system. It is a client of the R2 controller TCP/IP Server-Client architecture via Ethernet communication. It is developed in VC++ to get the Cartesian 3D position and velocity of the master input device, PHANTOM premium 1.5. Two different operation modes are available: one is the position mapping; the other is the velocity mapping, working like a 3D joystick. Also for visual servo controller, this program gets 3D pose of the target and sends the corresponding visual servoing velocity commands to the R2 controller.

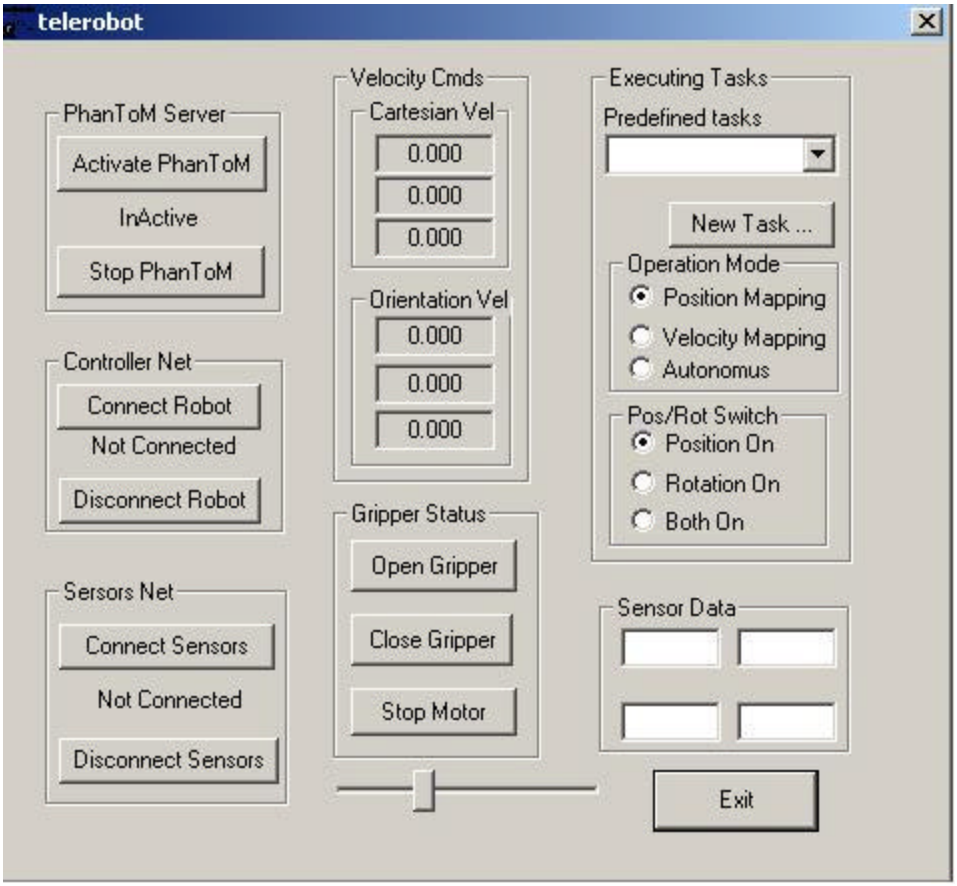


Figure A.6 Telemanipulation Interface

A.3.4. Teleoperation System Architecture

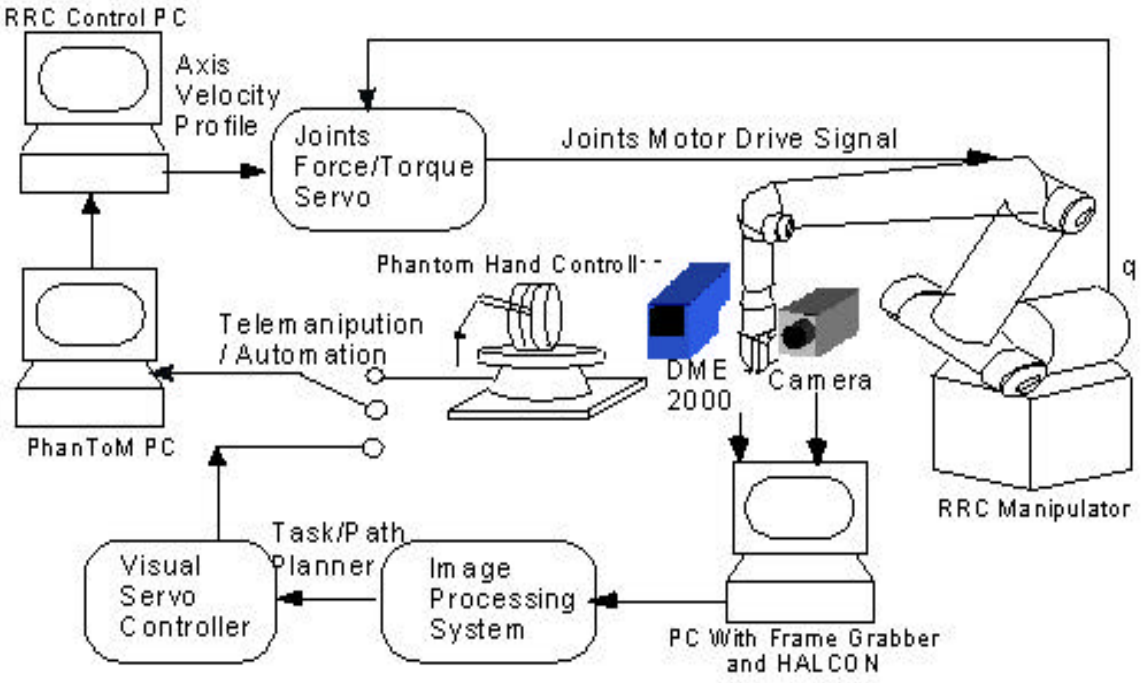


Figure A.7 Teleoperation System Architecture

A.4. RRC GUI

The graphical user interface (GUI) is provided by RRC. The RRC GUI includes jog control, program control, position feedback, client management and file management. This section describes the features of the RRC GUI. Figure A.8 illustrates the different windows, in a custom arrangement. The main window is shown in figure A.9.

Appendix A (Continued)

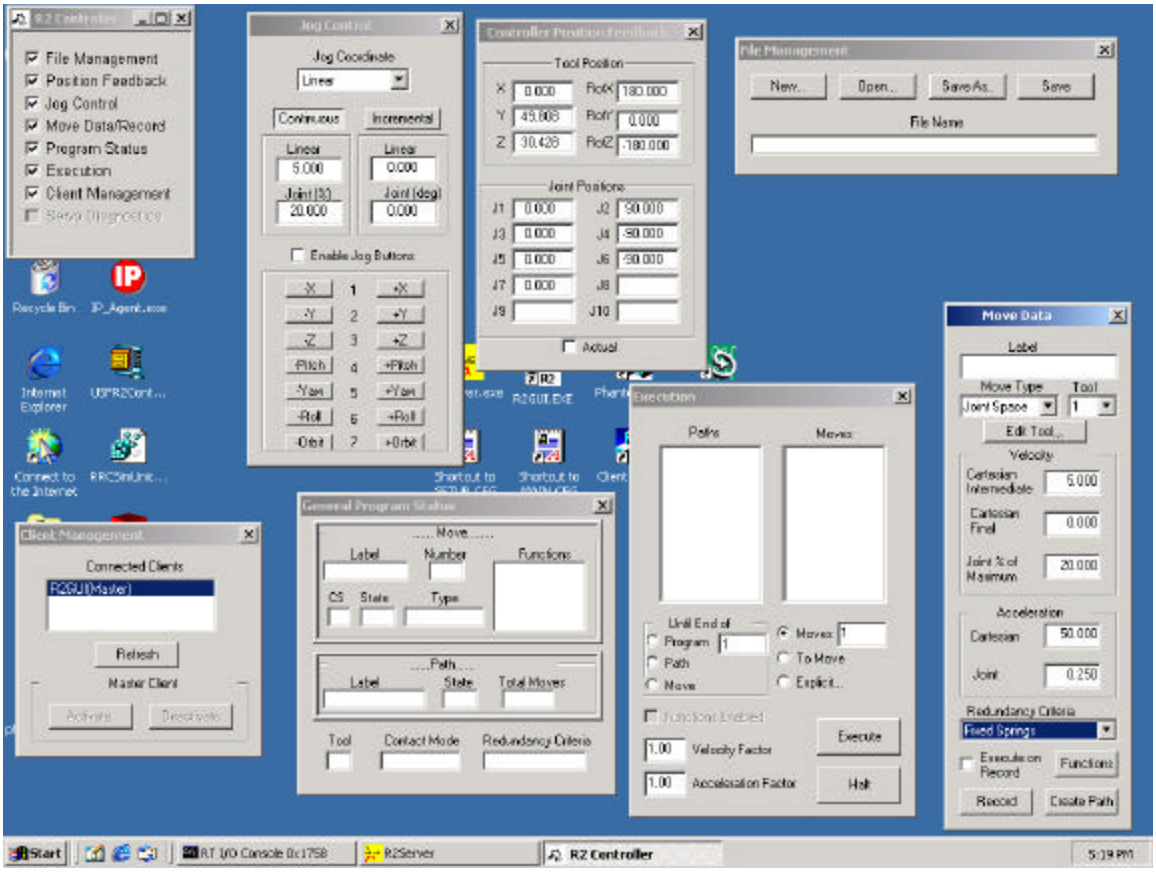


Figure A.8 RRC Graphical User Interface

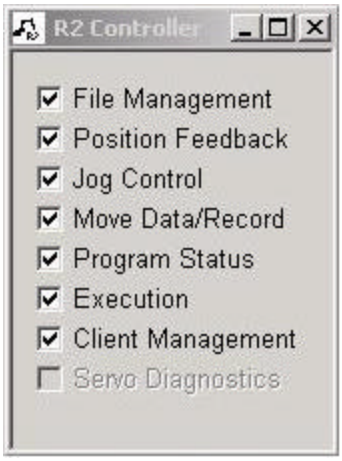


Figure A.9 RRC GUI Main Window

A.4.1. Safe Operating Instructions

As with any machine, a list of guidelines and instructions describes how to safely operate the robot and avoid causing injuries to humans, the robot, or the environment. Upon integrating the many components of the robot controller interface, a list of instructions was developed for the operation of the RRC manipulator. Not only do these instructions provide details for future users, it also points out the many features of the RRC GUI. There are three different modes in which to operate the robot: simulation mode, robot mode, and PHANToM client mode, explained in the flowing sections.

A.4.1.1. Simulation Mode

Instructions were developed for safe operation of the simulation of the telerobotic system. This mode has all the capabilities of the system without sending any commands to the RT Servo Controller. The following is a list of step-by-step instructions to safely operate the robot in simulation.

1. Flip the power switch on the back of the controller box to the "On" position.
2. Press the green controller on button to turn on the controller. (Press controller off to turn off). See figure A.10.
3. To operate the robot in simulation, make sure the main.cfg file has the simulation turned on, do this by the following steps.
4. Open the file to edit: \config\main.cfg (Right click on the icon)
5. On the second line, the simulation statement must read: “Simulation = (On).”
6. When the simulation is turned on, double click on the R2server.exe icon on the desktop. See figure A.11.

Appendix A (Continued)

7. Once the message says "Servo Initialized for Type 2 upgrade," then double click on the R2 GUI.exe icon on the desktop. See figure A.11.

8. Click the position feedback on the R2controller window to see the position of the seven joint angles and the global Cartesian coordinates of the robot.

9. To see visual simulation, double click on Solidworks file on the desktop of the PHANToM computer called: 1207iFA.SLDASM

Figure A.10 Controller Buttons

10. Click on RRC Simulation / Feedback Simulation, and then click connect. The robot should follow the same configuration of the robot position feedback window on the controller computer.

11. There are three different coordinate systems in which to jog (move) the robot: Joint space, hand space, and linear space. Choose linear for most applications.

12. The teach pendant allows for jogging as well. It works in conjunction with the jog control buttons on the screen.

13. To quit, first close all windows on the controller computer, and then terminate the R2.RTA process by clicking on the RT Process Manager (See figure A.11) and clicking local. Find the line with R2.RTA, and click: Kill Process.

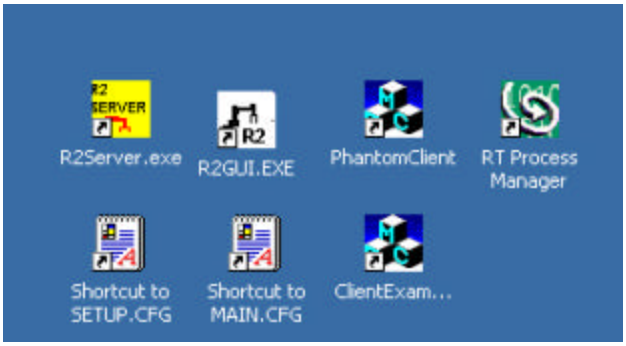


Figure A.11 Desktop Icons on Robot Controller Computer

A.4.1.2. Robot Mode

Instructions were developed for safe operation of the telerobotic system where all commands are sent to the RT Servo Controller. Some instructions are similar, so those steps are not repeated. The necessary instructions are as follows.

1. To operate the robot, turn the simulation off by changing the main.cfg configuration file. The icon is on the desktop, figure A.11.
2. Open the file to edit: D:\config\main.cfg.
3. On the second line, the simulation statement must read: "Simulation = (Off)."
4. Follow the same instructions for when the simulation is turned on.
5. Once the GUI is activated, the Enable Arm window will appear. Click the "Enable Arm" button, and then the computer will count for 20 seconds.
6. Upon being aware of the robot and its location, press the green machine start button, see figure A.10. If this is not done before the computer counts to 20 seconds, the Machine Start button will not activate the robot, and step 5 will need to be repeated. This is incorporated as a safety mechanism. The red e-stop button must be attended whenever the robot is enabled.

7. Now the robot is enabled, and the homing process can begin.
8. The teach pendant will show the seven joints. Move each joint separately to accommodate the joint angles for home position in table A.1. Once a joint has reached its home position, the computer will beep.
9. Once all the seven joints are in the home position, press and hold the red CNL button on the teach pendant until the homing window disappears. The robot will move a little bit to settle in the appropriate home position. Then start using the GUI functionality.

A.4.2. Jog Control

Jog control allows the user to manipulate the robot incrementally. Since the simulation acts as a client to the server, the jog control feature also controls the simulation as well. Jog control, shown in figure A.12, offers three different types of coordinate frames in which to move the robot, linear space, joint space, and hand space.

In linear movement, the user can activate the jog buttons and give commands to move in any axis in the Cartesian coordinate system, X, Y, and Z, and also adjust the orientation, roll, pitch and yaw. The GUI takes the commanded position and orientation in Cartesian coordinates and calculates the inverse kinematics to determine the low level commands to control the joint angles. Since there are six commands corresponding to the six degrees of freedom to define position and orientation, the seventh command is called orbit. The orbit command changes the joint angles of the manipulator while leaving the position and orientation of the end-effector unchanged. The speed in which the jogging

of the robot in linear space can be adjusted to run fast or slow, while the recommendation remains to operate the robot at a safe velocity.

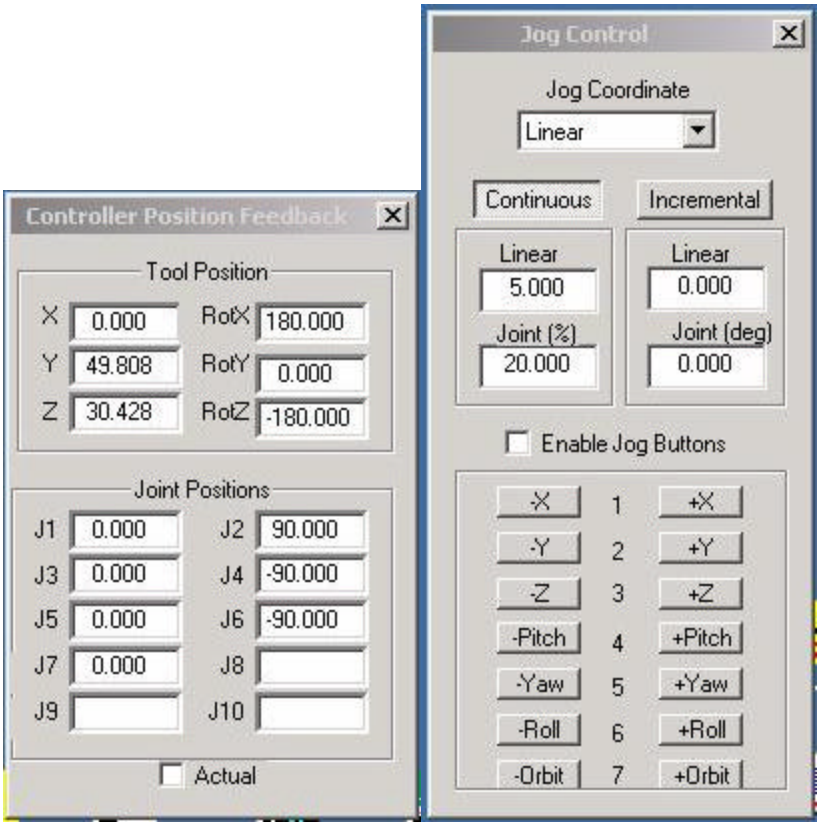


Figure A.12 Jog Control Window and Position Feedback Window

Another coordinate system is called joint space. Each of the seven jog buttons corresponds with its same numbered joint. For example, when the operator presses the +1 button, joint number one will change its angle in the positive direction, according to the velocity set by the user. During the homing operation, the joint space is used to adjust the joints individually to achieve the home position of the robot. This feature is advantageous, especially when the configuration of the robot needs to be adjusted slightly.

The last coordinate system is called hand space. This coordinate system changes with the orientation of the end-effector. The hand X, Y, and Z-axes are fixed on each of the three orientation axes: roll, pitch, and yaw. This is the coordinate system used in teleoperation.

A.4.3. Position Feedback

Position feedback is offered as another window in the GUI environment, shown in figure A.12. This window simply displays the current position of the robot. The values of each of the seven joints are displayed, as well as the corresponding position and orientation in the base coordinate frame. The current Cartesian coordinates are calculated from the manipulator's kinematics, according to its joint angles. These joint angles are received from the feedback of the manipulator. Resolver boards receive the seven joint angles, and send the exact feedback position to be displayed in the feedback window. This information is helpful to the user especially when operating the robot under simulation.

A.4.4. Teach Pendant

The teach pendant, figure A.12, is a hand held control device for operating the robot manipulator. The teach pendant is hooked up to the computer and provides real time control of the robot under the jog control mode. Once "Enable jog buttons" is activated in the RRC GUI jog control window, figure A.12, the teach pendant buttons are activated and coincide with the commands from the GUI on the computer screen. The

teach pendant allows the user to adjust the speed of the robot and change the coordinate system, as well as move the robot. Since the teach pendant operates in conjunction with the jog control buttons, fourteen buttons for the direct operation of the robot, depending on the coordinate system, are present on the hand held teach pendant. The advantage of using the teach pendant over the RRC GUI's jog control is that the operator can be away from the computer observing the robots movements without being obstructed by the computer monitor.

Figure A.13 Teach Pendant for RRC Manipulator

A.4.5. Program Control

Most robot manipulator control programs have the ability to program the robot through a graphical user interface or a teach pendant, to perform a series of movements to predetermined points. This is automating the robots motions. The GUI for the RRC manipulator has this function called, Move Data/Record. The robot can be programmed,

once moving there, to record a point in space. It saves the joint angle configuration corresponding to the appropriate x, y, z, and the rotation in x, y, and z. A series of these recorded points can be programmed and executed to perform a certain automated task.

For example, in the case of teleoperation, the teleoperator would like to change the tool on the end of the robot. This would require the teleoperator to position and align the robot to exchange tools. This process is advantageous to have automated before the teleoperator begins the tasks, so that in the event of a necessary tool change, the operator needs only to select which tool is the desired tool for the next task, and the robot can switch tools at the supervision of the teleoperator, instead of changing tools in teleoperation.

The operator must define the points that determine the automated path. The objective is to use the teach pendant or the jog control of the RRC GUI to move the manipulator to the desired points and record the points by clicking "Record" on the move window, see figure A.14.

From the RRC GUI main window of commands, figure A.9, check the box for move / data record, and a window shown in figure A.14 will appear. Click create path, and the program requests a path name. Recorded points can now be added the path. Click on the execution and the program status check box to reveal the path name and the recorded points, and to monitor the progress of the path execution, see figure A.16.

Appendix A (Continued)

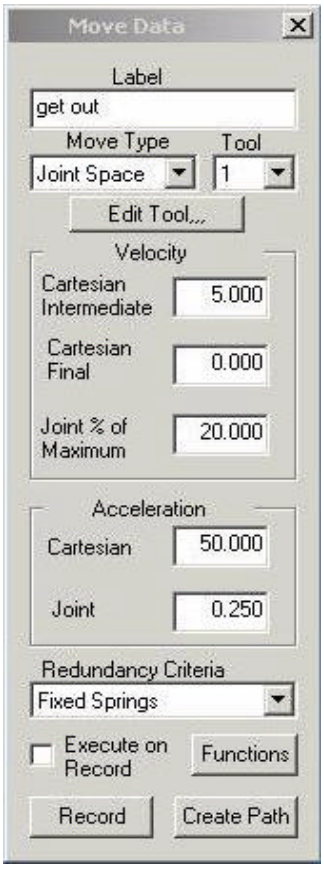


Figure A.14 MainWindow for Move Data / Record

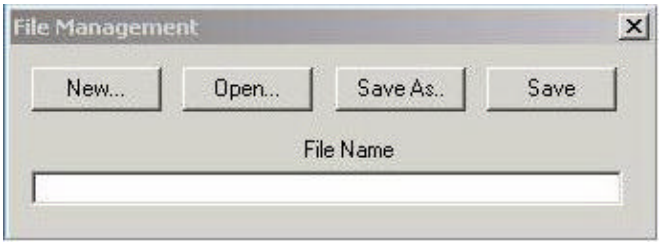


Figure A.15 File Management

Paths or a group of paths can be saved using the file management window, figure A.16. For example, in figure A.16, the path name is called "mountain." This path can be saved in a file and opened again at another time. Through program control, repetitive paths can be automated with a high degree of precision.

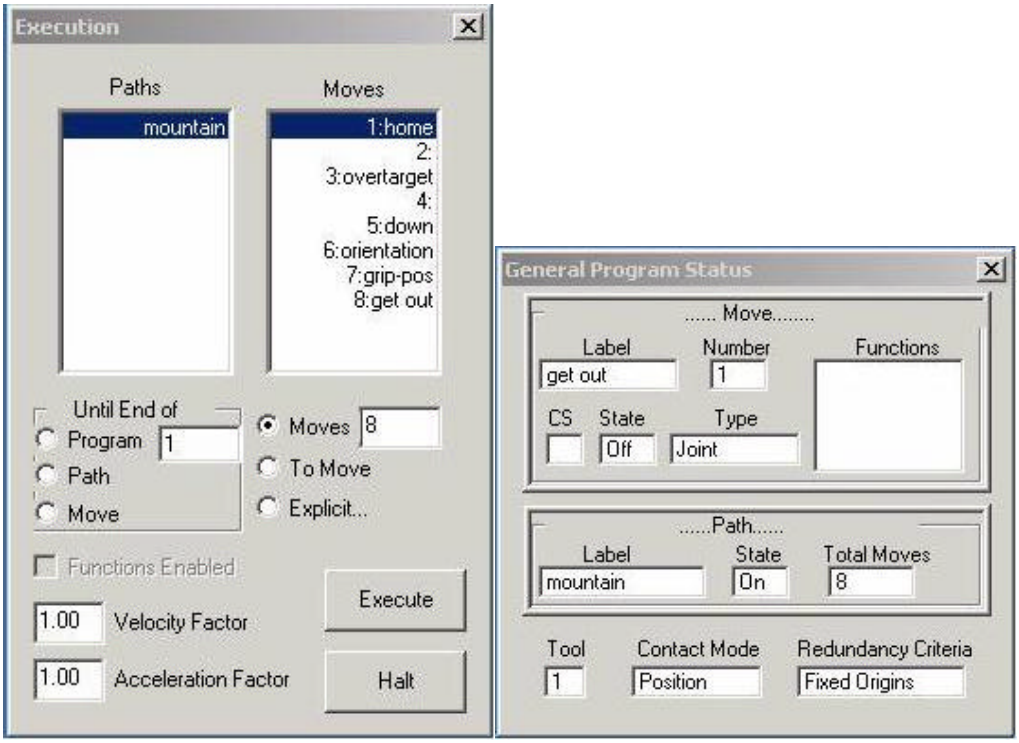


Figure A.16 Execution and Status Windows

A.4.6. Client – Server Interface

In the RRC GUI, the client management window displays the list of connected clients. Clients can be either active or passive. Every client is passive until made active by clicking the activate button in the client management window, figure A.17. Only one client can be active at once. Once a client is activated, that client can send commands to the RT servo controller, and receive position feedback data. The R2 server ignores the commands from a passive client. However a passive client can request feedback data

from the server, and will receive the most recent position feedback data. The active or master client has control over the robot, whether it is in simulation or robot mode.

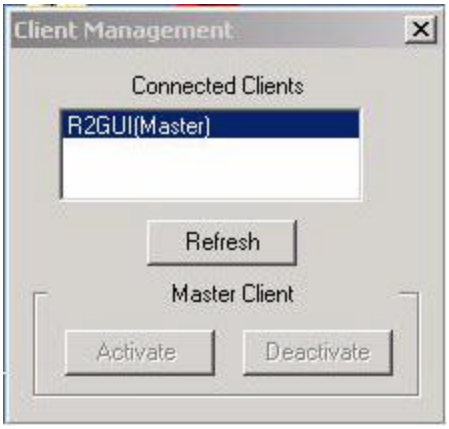


Figure A.17 Client Management Window on Robot Computer

Appendix B: Visual Servoing for Object Grasping

This chapter presents the strategy of enhancing teleoperation through Tele-autonomy. The basic theory and the application of robot vision are also presented.

B.1. Configuration of Vision System

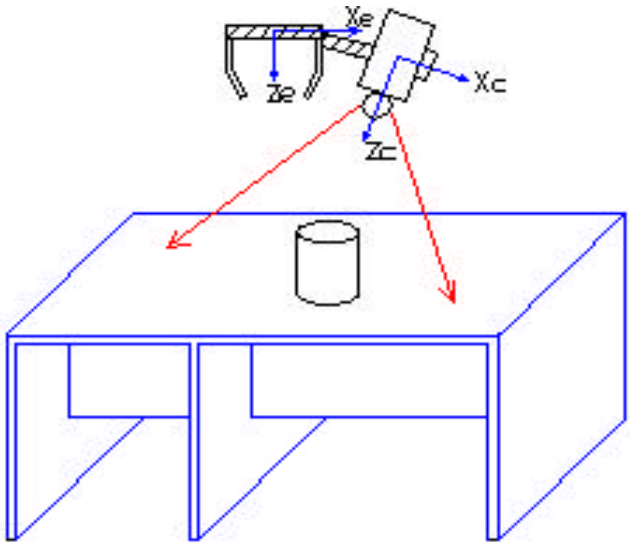


Figure B.1 Configuration of Vision System

In the previous research of this lab, the camera was mounted paralleled with the end-effector coordinate system. In that case, only the translation along Z-axis was taken into account for object pose determination. It was easy to get the relative translation between the two coordinates system by coarse measurement, not doing eye-hand calibration. But the disadvantage of that configuration is that the camera could not see the object when the end-effector is approaching it, thus limiting the usefulness of the

vision system. In this project, in order to improve the flexibility of task execution and keep object in the camera view always, the camera is mounted to the end-effector with some translation and rotation (See Figure B.1). In order for the manipulator to use a camera to estimate the 3D pose of an object relative to the end-effector, calibration of the vision system, including camera calibration and eye-hand calibration are essential.

B.2. 3D Pose Determination of Target with Respect to End-effector

Generally, in order to control robot using information provided by a computer vision system, it is necessary to understand the geometric aspects of the imaging process. Each camera contains a lens that forms 2D projection of the scene on the image plane where the camera is located. This projection causes direct depth information to be lost so that each point on the image plane corresponds to a ray in 3D space. Therefore, some additional information is needed to determine the 3D coordinates corresponding to an image point. This information may come from multiple cameras, multiple views with a single camera, or the knowledge of geometric relationship between several feature points on the target. In this project, the results of the shape-based matching, position coordinates (u, v), orientation θ and scale factor s, enable us to determine the 3D pose with 4 unknowns.

According to perspective projection, a point, ${}^{c}P=[x,y,z]^{T}$, whose coordinates are expressed with respect to the camera coordinate system, *C*, is projected onto the image plane with coordinates $p=[u,v]^{T}$, given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{z}\begin{bmatrix} x \\ y \end{bmatrix} \tag{B.1}$$

Appendix B (Continued)

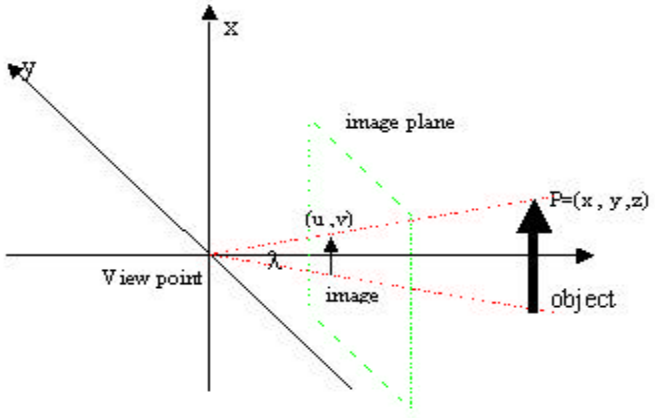


Figure B.2 Coordinate System for Perspective Projection

We assign the camera coordinate system with the x- and y-axis forming a basis for the image plane, the z-axis perpendicular to the image plane (along the optical axis), and with origin located at the distance λ(or f) behind the image plane, where f is the focal length of the camera lens. This is illustrated in Figure B.2.

We assign the tool coordinate system at the origin of the ROI (Region of Interests) of the object. So the coordinate values of the origin O is $O_o = (0.0, 0.0, 0.0)$. Let's assume there is a line segment located between O and $P(m, 0, 0)$ in the tool coordinates.

$$\overrightarrow{OP}_o = (m,0,0)^T \tag{B.2}$$

When creating shape model, it was assumed that the tool coordinate system is aligned with the end-effector except the translation along Z-axis (see Figure B.2). So the coordinate of the tool origin of the ROI is (0, 0, Z_0).

Appendix B (Continued)

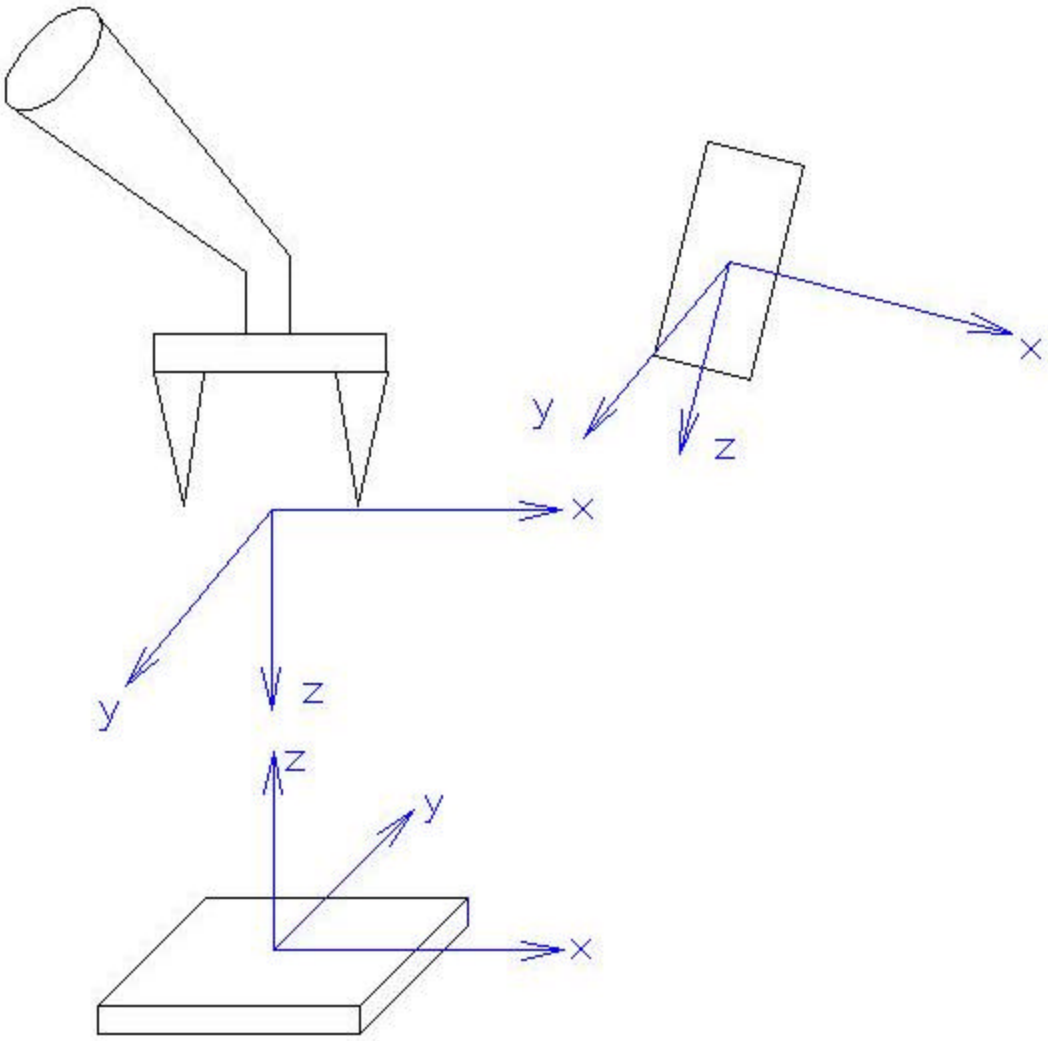


Figure B.3 Coordinates System Assignment for Vision System

When capturing dynamics images, the predefined line segment *OP* is moved to the coordinates as follows with respect to the end-effector system:

$$\overrightarrow{OP}_e = \begin{bmatrix} m\cos a + X \\ m\sin a + Y \\ Z \end{bmatrix} \tag{B.3}$$

Appendix B (Continued)

As we have obtained the eye-hand transformation ${}^{c}H_{e}$, the coordinates of line segment *OP* can be transformed into camera coordinates system as follows:

$$\begin{aligned}\overrightarrow{OP}_c &= {}^{c}R_e * \overrightarrow{OP}_e + {}^{c}T_e \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} * \begin{bmatrix} m\cos a + X \\ m\sin a + Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \\ &= \begin{bmatrix} (r_{11}X + r_{12}Y + r_{13}Z + t_x) + (r_{11}m\cos a + r_{12}m\sin a) \\ (r_{21}X + r_{22}Y + r_{23}Z + t_y) + (r_{21}m\cos a + r_{22}m\sin a) \\ (r_{31}X + r_{32}Y + r_{33}Z + t_z) + (r_{31}m\cos a + r_{32}m\sin a) \end{bmatrix}\end{aligned} \tag{B.4}$$

where $[{}^{c}Re, {}^{c}Te]$ is the eye-hand transformation matrix. In order to clearly express the transformation relationship, we might as well use symbols for the transformation matrix elements, instead of numbers.

In equation (B.4), if we let m=0, α = 0, we can get the coordinates of the tool system origin with respect to the camera system:

$$O_c = \begin{bmatrix} r_{11}X + r_{12}Y + r_{13}Z + t_x \\ r_{21}X + r_{22}Y + r_{23}Z + t_y \\ r_{31}X + r_{32}Y + r_{33}Z + t_z \end{bmatrix} \tag{B.5}$$

The perspective projections of the point O_c and P_c are as follows:

$$\begin{cases} u_{o,1} = f\dfrac{r_{11}X + r_{12}Y + r_{13}Z + t_x}{r_{31}X + r_{32}Y + r_{33}Z + t_z} \\ v_{o,1} = f\dfrac{r_{21}X + r_{22}Y + r_{23}Z + t_y}{r_{31}X + r_{32}Y + r_{33}Z + t_z} \end{cases} \tag{B.6}$$

Appendix B (Continued)

$$\begin{cases} u_{p,1} = f\dfrac{(r_{11}X + r_{12}Y + r_{13}Z + t_x) + (r_{11}m\cos a + r_{12}m\sin a)}{(r_{31}X + r_{32}Y + r_{33}Z + t_z) + (r_{31}m\cos a + r_{32}m\sin a)} \\ v_{p,1} = f\dfrac{(r_{21}X + r_{22}Y + r_{23}Z + t_y) + (r_{21}m\cos a + r_{22}m\sin a)}{(r_{31}X + r_{32}Y + r_{33}Z + t_z) + (r_{31}m\cos a + r_{32}m\sin a)} \end{cases} \quad \text{(B.7)}$$

The perspective projection of line segment *OP* in image plane is also a line segment *op*. In equations (B.6) and (B.7), if we let X=Y=0, Z=Z_0 and $\alpha = 0$, we can obtain the perspective projection of line segment *OP* during shape model creating (Figure B.4):

$$\begin{cases} u_{o,0} = f\dfrac{r_{13}Z_0 + t_x}{r_{33}Z_0 + t_z} \\ v_{o,0} = f\dfrac{r_{23}Z_0 + t_y}{r_{33}Z_0 + t_z} \end{cases} \quad \text{(B.8)}$$

$$\begin{cases} u_{p,0} = f\dfrac{(r_{13}Z_0 + t_x) + r_{11}m}{(r_{33}Z_0 + t_z) + r_{31}m} \\ v_{p,0} = f\dfrac{(r_{23}Z_0 + t_y) + r_{21}m}{(r_{33}Z_0 + t_z) + r_{31}m} \end{cases} \quad \text{(B.9)}$$

Appendix B (Continued)

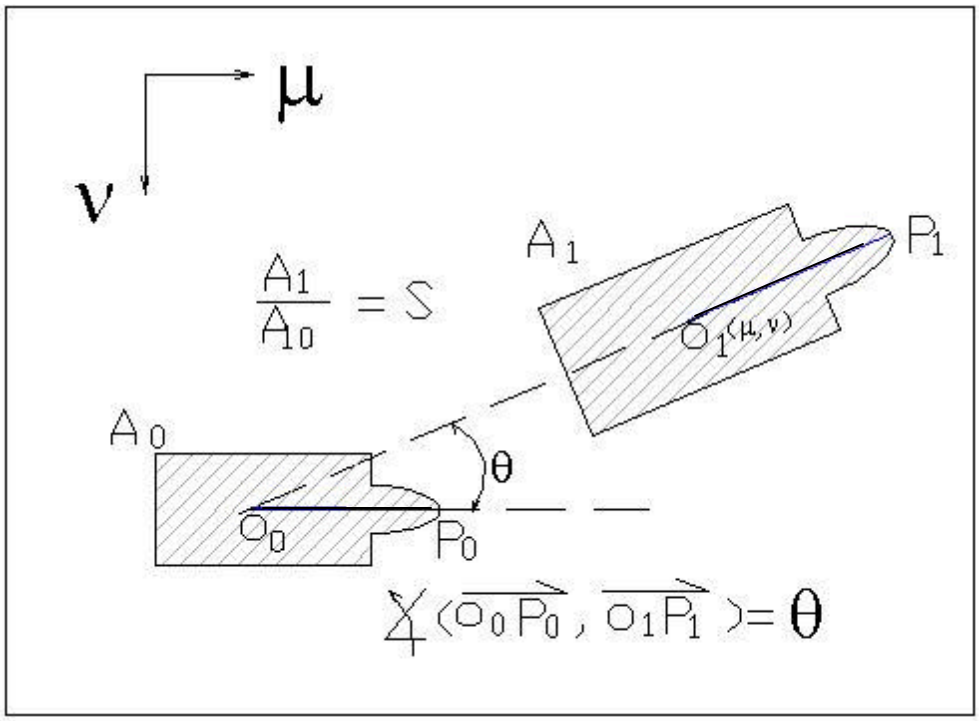


Figure B.4 Perspective Projection of a Line Segment in Image Plane

The projection of line segment *OP* at the initial creating shape model stage and dynamics vision are shown above as $o_0 p_0$ and $o_1 p_1$:

$$\begin{cases} op_0 = (u_{p,0} - u_{0,0},\ v_{p,0} - v_{0,0}) \\ op_1 = (u_{p,1} - u_{0,1},\ v_{p,1} - v_{0,1}) \end{cases} \tag{B.10}$$

Obviously, the orientation of a line segment between the ROI origin and a point on the bounder of the ROI represents the orientation of the model ROI. So orientation parameter θ out of the shape model matching equals to the angle between $o_0 p_0$ and $o_1 p_1$.

Appendix B (Continued)

$$\cos q = \frac{\overrightarrow{op_0} \cdot \overrightarrow{op_1}}{\left|\overrightarrow{op_0}\right| * \left|\overrightarrow{op_1}\right|} \tag{B.11}$$

In order to simplifying computation, we replace some long factors by single symbols.

$$\begin{cases} a_1 = r_{11} X + r_{12} Y + r_{13} Z + t_x \\ a_2 = r_{21} X + r_{22} Y + r_{23} Z + t_y \\ a_3 = r_{31} X + r_{32} Y + r_{33} Z + t_z \\ b_1 = r_{11} m \cos a + r_{12} m \sin a \\ b_2 = r_{21} m \cos a + r_{22} m \sin a \\ b_3 = r_{31} m \cos a + r_{32} m \sin a \end{cases} \tag{B.12}$$

The symbols replacement results in:

$$\cos q = \frac{k_1(a_3b_1 - a_1b_3) + k_2(a_3b_2 - a_2b_3)}{\sqrt{k_1^2 + k_2^2} \cdot \sqrt{(a_3b_1 - a_1b_3)^2 + (a_3b_2 - a_2b_3)^2}} \tag{B.13}$$

where

$$\begin{cases} k_1 = (r_{11} r_{13} - r_{13} r_{31}) Z_0 + (r_{11} t_z - r_{31} t_x) \\ k_2 = (r_{21} r_{33} - r_{23} r_{31}) Z_0 + (r_{21} t_z - r_{31} t_y) \end{cases} \tag{B.14}$$

After submitting b_1, b_2, b_3 into equation (B.13), we can see factor m is canceled out (equation (B.15)), thus proving the orientation is not related to the length of the selected line segment. This makes sense.

$$\cos q = \frac{k_1(k_3 + k_4 \tan a) + k_2(k_5 + k_6 \tan a)}{\sqrt{k_1^2 + k_2^2} \cdot \sqrt{(k_3 + k_4 \tan a)^2 + (k_5 + k_6 \tan a)^2}} \tag{B.15}$$

where

Appendix B (Continued)

$$\begin{cases} k_3 = r_{11}a_3 - r_{31}a_1 \\ k_4 = r_{12}a_3 - r_{32}a_1 \\ k_5 = r_{21}a_3 - r_{31}a_2 \\ k_6 = r_{22}a_3 - r_{32}a_2 \end{cases} \tag{B.16}$$

In equation (B.16), there is only one unknown, that is α. It can be solved straightforward after some algebra operation.

$$a = a\tan 2(-e_2 + \sqrt{e_2^2 - 4e_1e_3}, 2e_1) \tag{B.17}$$

where

$$\begin{cases} e_1 = (k_1^2 + k_2^2)(k_4^2 + k_6^2)\cos^2 q - (k_1k_4 + k_2k_6)^2 \\ e_2 = 2[(k_1^2 + k_2^2)(k_3k_4 + k_5k_6)\cos^2 q - k_1^2k_3k_4 - \\ \qquad k_2^2k_5k_6 - k_1k_2(k_3k_6 + k_4k_5)] \\ e_3 = (k_1^2 + k_2^2)(k_3^2 + k_5^2)\cos^2 q - (k_1k_3 + k_2k_5)^2 \end{cases} \tag{B.18}$$

In is necessary to note that there are two solutions for α from equation (B.15). Based on the simulation results, the solution shown in equation (B.17) is true; the other one is false solution and thrown away.

For each frame of input image, orientation q out of the shape model matching function is known, so the orientation α of the model around the *Z*-axis of the end-effector coordinate system is a function of*q*.

The scale factor *s* out of the shape model-matching algorithm represents the area ratio between the extracted model ROI from the input image and the pre-created model. It is assumed that the area of the model ROI is *A*. While creating shape model, the translation of the tool coordinate system along the Z-axis of the end-effector coordinates

system is $T_{z,0}$. Projecting this object into the plane whose normal is parallel with the optical axis of the camera yields the projected model shape, which has area as:

$$A_{o,0} = A\cos g \qquad \text{B.19)}$$

where γ is the angle between the *Z*-axis of end-effector coordinates system and the *Z*-axis of the camera coordinates system.

According to perspective projection rule, the projection of a polygon in image plane is also a polygon. The area of the model ROI in image plane is:

$$A_{0,i} = A\cos g \frac{f^2}{Z_{c,0}^2} \propto \frac{1}{Z_0^2} \qquad \text{(B.20)}$$

Where Z_0 is the *Z*-axis coordinate of the model ROI in the camera coordinates system at the shape-model creating stage.

For dynamic visions, the *Z* coordinate of the model object is updated. The area of the model ROI in image plane is:

$$A_{1,i} = A\cos g \frac{f^2}{Z_{c,1}^2} \propto \frac{1}{Z_1^2} \qquad \text{(B.21)}$$

From shape model matching,

$$s = \frac{A_{1,i}}{A_{0,i}} \qquad \text{(B.22)}$$

It can be obtained that

$$Z_{c,1} = \sqrt{s}\, Z_{c,0} \qquad \text{(B.23)}$$

From the relationship between the area and the $T_{z,i}$, it can be proven that:

$$Z = \sqrt{s}\, Z_0 \qquad \text{(B.24)}$$

where Z is the Z-axis translation of the tool coordinate system origin with respect to the end-effector coordinate system.

Once we know Z coordinate, we can use the position parameters (u, v) to solve X and Y parameters by substituting equation (B.24) into equation (B.6):

$$\begin{cases} X = \dfrac{-(r_{22}-\frac{v_{o,1}}{f}r_{32})[(r_{13}-\frac{u_{o,1}}{f}r_{33})Z+t_x]+(r_{12}-\frac{u_{o,1}}{f}r_{32})[(r_{23}-\frac{v_{o,1}}{f}r_{33})Z+t_y]}{(r_{11}-\frac{u_{o,1}}{f}r_{31})(r_{22}-\frac{v_{o,1}}{f}r_{32})-(r_{21}-\frac{v_{o,1}}{f}r_{31})(r_{12}-\frac{u_{o,1}}{f}r_{32})} \\ Y = \dfrac{-(r_{11}-\frac{u_{o,1}}{f}r_{31})[(r_{23}-\frac{u_{o,1}}{f}r_{33})Z+t_y]+(r_{21}-\frac{v_{o,1}}{f}r_{31})[(r_{13}-\frac{u_{o,1}}{f}r_{33})Z+t_x]}{(r_{11}-\frac{u_{o,1}}{f}r_{31})(r_{22}-\frac{v_{o,1}}{f}r_{32})-(r_{21}-\frac{v_{o,1}}{f}r_{31})(r_{12}-\frac{u_{o,1}}{f}r_{32})} \end{cases} \quad \text{(B.25)}$$

So far, the four parameters X, Y, Z and α are available for 3D pose. The pose of the object with respect to the end-effector system is:

$${}^{e}P_{o} = \begin{bmatrix} \cos a & -\sin a & 0 & X \\ -\sin a & \cos a & 0 & Y \\ 0.0 & 0.0 & 1 & Z \\ 0.0 & 0.0 & 0.0 & 1 \end{bmatrix} \quad \text{(B.26)}$$

So now we can implement pose-based visual servoing for the system.

B.3.Visual Servo Controller Design

Given an object pose with respect to the end-effector coordinate system, it is straightforward to directly implement target tracking. Let ${}^{e}p_{o}^{*}$ be a desired pose, which is

constant. It is only translated from the origin of the end-effector coordinate system along its Z-axis without any orientation. It also means that the end-effector is aligned with the object and ready for grasping. So in this pose, the only value is the z-axis translation *c*, which is defined 3 inch. ${}^{e}p_{o}^{*}$ is like this:

$$ {}^{e}p_{o}^{*} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & c \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.27} $$

From the pose determination in Chapter 4, the actual pose of the object in respect to the end-effector coordinate system is:

$$ {}^{e}P_{o} = \begin{bmatrix} \cos g & -\sin g & 0 & T_x \\ -\sin g & \cos g & 0 & T_y \\ 0.0 & 0.0 & 1 & T_z \\ 0.0 & 0.0 & 0.0 & 1 \end{bmatrix} \tag{B.28} $$

The pose error is defined as:

$$ P_e = {}^{e}P_{o}^{*} - {}^{e}P_{o} \tag{B.29} $$

Since the orientation is only around Z-axis, we might as well represent the rotation in terms of the unit vector $\hat{z}$ and rotation angle $\hat{q}$, we can define

$$ \Omega = -k_1 \hat{q} * \hat{z} \tag{B.30} $$

$$ T = -k_2 * t_e \tag{B.31} $$

where $\hat{q} = g$, $t_e = [T_x \quad T_y \quad T_z - c]$, k_1 and k_2 are proportional constants.

Appendix B (Continued)

The purpose of the visual servo is to produce velocity commands to drive the robot to a desired pose automatically. As shown in figure B.4, there are two different control modes to drive the manipulator. When the object is not in the scene of the end-effector mounted camera, the telemanipulation operation can transmit control commands through input device. Once the target is seen by the camera and the relative pose between the camera and the object is available, visual servo will take effect to generate control commands. These two control modes can be switched easily.

B.4. Tele-autonomy Design

Our telerobot-operation experience revealed that a typical ADL task is composed of a few motor behaviors (sub-tasks), namely looking_for goal, move_to_goal, align_with_goal, as shown in figure B.5.

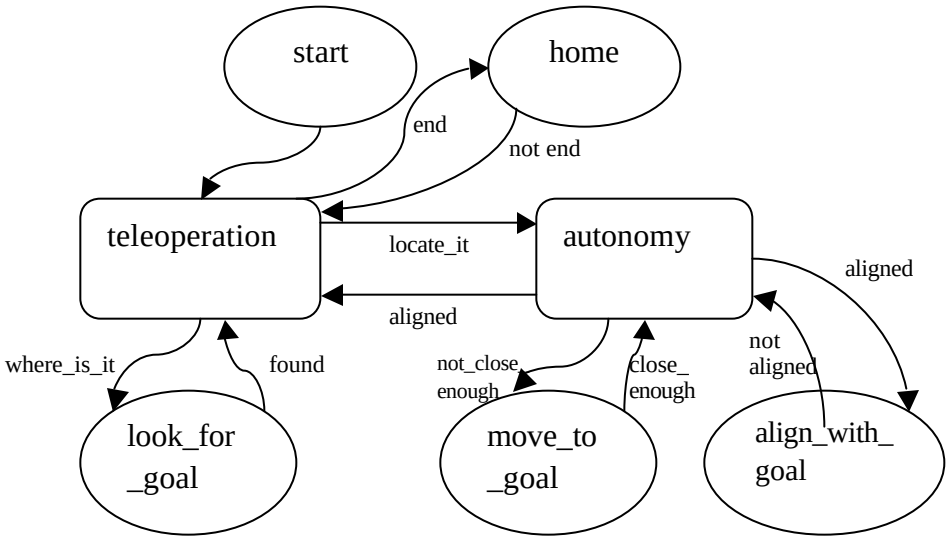


Figure B.5 Tele-autonomy Illustration

About the Author

Wentao Yu was born on February 5, 1972 in Wuhan, China to Dachu Yu and Hanmei Mei. He attended Southern Institute of Metallurgy in Jiangxi, China and graduated with his Bachelor of Science in 1994. Then he was employed by Shougang Group in Beijing, China, working as a mechanical and control engineer for three years. In 1997, he returned to school for graduate study at University of Science & Technology Beijing, where he attained his Master of Science in Mechatronics Engineering with a concentration in embedded control system in early 2000. After graduating with his Master, he had been working in a software company as an embedded software engineer. Some investigation into Ph.D. programs around the world led him to come to University of South Florida, doing rehabilitation robotics research under the direction of professor Rajiv Dubey. He completed this work in the field of intelligent telerobotics with assistance during 2003 ~ 2004. In summer of 2004, he got a senior control engineer job offer from TRW Automotive Inc. and started his new job late in August 2004.