uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Ismail Shakra

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Design and Development of a Virtual Reality Haptic-based Rehabilitation Framework for Post-Stoke Patients

TITRE DE LA THÈSE / TITLE OF THESIS

A. El Saddik

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Edward Lemaire

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Chung-Horng Lung

Wail Gueaieb

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Design and Development of a Virtual Reality Haptic-based Rehabilitation Framework for Post-Stroke Patients

by

Ismail Shakra

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirements for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Ottawa, Ontario, Canada 2006

Your file Votre référence
ISBN: 978-0-494-18467-7
Our file Notre référence
ISBN: 978-0-494-18467-7

Abstract

The recovery of hand functions in post-stroke patients relies on the length of therapy that is available to them. Rehabilitation exercises supervised by occupational therapists are characterized by repetitiveness and a constant increase in intensity. The facilities and time allocated to recovering stroke patients restrict the maximum level of rehabilitation that can be attained. Various efforts have been materialized into rehabilitation themes set in virtual environments and carried out via haptic devices. This thesis carries forward in that direction by implementing virtual reality, haptic-based exercises for the purposes of hand rehabilitation. This building step aspires to catalyze the motion to bring about a haptic-based rehabilitation system that can be set in the patient's own house to provide him/her with treatment that is not restricted by time and facilities and that offers continuous evaluation of the patient's improvement.

This thesis presents a framework that implemented virtual reality exercises carried out with the use of haptic devices with the aim of being used by recovering stroke patients. The exercises were tested with healthy subjects to collect information pertaining to the hand performance; namely about the movement and grip of the hand. The information collected was extracted from data recorded during the exercise, like position of the hand in the virtual space, and angles made by fingers when grasping objects. By analyzing the data carefully, the research effort deduced certain analysis patterns that would provide occupational therapists with a means to continuously evaluate a patient's performance, and hence provide him/her with adaptive recovery courses.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Abdulmotaleb El Saddik for all his support and guidance; this work would not have materialized if it were not for him. I would also like to thank my co-supervisor, Dr. Ed Lemaire for his valuable feedback and suggestions.

I would also like to thank the volunteers from the Multimedia Communication Research Laboratory, School of Information Technology and Engineering, University of Ottawa for giving their time to undergo the experiments.

I am indebted to all the people who have helped me with accomplishing this work; especially my brother Ayham, for his proof reading and suggestions, my friend Samer Samarah for the valuable assistance he has provided with the data analysis, and my undergraduate research colleagues, Alfred Luu and Shaun Spier, for their help with implementation.

Last, but certainly not least, I am thankful for the love and support that has been bestowed on me by my family and friends. They have been a source of strength and comfort throughout my life. Everything I have accomplished would not have been possible without them.

To My Parents

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF GRAPHS

LIST OF CHARTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Objectives and Motivation

For humans, physical tasks involve the use of fine sensory-motor skills, such as picking up a glass of water and putting it in a specific location. In the case of stroke patients undergoing rehabilitation to recover all or some of their lost hand functions, standard exercises administered by occupational therapists and conducted in pre-determined time sessions aim to help patients through rehabilitation. Due to lack of sufficient man power and time, full rehabilitation usually does not materialize for many patients. In practice, most patients require considerably more time to achieve optimum rehabilitation. This is understandable since rehabilitation exercises must be repeated periodically to fulfill their purpose. In addition, the level of difficulty of each exercise must increase with each session, depending on the patient's progress.

The work presented in this thesis aims to lay the foundations of a haptic-based virtual reality type system that circumvents the said hurdles in order to provide a safe and fully functional rehabilitation system that is accessible to a recovering stroke patient at all times of this research is to create a haptic-based virtual reality environment that incorporates rehabilitation exercises aimed to help post-stroke patients to recover vital hand functions. Subsequently, this environment is created and tested with University of

Ottawa students. The performance of the students is evaluated. The framework was able to extract information that serves as a building corner stone for occupational therapists.

The ***objective*** of the thesis is to analyze a subject's data to extract specific information about the concerned subject. When applied to future patients, patient-specific analysis of extracted data would provide information that could assess the occupational therapist to continuously evaluate the patients' performance and therefore devise courses of therapy that will help the patient to a quicker and more efficient recovery.

1.2 The Research Problem

Recovering stroke patients are typically seen for half hour sessions, once or twice a day; hardly enough time for a patient to recover, especially that it is decreased to once or twice a week when seen as an outpatient. The time elapsed from admission to discharge is around 42 days [32]. At the Ottawa General Hospital, an average length of a rehabilitation session for a patient runs for typically an hour. This number is double that of an average session length in a US private healthcare facility. A patient is seen for around 25 times a week. Not only will a haptic-based VR system allow a patient all the required time to recover by providing intensive exercises that are repetitive in nature, which is necessary for recovery [23], but it will do so by providing this service from the comfort of the patient's home and consequently allow for continuous evaluation of performance.

The process of implementing a framework that aims to induce haptic-based post-stroke rehabilitation in virtual settings requires a delicate balance to be stricken between ambition and reality that is dictated by available resources and time restrictions. An ideal situation would see such work through to the shores of realizing a framework that provides a patient with a personalized station at home, where the patient can log in at pre-determined times that were set by the therapist and perform the appropriate rehabilitation exercises, have the recorded data stored then analyzed by an intelligent context-aware mechanism. The data repository for each patient will be updated accordingly and the

results of each session will be sent to the responsible doctor or occupational therapist. That would be a full realization of the ambitions of people involved in haptic-based rehabilitation in a virtual setting. Current resources available now simply do not allow that. In all works concerned with the subject matter, any progress is bound by available resources and equipment. However, for a framework to possess the potential to realize the final objective, it needs to: incorporate simple rehabilitation exercises that provide a patient with the means to train a stroke-afflicted hand; possess a continuous evaluation system that analyzes recorded patient data from an exercise. Most work that has already been accomplished in haptic-based hand rehabilitation in a virtual environment used a variety of haptic devices in virtual environments that implemented different hand exercises. However, previous works have been lacking in meticulous data analysis that carefully 'dissected' a subject or a patient's performance to obtain granular details about the problems faced in handling objects.

A framework that provides data analysis to yield a detailed evaluation about a patient's hand performance would a valuable contribution to post-stroke hand rehabilitation. The resulting evaluation will help a therapist to detect vital signs about the patient's status and provide him/her with in-depth information about minute details about the hand, like the trouble with a certain finger for example.

1.3 Thesis Contributions

This work's contribution to the field of occupational therapy owes its accomplishment to the manner of analyzing collected data and extracting useful information that was otherwise just stagnant in the form of numerous numbers in tables waiting to be processed. Such analysis owes its merits to creative thinking. This notion is the basis of 'intelligence' or 'context-awareness'.

1.4 Resulting Publications

Five papers have been published:

1. I. Shakra, M. Orozco, A. El Saddik, S. Shirmohammadi, and E. Lemaire, "VR-Based Hand Rehabilitation Using a Haptic-Based Framework," Proc. IEEE Instrumentation and Measurement Conference, Sorrento, Italy, April 2006.
2. I. Shakra, M. Orozco, A. El Saddik, S. Shirmohammadi, and E. Lemaire, "Haptic Instrumentation for Physical Rehabilitation of Stroke Patients," Proc. IEEE Workshop on Medical Measurement and Applications, Benevento, Italy, April 2006.
3. M. Orozco, I. Shakra, and A. El Saddik, "Haptic: The New Biometrics-embedded Media to Recognizing and Quantifying Human Pattern," Proc. 13th Annual ACM International Conference on Multimedia (ACM-MM 2005), Singapore, November 2005.
4. J. Zhou, X. Shen, I. Shakra, A. El Saddik, and N. Georganas, "XML-Based Representation of Haptic Information," Proc. 4th IEEE International Workshop on Haptic Virtual Environments and their Applications (HAVE 2005), Ottawa, Canada, October 2005.
5. M. Orozco, I. Shakra, and A. El Saddik, "Adaptive Haptic Framework," Proc. IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, Giardini Naxos, Italy, July 2005.

1.5 Organization of Thesis

In this chapter, the objective and motivation for the current research was presented at first. Then, the virtual reality haptic-based rehabilitation framework was briefly introduced. Finally, the thesis contributions, resulting publications, and thesis organization were listed.

Chapter 2 introduces the background information related to this thesis, such as introduction to haptics and virtual reality. The literature review for the concurrent research efforts has been presented and dissected.

Chapter 3 provides an overview of the proposed framework. The framework's components are presented and its shortcomings are highlighted. Throughout the discussion of the framework's components, the experimental setup and the hardware and software are explained. The exercises implemented are also included in this chapter.

Chapter 4 shows the results that were obtained and the analysis of those results.

Chapter 5 wraps up the thesis. It suggests ideas for future research.

CHAPTER 2

BACKGROUND INFORMATION AND RELATED WORK

There are various efforts to improve applications that feature haptic devices facilitating rehabilitation-related processes. The framework described in this thesis is but one of those efforts and it targets three main issues in the course of its implementation: designing simple hand exercises that are concerned with the rehabilitation of vital hand skills that have been lost or attenuated due to a stroke; collecting data about a hand's position in virtual space: angles drawn by each of the five fingers in the process of bending them to grasp an object, and velocity of the hand across the three plane axes; analyzing the collected data in such a manner to lay the foundations of a future intelligent system that will match data together to infer important conclusions about a patient's case and suggest solutions to the occupational therapist to help him/her in devising an appropriate rehabilitation course that is tailored to the concerned patient.

The work described in this thesis, and for that matter, any work that navigates in the realms of haptic-based and virtual-reality based rehabilitation exudes vibes of high costs, high demand for processing power and qualified personnel that prompt people in the rehabilitation and occupational therapy field to question the need for such technology, and maybe even raise ethical issues about replacing human power with technology.

Beside the rationale behind such efforts that were brought about in chapter 1, there are some points that must be pointed out to try to erase residing doubts.

This effort is complimentary to the work that will continue to be done by occupational therapists. An intelligent system that provides OTs (Occupational Therapists) with behavioural-based analysis of patient's data will help them in devising better solutions and rehabilitation schemes depending on a patient's personal preferences and needs. Also, such a system will provide OTs with enough material to deduce certain patterns displayed by certain groups of patients that might in turn open the door to new hypothesis in hand rehabilitation. Finally, engaging a patient in haptic-based virtual reality exercises will maintain the motivation necessary to complete the repetitive exercises on a daily basis with the same level of enthusiasm and hence produce consistent data that will materialize into highly accurate analysis. This is in addition to the instant feedback that will help OTs in designing therapy that is unique to each patient [17].

2.1 Background Information

Traditional rehabilitation tests that deal with hand functions provide a patient with repetitions of exercises and exhibit the feature of increasing intensity to challenge the patient and improve the hand power. They aim to improve agility of the hand and strength of grip. The most famous test that is used in hand rehabilitation, the Jebsen hand test [14]. This test features a variety of items that can be found around the house, like soup cans, paper clips, pencils, etc... The test involves seven tasks using items like pencils, playing cards, cans, and spoons to perform tasks like writing short sentences, simulate feeding from a can using a spoon, turning cards, and grasping objects with varying weights. Like the Jebsen Hand Test, the Box and Blocks test [15] involves tasks manipulating objects. But in the Box and Blocks test, the objects are cubes of varying sizes. A patient is asked to move blocks from one section to another.

Haptic-based hand rehabilitation deploys virtual reality and haptic technology to accomplish the task of performing rehabilitation exercises, so basic knowledge about virtual reality and haptics will be presented below.

2.1.1 Virtual Reality

Visual stimulus is very important in many tasks of perception. Virtual environments provide an interface to the real world and closer realistic environment, which can be seen as an extension of the current computer imagery technology. This synthetic image system supports a Virtual Reality (VR) application to recreate an essential scenario for rehabilitation activities. VR offers the potential to create systematic human testing, training and treatment environments that allow for the precise control of complex dynamic 3D stimulus presentations, within which sophisticated interaction, behavioural tracking, and performance recording and analysis is possible [17]. McLaughlin et al have developed an immersive motor rehabilitation system with force feedback interaction. Their system is based on a stereoscopic Head Mounted Display (HMD) as mechanism of visual interaction. A Virtual Environment (VE) is presented in order to establish a closer contact between patients and medical staff as main objective [48]. Substantial research that involves Virtual Reality has been carried out in the areas of therapy and rehabilitation [33, 11, 35, and 53].

2.1.2 Haptics

Linguistically, the word haptic is derived from the Greek verb "to touch" or to handle. It now refers to the expanding discipline that is associated with the study of touch through human computer interaction. Haptic technology has developed rapidly and has been the center of attention of many research communities. As a result, concerned vendors have thrived on creating haptic-related products to be deployed commercially, in development, and for research purposes, as well as serving as prototypes. In the pursuit of making haptic technology more applicable in daily devices, several applications have been searched and revised. Many haptic devices have been developed for use in particular

applications; as a consequence, the adaptation to innovative tasks requires further analysis and implementation of new systems to overcome particular limitations. In some cases the application has to be adapted to the haptic device specification. Moreover, if the application is growing in complexity, particular parameters and scenarios will have to be adjusted. Virtual environments are strongly linked with haptic-based applications. Such environments require the visual sensory channel to produce more realistic sensations. To handle the complexity of the graphical representation and its haptic interactions demands a system to be flexible to different formats and different levels of resolution. In addition to the graphic management handling, the complex calculations for the haptic rendering aspect require an efficient and intelligent algorithm able to learn and understand the requirements for the user application.

Research involving haptic devices aspires to bring about the complex sense of touch, force, and hand kinaesthetic in human-computer interaction. Many applications have been developed for a particular objective and by using the facilities provided by the commercial vendor or the prototype developed in a lab. The applications have to be adapted to the limitations of the device and the provided supporting software platform. Developing new frameworks in software engineering relies on reusing existing software components, readapting to new functionalities, and customizing the environment. Via this approach we envisioned a system that can be flexible and adaptive to the current haptic technologies already available. This implies integrating many fields such as object-oriented design, software reuse, pattern recognition and generalization.

Current haptic application development is inspired by a variety of purposes. Creating tools that impart the feeling of touch in a virtual environment is computationally expensive since the scenario is complex in nature. An application layer in a haptic-based system would obviously be on top and would be responsible for providing an automatic framework that converts the simulated virtual world scenarios to sensory channels via a haptic application. This layer is intended to handle a large variety of specialized services. It is integrated with a collection of functional components. Each functional component comprises a library of routines, which performs a specific task in the process. Building a

haptic-based application requires importing different components that incorporate the whole simulation scenario. There would obviously be a visual component that depicts a graphical representation of the virtual scenario, which can be represented by different data structures. Another component is related to the haptic rendering process, which comprises two procedures; contact detection and force feedback calculation. Different collision detection algorithms are already available [37].

2.1.3 Haptic-Based Rehabilitation

Advancements in virtual reality and haptic technology have always ignited interest in applications that can make use of these advancements to simulate real life situations in which users can get a real feel from the interaction of hardware and software. Medicine is a field that has benefited from haptic-based technology in a virtual environment and will still do. Applications in tele-surgery and optics are numerous. A relatively new field which makes use of haptics is rehabilitation. It is only natural that people working with haptics picked up an interest in haptic-based rehabilitation; the physical exercises and strenuous routines involving handling everyday objects that cast their stamp on rehabilitation are a natural pick for haptic-based applications. With the benefits of inducing enthusiasm and interest in a patient to perform exercises of repetitive nature that lead to consistent data capturing that is continuous (instant feedback), haptic-based rehabilitation has thrived under the umbrella of serious research efforts.

2.2 Related Work

The Interdisciplinary Study of Neuroplasticity (brain's ability to recognize itself by forming neural connections throughout life [19]) and Stroke Rehabilitation (ISNSR) research group at the University of Southern California include research teams that work on different aspects of stroke rehabilitation. ISNSR divides the work on stroke rehabilitation into five areas:

- Clinical studies to examine biological aspects that pertain to post stroke symptoms.

- Study based on an animal mode of ischemic (from ischemia: inadequate blood supply to a local area due to blockage of blood vessels to the area [19]) injury to determine the effectiveness of cellular mechanisms in inducing recovery after a stroke.
- Study on the influence of behavioural experience on Neuroplasticity in stroke rehabilitation, again using an animal mode of ischemic injury.
- Computational studies on grasp and reach with the goal to predict neuroplastic events.
- Development of virtual environments that target physical, psychological, and cognitive concerns of rehabilitation.

Haptic-based hand rehabilitation falls under the last area of research. The exercises implemented usually used two or three haptic devices to realize the virtual environment-based set of rehabilitation exercises that involved subjects (sometimes patients) handling virtual objects to perform tasks involving spatial movement, rotating objects, and guiding objects through pre-set paths [17] and [34]. The goal to be accomplished is best described in [17]: "Develop task-specific virtual exercise environments which will trigger and reinforce the compensatory brain mechanisms that facilitate recovery from stroke." The work documented in [17] involved developing exercises to be performed using CyberGrasp [46]. The intended tasks include placing books on shelves, stacking up objects, and simulate a pouring motion from a glass. Similar work done by the same group of researchers, like that in [18], show improvements in using different haptic devices. Haptic devices like the PHANToM [37] and the CyberGrasp [46] were used to perform tasks that involved spatial rotation, fine motor movement [17], and also some vibrating mechanisms [34].

There were research efforts that involved the use of other haptic devices, like the CyberGlove [46] and Rutgers Master II [5], as can be seen in the work done by the collaborative research group from Rutgers University, New Jersey Institute of Technology, Department of Developmental and Rehabilitative Sciences at the University of Medicine and Dentistry of New Jersey. The work documented in [28, 13, 12, 5, 8, 20,

7, and 4] used the mentioned haptic devices to engage patients (usually four subjects) in the chronic state in virtual reality exercises that target four major hand impairments: finger range of motion, finger speed of motion, finger fractionation, and finger strength. The CyberGlove was used to carry out the exercises that test the first three impairments, the Rutgers Master II (RMII) for the finger strength test since this device provides force feedback, while the CyberGlove does not, thus restricting it to those exercises that test positional aspects (range of motion, finger speed of motion, and finger fractionation). There were tests carried out that targeted ankle performance [4] which constitutes a valuable study. Each test began by showing the patient a transparent hand on the screen of the PC workstation with the desired flexing position that the patient should match. The test then commenced with the appropriate exercise.

For range of motion, the test was carried out in two phases; one for the thumb and the other for the rest of the fingers. The patient would bend his fingers to reveal an image on the screen. The higher his/her range of motion was the more of the image that was revealed. For speed of motion, the test depicts a flying butterfly that the patient would scare away by closing the hand fast enough. If the patient did not provide enough speed the butterfly would not be scared away. The fractionation test asks the subject to press a virtual piano key while keeping the other fingers extended. The difference between the active finger flexion and the maximum flexion of the passive fingers was the criteria for performance. Finally, the strength test measures a piston's displacement that the patient could achieve against a constant force applied to the fingers. The VR simulation consisted of a virtual model of the RMII glove attached to a patient-controlled virtual hand.

The work described above is concerned with evaluating improvement in each of the parameters concerned for patients in their chronic state and also by learning about patients' expectations by knowing what they thought of the exercises.

Avizzano et al [2] investigated the influence of a proportional feedback program into a haptic interface in motor learning skills exercises that involved 20 healthy subjects

drawing a shape as precisely as possible. The subjects used desktop developed at Simultaneous Presence, Telepresence, and Virtual Presence (PERCRO) at the Scuola Superiore S'Anna in Pisa, Italy. The work presented by this group bases relies on the assumption that motor learning processes can only be inferred from a person's performance, and is comprised of three stages: a cognitive stage that relies heavily on sensation and perception; an associative stage that involves learning and refining the motor skill; an autonomous stage that sees the motor skill become automatic by virtue of conscious decisions becoming unconscious translations of stimuli into response. The results show that the user's response improved significantly after training using the haptic interface. This research effort is significant in the study of hand rehabilitation as it shows that using haptic technology can enhance hand performance by improving motor skills.

Another research effort exhibited by PERCRO uses haptic feedback in a virtual environment to treat motor dexterity disabilities in the arm and hand. It is documented in Prisco et al [29]. The work presented in the mentioned study makes a point of immersing the user in the virtual environment and use the feedback to analyze the performance of healthy subjects. The actions performed by a subject involved tasks that can be done in the kitchen, like putting a coffee pot on a flame, pressing down a toaster button, and manoeuvring a piece of a glass on a kitchen table without hitting a vase. The work presented in [29] emphasizes that virtual environment technologies are ready to simulate real life applications and can be efficient in rehabilitation schemes.

Connor et al [9] uses a haptic guided errorless learning with an active force feedback joy stick and computer approach to rehabilitation of cognitive deficits following stroke. The work presented in this effort derives its inspiration from the fact that cognitive impairments following a stroke result in poorer rehabilitation outcomes. Errorless learning is based on preventing or minimizing errors from being made during early learning trials and is seen by Connor et al [9] as a proven method of teaching new information to individuals with memory problems and thus is deemed worthy of use in other types of cognitive and perceptual motor deficits. A joystick and electronic gear box unit act as an input device to a desktop computer via an interface device. 12 users with

different degrees of affliction of visual perceptual deficits used the described input device to move between stimuli displayed on screen and make selections by pressing a button on the joystick. Results obtained from each patient were treated separately. The analysis of data focused on response speed and deviation from the straight line path in the trajectory from start location to target on screen. The results of this research effort showed that errorless learning used with a haptic guidance joystick failed does not result in faster and more direct movements to targets for stroke patients. Some patients showed more improvements than others, but overall, it was concluded that a more careful of study of each patient's needs was necessary.

Although the work described above is more concerned with visual deficits rather than stroke-afflicted hand functions, errorless learning is a proven method in teaching that can be applied to rehabilitation. That however must be preceded by an extensive study of a patient's state and the feasibility of extending the methods of errorless learning to the specific case of stroke-afflicted hand functions.

There are many other research efforts that feature haptic-induced virtual environments for purposes of hand rehabilitation, like Popescu et al [28]. This work introduces a virtual-reality based telerehabilitation system with force feedback and makes use of the Rutgers Master II glove and also features a virtual reality exercise library that provides hand exercises to patients to be performed at home.

The works that deal with haptic-based rehabilitation in a virtual environment displayed the use of several haptic devices induced in virtual environments with varying themes of exercises to assist patients in rehabilitation. The works used with healthy subjects gave were investigative in presenting new ideas. Some works have been specific in targeting certain aspects of hand parameters, [28, 13, 12, 5, 8, 20, 7, and 4]. Some showed more adherence than others to traditional hand rehabilitation tests, [17, 18, and 34]. Other works like Connor et al [9] investigated the use of established learning methods like errorless learning in rehabilitation. The numerous tests that have been implemented throughout the works have had varying degrees of success. That mainly depended on the

degree of care given to the nature of subjects being tested and the analysis performed on the gathered data.

This thesis continues with the tradition of using haptic devices to enable subjects to manipulate everyday objects in a virtual environment. The thesis goes on step further than other works by analyzing subjects' data more meticulously and in a manner that could provide occupational therapists with information that will help in assessing patients and in devising more efficient recovery themes. This statement is based on the fact that the analysis presented in this thesis 'dissects' the information from each individual. Such careful study of each individual performance can be thought of as 'tweaking' the assessment process and rendering it more efficient.

CHAPTER 3

FRAMEWORK OVERVIEW

In this chapter, the framework is presented and described systematically. The framework is comprised of four parts: Sensory system component, haptic software simulation component, client application component, and the haptic/behavioural data system component. First the sensory component is described. Then, the haptic software simulation component is discussed. Next the client application (Virtual Reality Based exercises) is introduced, and the rehabilitation exercises are described. Finally, the haptic/behavioural data system component, which is where the data will be analyzed, is studied.

3.1 Framework Description

Dedicated software and hardware are required to set an architectural design, which involves VE and haptic feedback, enabling users to follow rehabilitation activity according to specified standards.

The proposed framework is depicted in Figure 3.1. It is made up of four components: 1) sensory system; 2) haptic software simulation system; 3) client application; and 4) haptic/behavioural data system.

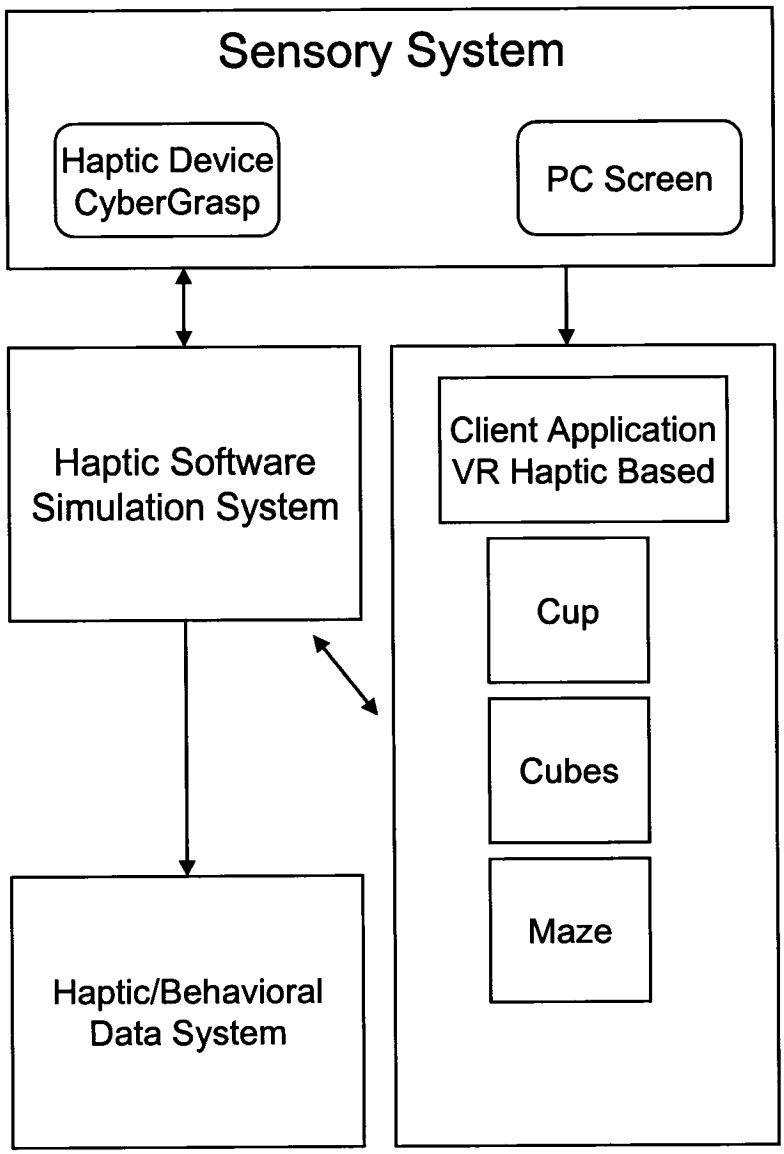


Figure 3.1 Proposed Framework Showing the Four Components

3.1.1 Sensory Component

Sensory items may include a variety of devices depending on the available resources and the need of the specific application. Future expansions in this framework will surely feature new devices that can fit into the sensory system. Similarly, new applications can be added as the need arises and as concerned researchers become more proficient in this haptic-based field.

The sensory component of the system is embedded within the haptic and visual interfaces. Tactile and kinaesthetic stimuli are provided by the CyberGrasp system [45] and [46]

CyberGrasp

The CyberGrasp station (Figures 3.2 to 3.4) is used to conduct the experiments and acquire the necessary data from the subjects.

Figure 3.2 CyberGrasp Station: CyberGlove, CyberGrasp, and CyberForce Armature

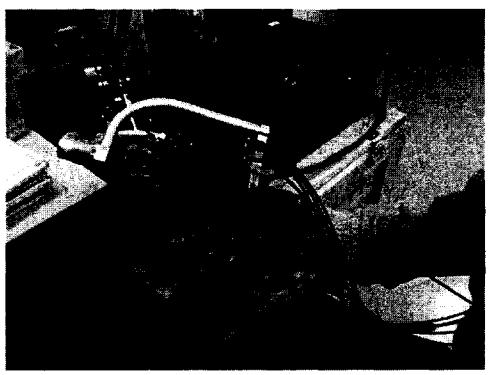


Figure 3.3 CyberGlove fitted with CyberGrasp

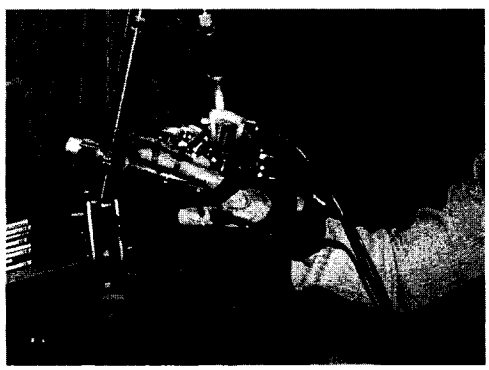


Figure 3.4 View of CyberGlove worn with CyberGrasp

The CyberGrasp station consists of three hardware items: CyberGlove, CyberGrasp, and the CyberForce armature. The CyberGlove collects data that is related to the hand, such as the bending of the joints of the finger. The data collected is used to display a realistic representation of the hand on screen. The device provides up to 22 high-accuracy joint-angle measurements. The CyberGrasp is an exoskeleton capable of generating force feedback on the fingers. It can indulge the user in a feeling of actually grasping an object or tapping it with a finger. The device can generate forces that are roughly perpendicular to the fingertips. The force on each finger can be specified individually. Finally there is the CyberForce armature that has two functions, one of which is tracking of the movement of the hand. It is capable of that because it allows for six-degrees-of-freedom and is also capable of measuring hand rotation and translations. CyberForce also creates the feeling of the weight of an object, thus allowing a patient to feel gravity. The CyberGlove is attached to the CyberForce armature and a CyberGlove is worn in conjunction with the CyberGrasp. Visual interaction is delivered by a conventional computer screen (CRT). Table 3.1, table 3.2, and table 3.3 summarize the technical parameters of the CyberGlove, CyberGrasp, and CyberForce.

Table 3.1 CyberGlove Technical Parameters

Item	**Description**
Sensor Resolution	0.5 Degrees
Sensor Repeatability	1 Degree (typical standard deviation between glove fittings)
Sensor Linearity	0.6% maximum nonlinearity over full joint range.
Sensor Data Rate	150 records/sec (unfiltered); 112 records/sec (filtered). Programmable sample period of polled I/O. (Rates listed are for 18-sensor records at 115.2 kbaud. Higher rates possible with fewer sensors enabled).
Size	One size fits most; 3 oz; 10 ft. glove cable standard (25 ft. cable optional).
Instrumentation Unit	10.00" x 6.25" x 2.75"; 27 oz.

Table 3.2 CyberGrasp Technical Parameters

Item	Description
Maximum Continuous Force	12N per finger
Force resolution	12-bit resolution
Weight (minus CyberGlove)	350 g
Workspace	1 meter radius hemi-sphere
Host Interface	RS-232 and Ethernet are supported

Table 3.3 CyberForce Technical Parameters

Item	Description
Positional Force	8.8N max (6.6N min)
Weight	19lbs. (without CyberGrasp)
Workspace	12"x12" square swept through 133 degrees with radius of 19.5"
Position Resolution	0.0024" max (0.0029 min)
Orientation Resolution	0.09 degrees
Instrumentation Unit	Force-Control Unit (FCU) and power supply included
Interface	Ethernet

The CyberGrasp Station had to be calibrated each time a user logged in. The process of calibrating all the devices takes 3-5 minutes each time.

Force limitations with the CyberForce are due to the fact that there is a limit to the amount of force that the CyberGrasp and the CyberForce can generate. Typically the force that CyberGrasp provides is sufficient for most uses (12N). CyberForce on the other hand might not be able to generate as much force as that of CyberGrasp (8.8N max). CyberForce has a mechanism which will automatically power down the actuators should the stress on the actuators prove to be too great. Another limitation to the CyberForce device is the amount of load/repetition that armature can handle. The steel cables used to generate the force effect cannot handle heavy loads and repetition. The steel cables have been known to snap due to fatigue. Even though there is force on an

object that prevents it from being penetrated, the force is not large enough to prevent a user from deliberately penetrating the object. Furthermore, in order to turn on the actuators on the CyberForce, one needs to keep their foot on the pedal. Once released the user must reapply their foot and press the on button located on the CyberForce unit in order to turn on the actuators again.

As with the CyberGrasp, it produces forces via a network of tendons routed to each fingertip. These tendons are cables, which are connected to actuators; each actuator corresponds to one finger and pulls on the cables to simulate force felt when grasping an object. The limitation is that the device can generate a pulling force and cannot push on the fingers. CyberGrasp can also be somewhat cumbersome, because of all the cables.

3.1.2 Haptic Software Simulation Component

The haptic software component simulates the complex calculations involved in the haptic rendering process loop, maintains synchronization with graphic rendering, and records haptic behavioural data for further analysis. The complex haptic interaction required is handled by the Virtual Hand Software Development (SDK) Toolkit [45] and [46]. This software API offers an object-oriented model with an accompanying C++ library. It includes a device configuration utility and offers real-time collision detection capabilities between 3D representation object and supports full network ability to simulate collaborative environments. Figure 3.5 on the next page is a UML representation of the framework.

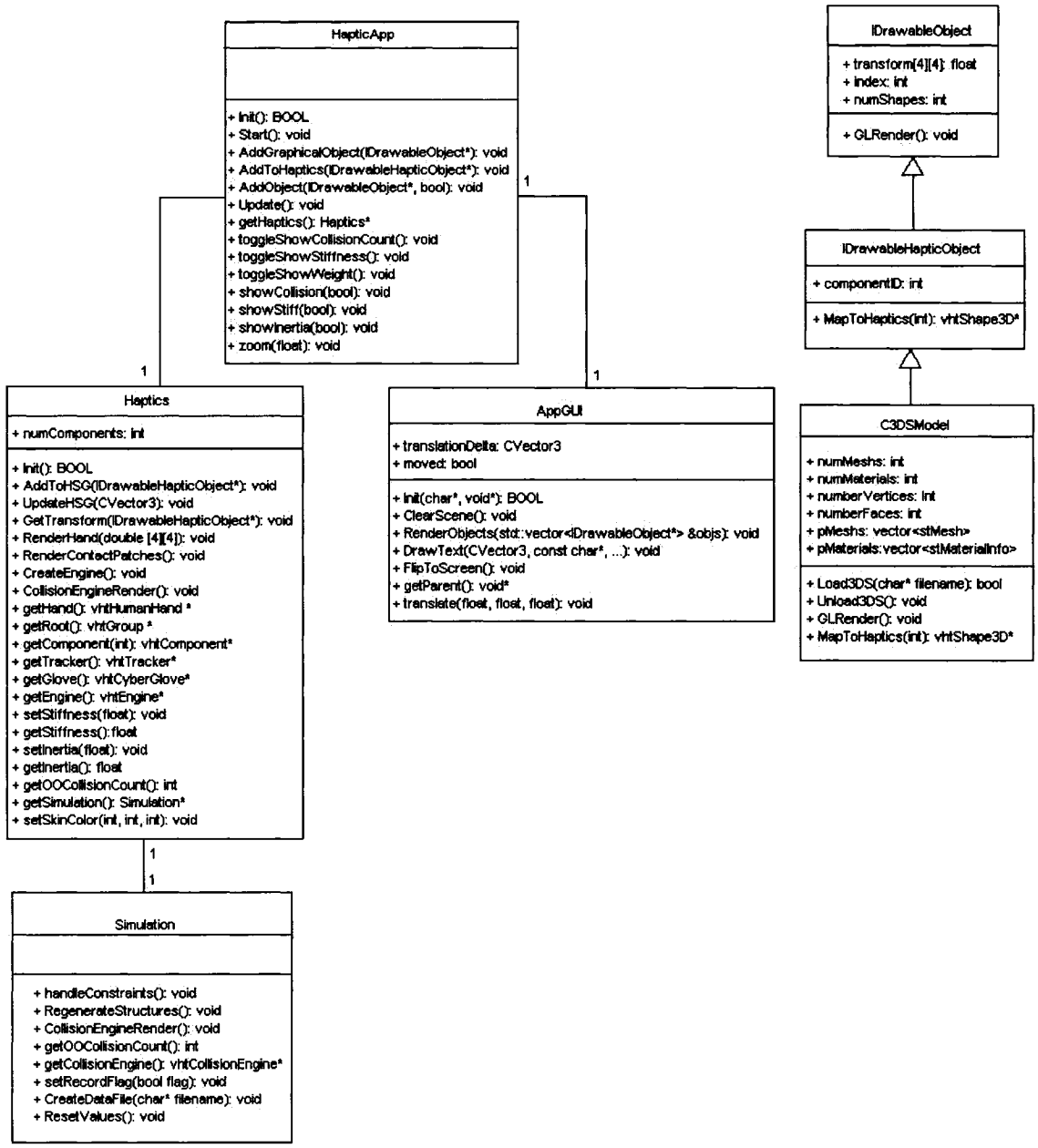


Figure 3. 5 UML Representation of the Framework Showing the Forming Classes Adding Objects to HapticApp

The haptic simulation is all about the collision detection process that handles, haptically and graphically, the interaction between the virtual hand and the 3D objects in the VE. Collision between objects is determined by a contact patch approach. The contact patch is an infinite haptic plane that allows for haptic feedback. To go about doing its job, the contact patch sets the following haptic properties: stiffness, damping, distance, and

inertia. Contact patches are usually placed where a collision occurs (at witness points); however, they can be placed anywhere in the environment. The collision detection methodology provided by the virtual hand toolkit supports two implementations of the GJK (Gilbert-Johnson-Keethri) algorithm. The two implementations are the Vclip collision factory [21] and SOLID collision factory [44]. As for "Grasping," this process allows the user to pick up the virtual object and manipulate it in virtual space. Once grasped, the object will appear to be attached to the hand until the user releases their hand. Upon release, the object will remain in its current position and orientation in space. All dynamic objects can be grasped and manipulated.

Software Development Kit (SDK)

The SDK that accompanies the CyberGrasp system provides a variety of features that are summarized in table 3.4.

Table 3.4 SDK Features that Provide the Virtual Environment

Offers an object-oriented model with an accompanying C++ library.
Provides a general framework for constructing hand-enabled simulations from scratch or for integrating hand-interaction into existing applications.
Includes a device configuration utility that lets users calibrate and configure Immersion's CyberGlove products plus supported trackers in an intuitive and graphical manner.
Offers real-time collision detection capabilities between 3D digital objects.
Provides a force feedback interface for CyberGrasp and CyberForce users.
Offers full network support. A user can run an application on a host computer while getting device data from another machine, permitting interaction with geographically distributed teams.
Grasping algorithms for picking up any arbitrary object loaded into the scene.
"Ghost-hand" support for managing position-tracker offsets to prevent the graphical hand from passing through objects.
A fast polygon-level collision-detection engine, including an open API for support of specialized third-party collision modules.
An open API for model import and interfacing with third-party visualization software.

A VRML/Cosmo (SGI Optimizer 1.2) implementation is included.
Significantly improved overall structure with better run-time integrity and more complete error handling.
A complete set of open source demonstration applications showing how each of the toolkit features can be used in your development.

3.1.2.1 Limitations

Below, the main problems faced with this framework component are laid out.

Software Development Kit

Software limitations pertaining to the virtual hand SDK toolkit lie in linking the VRML parser included in the package to the workspace. There were problems with linking libraries and thus existing projects or workspaces are used as a template for programming. Furthermore, the VRML parser provided cannot handle textured images, but color is not a problem.

Collision Detection

Another major problem is collision detection. Even though multiple collision detection is possible, multiple collision force feedback is not as simple as single object collision force feedback. There is another way to improve the precision of the collision engine by creating an object using multiple meshes to represent each crucial part of an object. For example, a teapot can be created using three meshes. One mesh will be created for the handle, one mesh will be created for the spout, and one will be created for the main body of the pot. By deeming each mesh a different object, but of the same parent, the handle, the body of the pot and the spout are better defined than considering the teapot as one whole object. There are certain limitations to the optimized geometry; there is a trade off between faster haptic and collision detection. The algorithm mentioned above limits the accuracy of more complex objects. Therefore, one might be able to gain more precise haptic representations using other collision engines. However, in order to implement collision engines other than those provided by virtual hand SDK, one should consult

Immersion corps concerning the details of doing so. Currently all objects have to be added to the haptic scene graph (root) before creating the simulation. Because of the collision engine created in the simulation class, optimized geometry for the objects is in effect when the engine is constructed. However, VHT (Virtual Hand Toolkit library that comes with the API) library provides methods to delete and add objects from the haptic scene graph and rebuild the geometry.

Grasping and Manipulating Objects

Grasping and manipulation is also a problem. Once the object is grasped it is literally attached to the hand and therefore cannot be manipulated unless that is done through the whole hand. For example, once the object is grasped, one cannot roll the object around in their hands. The object is just stuck to the hand as if it is glued until it is released. Grasping does not necessarily reflect that of real life grasping. A user might not be able to grasp the object with just two fingers and sometimes the user is capable, depending on the size and shape of the object being grasped. Furthermore, it is also possible to grasp an object without physically grasping it due to the nature of the algorithm. One can curl his/her fingers...

If one is unable to grasp an object getGraspManager ()->constrain() and getGraspManager()->reset() must be called. These two functions must be called in order to allow the hand to grasp and un-grasp the object. Finally, there should be a mechanism to check whether to set grasp or ungrasp. Usually to set or unset grasping one would use the function setGraspedPatch(vhtContactPatch::isGrasped) or setGraspedPatch(vhtContactPatch::notGrasped).

Vibrations Felt by CyberForce during Collision

Vibrations will always exist, it is only a matter of how to decrease or limit them. For most cases, intense vibrations occur when the virtual hand tries to penetrate a solid object. That is, when the contact patches are created in the hand rather than just on the surface of the hand where the actual contact should be.

There are cases where one wants the contact patches to end up inside the hand rather than just the surface, but for most simple applications, the contact patches should be created on the surface of where the collision occurred. Vibrations can be prevented if the user does not force the virtual hand to penetrate the solid object. If forced penetration is not the problem, one can try to decrease the stiffness of the contact patch or increase the damping on the contact patches. A decrease in stiffness will affect the force feedback of the CyberGrasp. For example, when grasped, the object will not feel as solid as it was prior to the change in stiffness. If the vibrations become violent, releasing the foot pedal will turn off the actuators of the CyberForce unit.

Vibrations Felt by CyberGrasp during Collision

Vibrations felt by the CyberGrasp can be due to the contact patch not being created correctly. Therefore, it is imperative that one checks all contact patch properties that are required to perform the force feedback. There was also a situation where vibrations felt by CyberGrasp were experienced which could have been caused by improper use of the update function.

When update from vhtHumanHand is used, the contact patch will be deleted and a new one will be created in its place; however, if this function is used frequently, vibrations will occur because the patch is constantly delete and recreated. One solution to this is to not create a new contact patch every time update occurs, instead, the contact patch is left and only the updateVisualGeometry() of vhtHumanHand is used. The function updateVisualGeometry() only updates the visual representation of the object and will not delete the patch like before when using update().

Inability to Feel Object

Most of the time, when one cannot feel an object, it is due to the improper use of the contact patch. If the contact patch created is not in the right place, then one will not be able to feel the object. Also if the contact patches properties were incorrect then one would not be able to feel the force feedback. For example, if one sets the stiffness to zero, then there will be no stiffness or rigidity to the object; thus, one will not be able to

feel the object. Another problem could occur when dealing with multiple objects; if the second object created is not added to the scene root, one will not be able to feel the second object.

Objects Feeling Heavy/ Weightless when Grasped

To make sure that the weight of the object is felt, the foot pedal must be applied since it turns the CyberForce actuators on. After actuators are on, if the object still feels weightless, the value entered as the contact patches inertia needs to be checked. The setInertia() function is what dictates the weight of the object. This is also true if the object feels heavy, one can decrease the inertia to decrease the weight. Another crucial step is to make sure that the object gets its mass property set and computed properly.

Jolts during Collision

Jolts can occur when the contact patch is created in virtual hand instead of being created on the surface of where the collision occurs. This problem would likely not happen unless the contact patch is assigned manually, because for most applications, the contact patch will be set at the proper offset of where the witness points are.

It is recommended that void RenderContactPatches() function should be used in order to determine where the contact patch is being created. Knowing that location will explain why the jolts are occurring.

Unwanted Attraction/ Repulsion

Attraction or repulsion that one feels is also due to the location of the contact patch that is created. This type of problem would most likely occur when the contact patch is being manually given an offset other than where the witness points are. Repulsion would occur if the contact patch were created on the surface of the hand that is closer to the object than the side colliding with it. An attraction occurs when a contact patch is created on the surface of the hand that is farther from the object one is colliding with. Therefore, it is recommended that one use the function RenderContactPatches() to visually see where the

patches are being created. Knowing the location of the patches will help explain when and where the unwanted attraction/repulsion is occurring.

Penetration of Object/ Premature Force Feedback

Penetration of the object and premature force feedback are both due to the minimal translation distance (MTD). This value is the minimum distance between two when the collision detection was be initiated. Therefore if MTD = 0.2, that means that at 2mm the collision detection will occur or force feedback will be felt at that distance. If the MTD value is negative, this will allow for penetration to occur. Usually it is recommended for a good realistic feel of an object that MTD be set to <= 0.2.

Mapping Imported 3D Images to Haptics

Mapping a foreign object to haptics entails creating an instance of vhtVertexGeometry(), then that instance will have to set the number of vertices. Afterwards, an instance of vhtShape3D will be created by passing the vhtVertexGeometry instance to the constructor of vhtShape3D. Once the instance of vhtShape3D is created, that object must be added to the scene root. Therefore, before creating instances of vhtVertexGeometry(), one needs to know the number of vertices and the corresponding matrices to map to the haptics.

Precision of Imported Objects/ Optimizing Geometry

Objects in virtual hand sdk process collision detection using optimized geometry. To imagine what this optimized geometry is, consider a plastic bag and wrap it around a star tightly. The haptic optimized geometry of the star object will create a pentagon. Wrapping a cross tightly with a plastic bag would make a diamond shaped object rather than a precise cross.

One can see that there are certain limitations on how accurately one can represent real objects haptically using the current collision engine used by virtual hand sdk. Therefore, one might be able to gain more precise haptic representations using other collision engines. However, in order to implement collision engines other than those provided by

virtual hand sdk, one should consult Immersion Corp for the details of doing so.

There is another way of improving the precision of the collision engine by creating an object using multiple meshes to represent each crucial part of an object. For example, a teapot can be created using three meshes. One mesh will be created for the handle, one mesh for the spout, and one for the main body of the pot. By making each mesh a different object, but of the same parent; the handle, the body of the pot and the spout are better defined than considering the teapot as one whole object.

One can see this theory in action by using the CollisionEngineRender() function. When using this function, each mesh should generate a bounding sphere around it. The same can be said about the virtual hand which is actually made up of multiple bounding spheres for each segment of the hand.

Multiple Collision Force Feedback

In the situation of multiple collision force feedback, the user is dealing with two or more objects. The first thing a user has to do is to make sure that each object created is added to the root node separately using the addChild() function and also create a separate vhtComponent for each object. These objects will be both added to the root node, but their parent node will be different. If their parent nodes are the same, the two objects will be considered to be one object; in other words, when one object is moved, the other follows.

The goal of multiple force feedback collision is to allow a user to grasp one object and collide that grasped object with another. The user should feel as if he or she used an object to push against another solid object and feel resistance. In order to feel this object on object collision, one would need to create a contact patch on the palm of the virtual hand when the grasped object collides with another object. Therefore, one has to determine the location of the palm node that is required to set the position of the contact patch offset. Also, one needs to determine the direction of travel of the grasped object in

order to determine the normal of the palm to perform the calculation of selecting the proper palm node for collision.

To determine the normal to be taken, the virtual hand has to be tracked to determine which axis it is moving along. To track the movement, the vhtTracker class was used to track the position of the virtual hand. As the hand moves, the previous position of the hand and the current position are compared.

Depending on whether the current position is greater than or less than the previous position along an axis, the proper normal is assigned. Furthermore, to prevent the normal from changing too quickly, the difference between the current and previous position has to be greater than a specified value before evaluating which normal to use.

Sample pseudo-code: Checking direction of travel and setting normal

```
// check if movement is large enough to warrant a change in normal
If((fabs(currentpos.x - oldposition.x) > 2.0) ||
      (fabs(currentpos.y - oldposition.y) > 2.0) ||
      (fabs(currentpos.z - oldposition.z) > 2.0))

      // check if x movement is greater than y & z movement
      if(((fabs(currentpos.x - oldposition.x)) > (fabs(currentpos.y -
      oldposition.y))) && ((fabs(currentpos.x - oldposition.x)) >
      (fabs(currentpos.z - oldposition.z))))

      if(currentpos.x > oldposition.x)
      tempNormal.x = -1.0;

      else
      tempNormal.x = 1.0;

      tempNormal.y = 0.0;
      tempNormal.z = 0.0;
```

As one can see from above, this is how the normals were set for direction of movement along x-axis. Movements along the y and z axis are done in a similar fashion.

When developing multiple collision force feedback, it is ideal to deal with collisions one dimension at a time and the easiest dimension to test the collision on is in the x domain. Therefore, the explanation of palm node selection will be dealt with in the x domain. The incorporation of the other dimension will be discussed later on in this

explanation. To create the proper patch at the right location, one would need to determine the proper palm node. The normal will always be the opposite to that of the direction of motion. For example, if the grasped object is traveling from right to left, the normal will be in the positive x direction and if the grasped object collides with another object on the left hand side, the contact patch should be created on the surface of the palm closer to the collision between the two objects.

The proper palm node is the node that is farther from the palm center. To find this node, a for loop was used to loop through all palm nodes to determine which one is greatest by subtracting the palm node coordinate with the palm center and performing a dot product with the normal vector for the x direction. After the dot product is performed, the proper palm node will be selected. Ideally, the contact patch will be created at the same coordinate of where this palm node is; however, the feeling of a contact patch won't be really solid. Therefore one can place the contact patch slightly inside the palm. Note that this was an option, to allow for a better feel of the collision. One might decide that the feeling of the contact patch being created right where the palm node is might be sufficient to convey collision or feedback. See Figures 3.6 to 3.8 below.

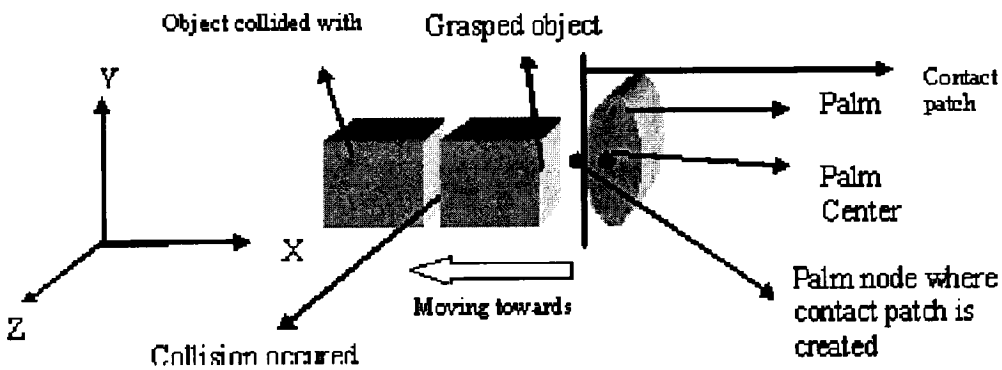


Figure 3. 6 Contact Patches Simulate Multiple Collisions

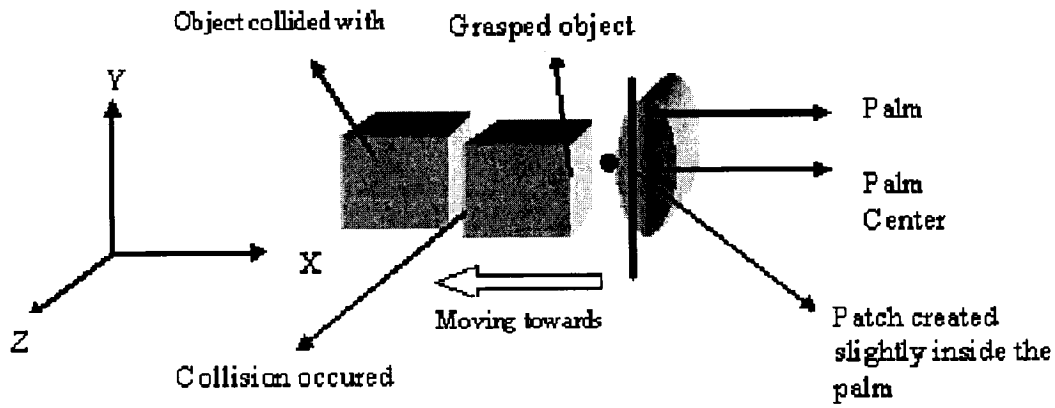


Figure 3. 7 Tweaking Collision Feedback by Contact Patches

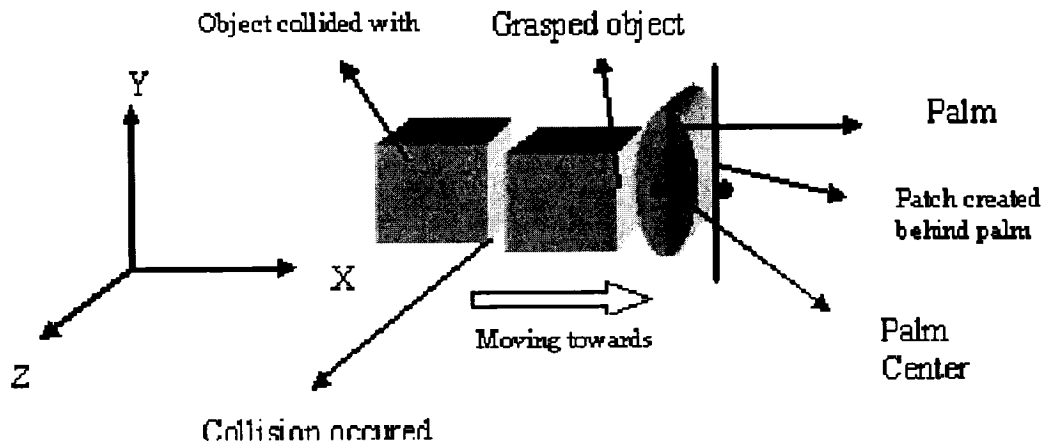


Figure 3. 8 Contact Patches Simulating Attraction

3.1.3 Haptic/ Behavioural Data Component

The haptic/behavioural data system component involves processing data files obtained from subjects' trials. The data files show several physical attributes that were recorded such as the time stamp that establishes the start and end of each exercise, the 3D world coordinates of the virtual position, and the joint angles for the middle phalange of each finger. Additional information can be computed, like the total distance covered and average velocity along the X, Y, and Z axis. The results and analysis will be presented in chapter 4.

3.1.4 Client Application Component

The client application features three virtual reality exercises that require the use of the haptic devices. The exercises involve handling virtual objects of different shapes in the 3D environment to accomplish certain tasks.

The data collected and used for analysis was taken after familiarizing the subjects with the exercises. This involved taking the five subjects that were put to the tests through several trials over a period of three weeks before running the final trial for each subject. All the subjects were seven healthy male, right-handed University of Ottawa students 22 to 36 years of age. After a preliminary analysis the three subjects with the most stable performance were chosen to undergo the final tests.

A typical session starts with initiating the environment and starting the SDK (Software Development Kit) device manager, as shown in figure 3.9 below. The subject wears the glove before the CyberGrasp is hooked up to the fingers. Insuring that the sensors are not too tight around the subject's fingers is important since the subject's comfort is the first step toward a session of accurate measurement taking. One more task to fulfill before starting an exercise is calibration.

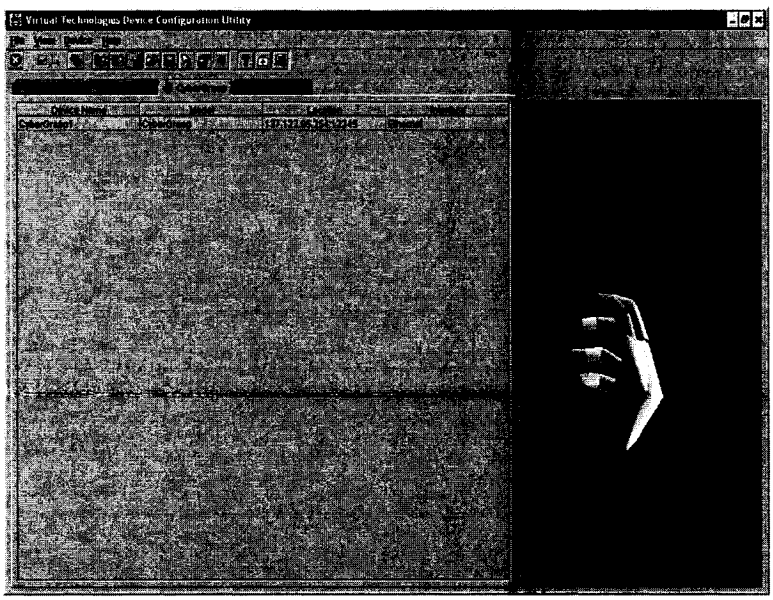


Figure 3.9 Driven Screen Allowing for the Manipulation of the Virtual Hand

3.1.4.1 Calibration Process

As no two users will have the same grasp or pattern of movement, the SDK comes with a calibration feature that will take into account a user's preferences and orientation in 3D space. Calibration begins by requesting the user to curl his/her index finger and touch the thumb. Next comes a phase that requests the user to curl his/her fingers excluding the thumb toward the palm of the hand a number of times until the system forms a profile for the user's movement of fingers. The same process is repeated for the thumb. Calibration is complete with adjusting the tracker position on the screen according to the user's preference. All the user needs to do is pick a position for his seating and arm movement that is comfortable and which allows for complete freedom to perform any exercise easily. A typical calibration process takes less than two minutes to complete, and becomes even shorter when a user becomes more accustomed to the haptic device, though the SDK offers to save that setting for the user so that the calibration process needs not be repeated every time that user logs in to perform an exercise. Figures 3.10 to 3.14 show snapshots of the calibration screens for a sample trial.

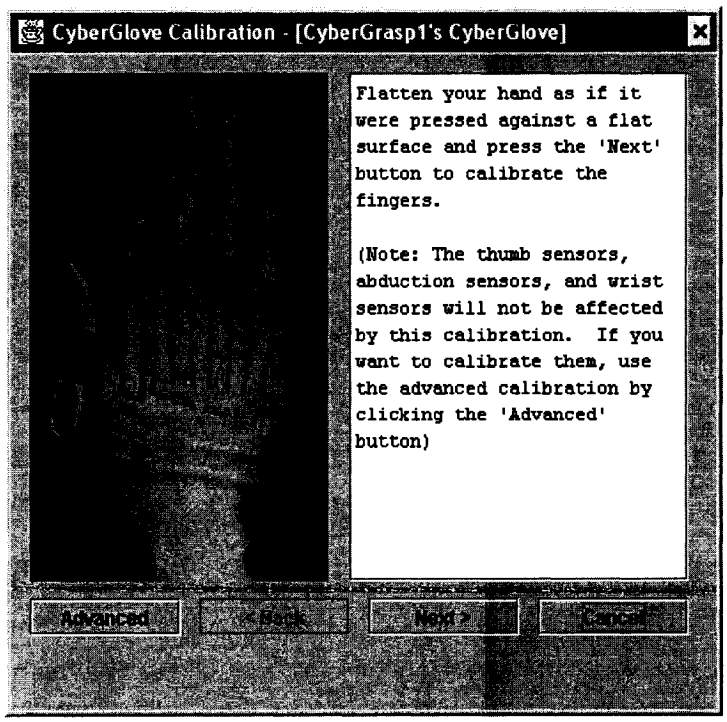


Figure 3.10 Step of Calibration Process for CyberGlove; Start with Hands Flat

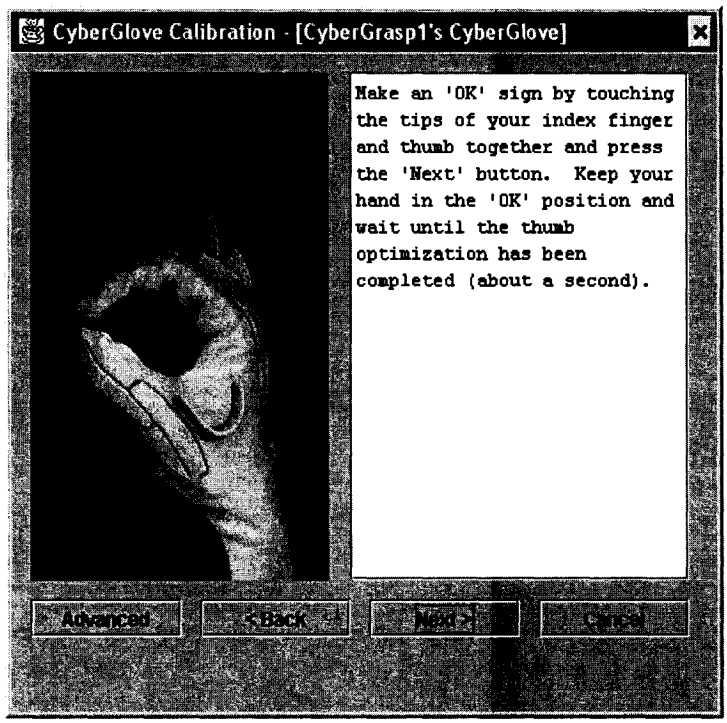


Figure 3.11 Second Step of Calibrating CyberGlove; Make an OK Sign

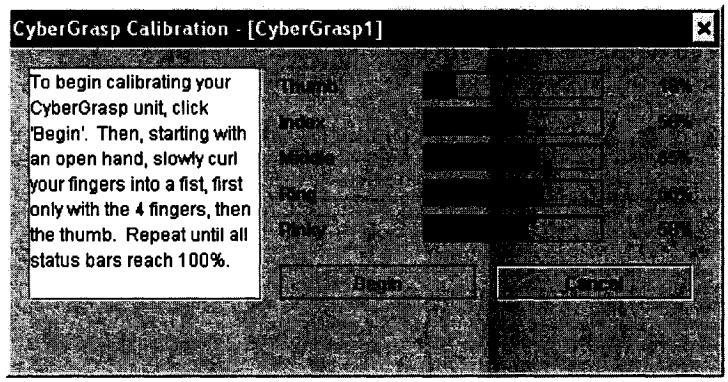


Figure 3.12 Calibrating CyberGlove Final Step; Curl Fingers and Thumb

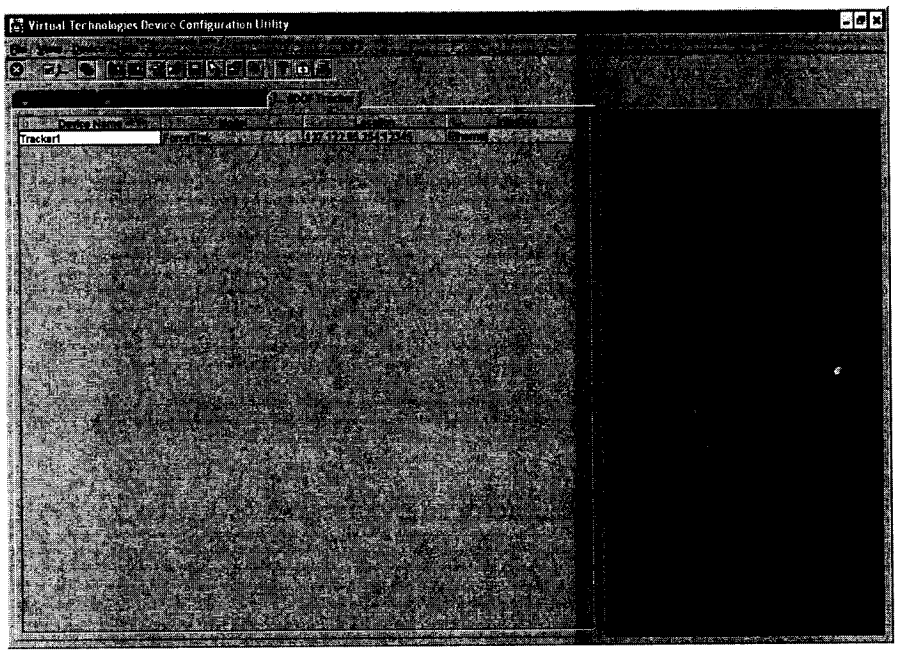


Figure 3.13 Calibrating Tracker; Subject Gets into Comfortable Position

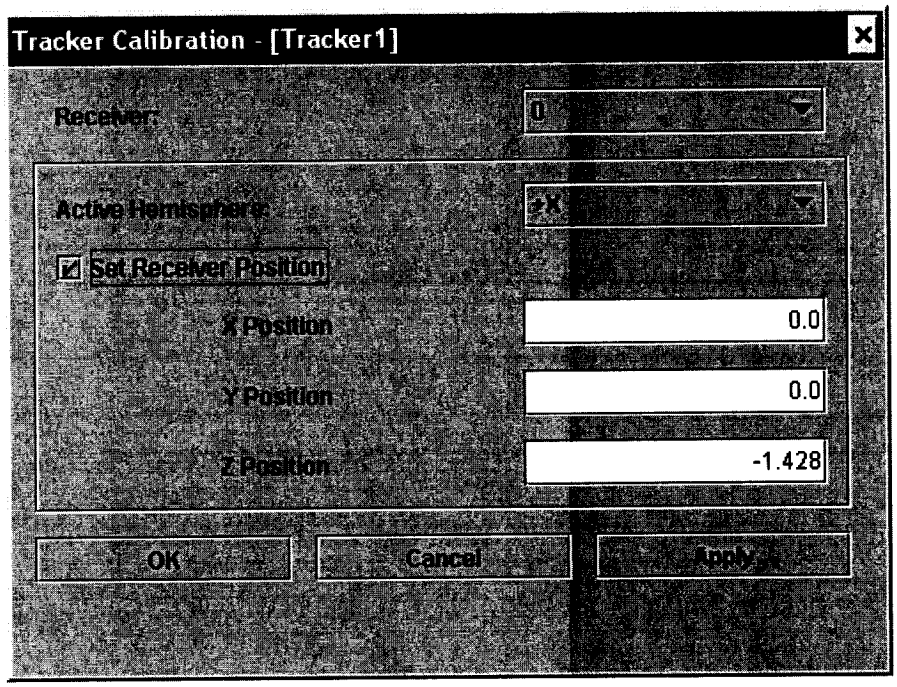


Figure 3.14 Calibrating Tracker; Lock Position

3.1.4.2 Exercises

Three exercises have been designed, each of which aims to test certain abilities of a subject. Based on the findings and experiences of previous related works, these exercises form a basis for future improvements in this field of haptic-based rehabilitation. The simplicity of the exercises is important since they should eventually indulge recovering stroke patients with limited hand function abilities. They are diverse enough to allow for a combination of exercises that can be appointed by an occupational therapist according to each patient's case. A patient with a more severe case would typically start with one or two exercises. Varying the difficulty and intensity of the exercises is a decision for an OT (Occupational Therapist) to make, as well as adding or omitting an exercise from a patient's rehabilitation schedule.

The three exercises are:

- Handling a virtual cylindrical cup across the virtual space.
- Moving a set of eight virtual cubes across a partition and arranging them according to a color-based model.
- Navigating a virtual maze using a virtual stick.

The data recorded throughout an exercise provides information about: the X, Y, and Z position of the hand on the screen, the proximal angle (middle phalange) made by each finger, and lastly, the time elapsed during the course of the exercise. More information was extracted from this set of raw data: velocity across each of the X, Y, and Z axis; total distance covered across each axis; and the idle time for each finger during an exercise. These findings will be discussed in chapter 4.

Handling a Cup

The subject has a view of a cylindrical cup that he/she can grab by moving their hand which appears on the screen. There is a virtual ceiling that the subject can touch using the cup. See figure 3.15.

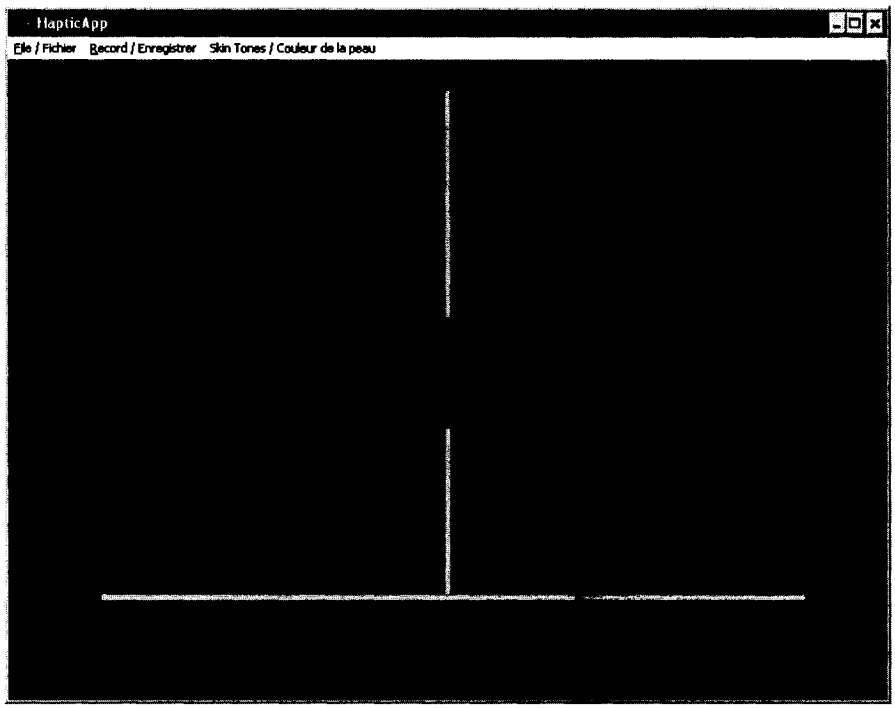


Figure 3.15 Snapshot of the Cup Exercise

The exercise was performed three times, once a day by each subject, setting the weight to the maximum that can be handled by the CyberGrasp cables (25 units). The subjects were asked to lift the cup in a vertical motion and touch the ceiling. The collision detection algorithm available allows a subject to feel that collision. Since repetitiveness is a key in rehabilitation, this custom was honoured by requesting that each subject performs those tasks ten times. The subjects were asked to move along the vertical line shown in the figure. They were also instructed to keep the hand as steady as possible and avoid movement 'into' the screen, i.e. avoid movement along the Z axis. A firm enough grip to be able to grab and hold a drinking cup is an obvious goal of rehabilitation for a recovering stroke patient. The grabbing motion and the time elapsed with the subject keeping that grip on the cup while performing the exercise aim to accomplish that goal of helping a future patient to improve his/her grip.

Arranging Cubes

This is the most difficult exercise to perform. It involves eight cubes with varying colors, and a virtual partition placed in the middle of the screen. Each cube's weight was set to the minimum allowed (1 unit). The subjects were asked to move the cubes, one by one, from one side of the virtual space to the other across the partition. The partition will not allow for a cube to penetrate it due to the collision detection algorithm. The subjects were then instructed to arrange the cubes according to color to form one big cube with six surfaces, each surface having one uniform color. Again, each subject performed the exercise three times on three separate days. See figure 3.16 below.

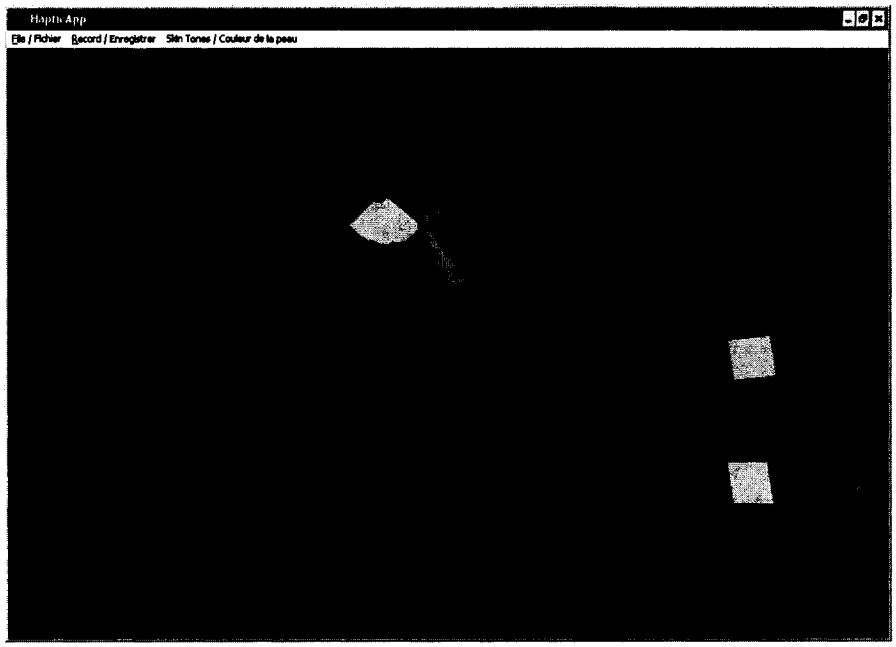


Figure 3.16 Snapshot of the Cubes Exercise

Not only does this exercise help a future patient develop a feel for an object with sharp edges, but it also tests his/her perception of patterns and also the agility and strength of hand to grab, move, and arrange the cubes. A patient on course of recovery will show more stability as his/her condition improves.

Navigating a Maze

As figure 3.17 shows, a subject sees a maze and a stick with a thin cylindrical shaped handle.

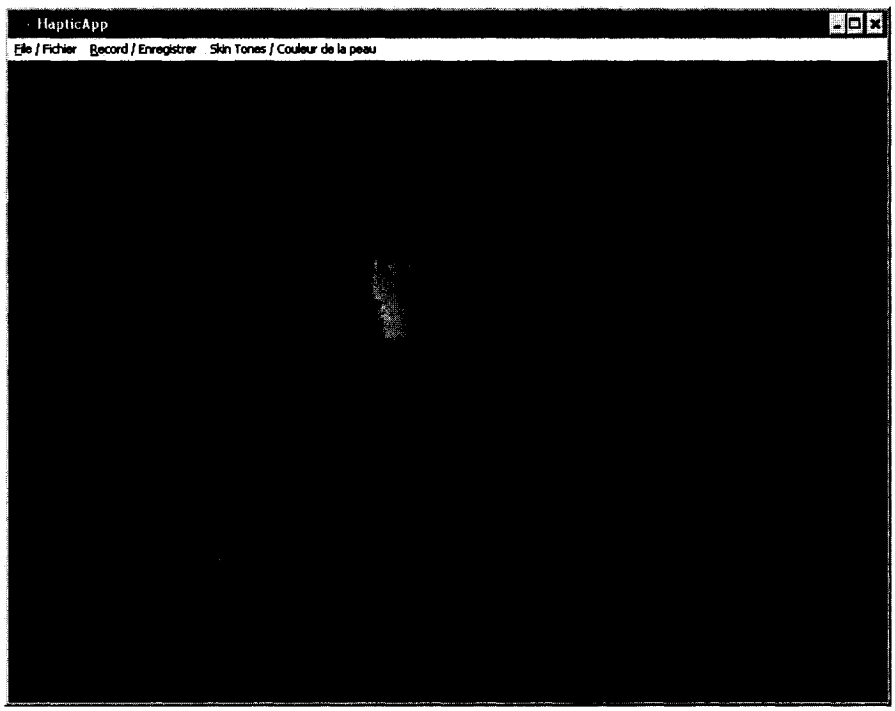


Figure 3.17 Snapshot of the Maze Exercise

Though there is the task of grabbing the stick, the main task here is to navigate the maze using the stick. For a future patient, this test can be thought of as an exercise of improving the steadiness of the hand while performing a task which requires some concentration (not colliding with the walls). Like the two other exercises, the maze exercise was performed once a day on three different days.

The framework's components have been described in this chapter. The virtual exercises produced numerous results that were analyzed and are presented in chapter 4.

CHAPTER 4

RESULTS AND ANALYSIS

The haptic-based exercises that were conducted were used to extract raw data that was processed to evaluate a subject's performance. As mentioned in a chapter 3, the raw data parameters recorded in each exercise included: the angle of the middle phalange of each finger, the X, Y, and Z coordinates of the hand, and the time elapsed to conclude the exercise. That data was processed to yield: the total distance covered across each axis, the idle time of each finger, and the average velocity of the hand during a session. In this chapter the findings will be presented and analyzed to:

- Infer information that pertains to the way that each subject manipulated the virtual objects;
- Deduce tips and recommendations that will help occupational therapists in the future to assess a post-stroke patient's performance of virtual rehabilitation exercises inspired by the ones presented so far. Assessment based on continuous evaluation will make the recovery process faster and more efficient.

4.1 Data Collected and Extracted

The analysis done on the recorded data from each exercise (proximal angle of finger, position of hand on screen, and time elapsed) yielded the following information for each subject:

- The distance covered by the hand along each axis (measured in centimetres)
- The Idle time of each finger (the change in the proximal angle was considered, measured in seconds). To find the idle time, the number of points in time for which there is no movement is computed. The resulting number is subtracted from the total time for the exercise in question.
- The speed attained during each exercise, along each axis, and the average speed for all axes (measured in cm per seconds).

The virtual hand suite allows for the recording of all the angles of each finger. Figure 4.1 shows the sensors available on the CyberGlove. The SDK can be programmed to record the angle of the middle phalange (proximal) of each finger. Since this is the angle that exhibits the wider range of change, it was chosen for the analysis. A wide range of change in the angle allows for better results when processing data. In the future, occupational therapists will be able to better analyze a patient's hand performance if angle changes in the finger vary within as wide a range as possible.

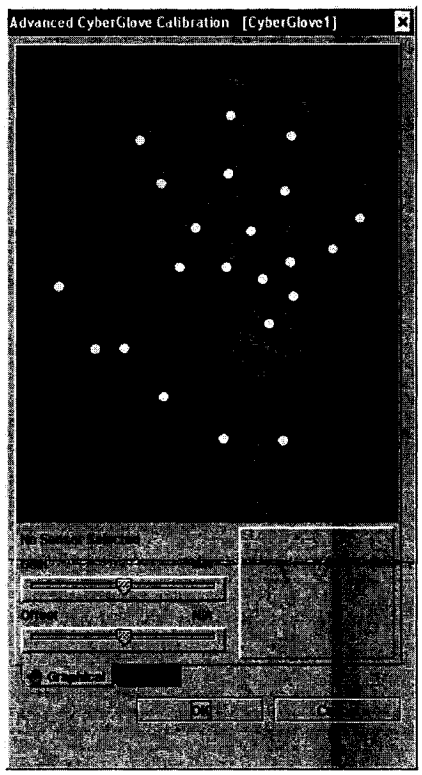


Figure 4.1 Calibration Option of CyberGlove Showing all Sensors

4.2 Analysis

Analyzing data extracted from the exercises was done to fulfill two objectives:

- Provide an occupational therapist with useful ways of looking at a future patient's performance by specifying certain parameters that should be monitored. Continuous evaluation of these parameters will direct the therapist to devising updates to the rehabilitation course of the concerned patient.
- Deduce patterns about a specific subject's performance in any exercise and in the overall manner by which the subject goes about using his/her hand. This will help a therapist in the future to devise a 'personalized' rehabilitation course for a patient.

The information that is in the graphs to grasp will be used by therapists to:

- Monitor the changes in the total distance covered during each exercise. This will allow the therapist to determine improvements in performance. Taking the maze exercise for example, the distance covered along the Z axis can be taken as a deviation from the XY plane during the exercise; since the exercise involves navigating the maze in the X and Y dimensions only, a decrease in 'Z' distance denotes improvement performance. That rationale cannot be applied when studying the cubes exercise. Moving and arranging the cubes requires movement along all axes. However, the total distance covered can be a measure of how easily the patient handles the cubes. Values for distance are rounded to the nearest whole number.
- Look at the idle time for each finger during the cup exercise is a measure of the firmness of the grip. By looking at each finger, a therapist can determine the condition of that finger and implement changes to the rehabilitation exercises accordingly. Values for time are taken to two decimal points.
- The speed of the hand while performing an exercise is an obvious measure of the overall performance. An increase in speed will be an indication of improvement. All speeds values are rounded off to the nearest whole number.

Next, the three exercises will be studied one at a time.

4.2.1 Cup Exercise

The cup exercise involves vertical motion and therefore all three subjects showed a larger distance covered along the Y axis.

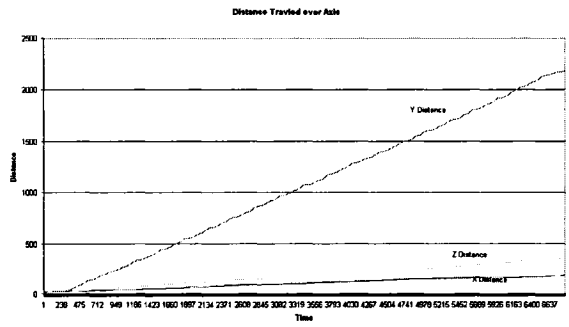
Distance Traveled over Axes

Two parameters were extracted from the distance covered: the average speed attained over the Y axis, and the distance accumulated over the Z axis.

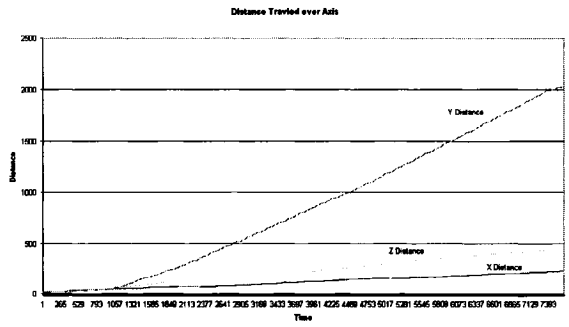
Speed along Y Axis

The speed attained over the Y axis can be used by a therapist to determine a patient's comfort in performing the task of lifting a cup of water. Distance accumulated over the Z axis can give a therapist an idea of the deviation from the vertical path that a patient exhibits. The results obtained from healthy subjects may be used by a therapist to set a target for the patient at the end of a time period that the therapist determines. That time period depends on the patient's severity of infliction and his/her progress.

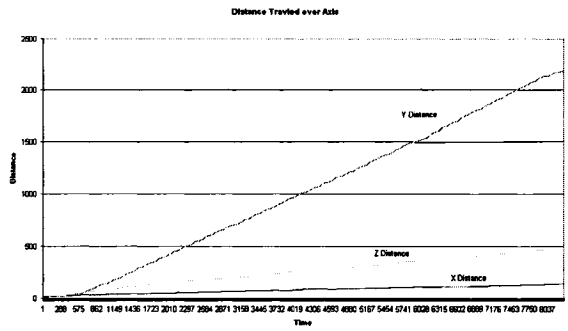
Graphs 4.1 to 4.3 show the distance covered along each of the X, Y, and Z axes for subject 1. Subject 1, as with the other two subjects performed the last phase of testing three times; each graph represents a trial. Cup1 means the first trial of the cup exercise.



Graph 4.1 Distance Traveled, Cup1, Subject 1



Graph 4.2 Distance Traveled, Cup2, Subject 1



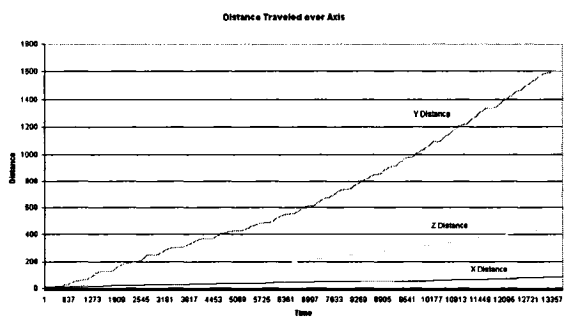
Graph 4.3 Distance Traveled, Cup3, Subject 1

The values for speed attained over the Y axis and the distance accumulated over the Z axis for subject 1 for each trial were extracted from the graphs and are shown in Table 4.1 below.

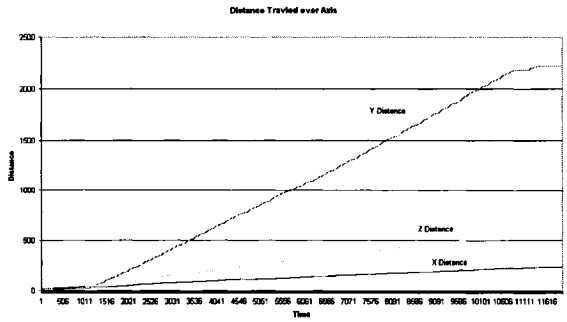
Table 4.1 Cup Exercise for Three Trials for Subject 1Showing Time Elapsed, Distance Covered along Y Axis, Distance Covered along Z Axis, and the Hand Speed Attained along Y Axis

Trial	Time (seconds), Trials 1, 2, 3	Distance Y (cm), Trials 1, 2, 3	Distance Z (cm), Trials 1, 2, 3	Speed Y (cm per s), Trials 1, 2, 3
1	6.83	2183	365	320
2	7.61	2028	448	266
3	8.28	2183	488	264

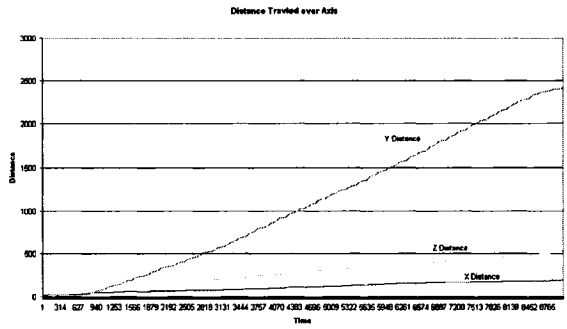
Computing the average 'Y' speed for subject 1 for all three trials yields 283 cm. By looking at graphs 4.4 to 4.6 for subject 2 for the cup exercise, the same analysis is performed.



Graph 4.4 Distance Traveled, Cup1, Subject 2



Graph 4.5 Distance Traveled, Cup2, Subject 2



Graph 4.6 Distance Traveled, Cup3, Subject 2

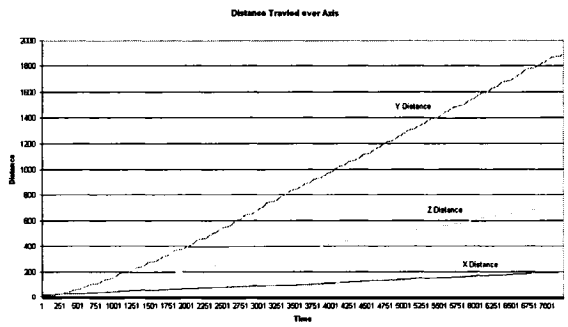
Table 4.2 tabulates the information extracted from the graphs.

Table 4.2 Cup Exercise for Three Trials for Subject 2 Showing Time Elapsed, Distance Covered along Y Axis, Distance Covered along Z Axis, and the Hand Speed Attained along Y Axis

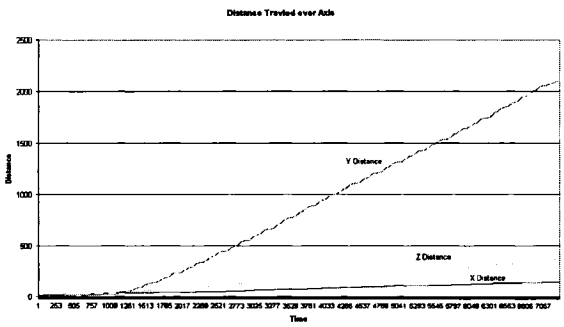
Trial	Time (seconds), Trials 1, 2, 3	Distance Y (cm), Trials 1, 2, 3	Distance Z (cm), Trials 1, 2, 3	Speed Y (cm per s), Trials 1, 2, 3
1	13.64	1603	453	118
2	11.97	2223	585	186
3	9.03	2414	531	267

Computing the average 'Y' speed for subject 2 for all three trials yields 190 cm.

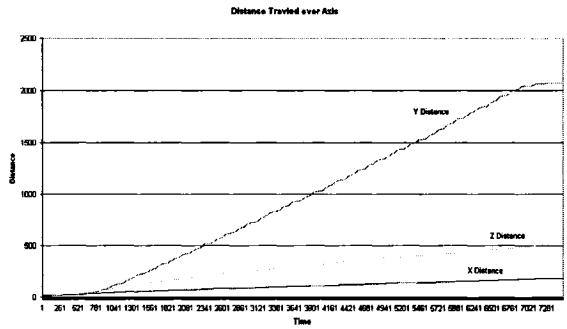
Similarly, graphs 4.7 to 4.9 convey that information for subject3 for three trials.



Graph 4.7 Distance Traveled, Cup1, Subject 3



Graph 4.8 Distance Traveled, Cup2, Subject 3



Graph 4.9 Distance Traveled, Cup3, Subject 3

As before, the results are summarized in table 4.3 below.

Table 4.3 Cup Exercise for Three Trials for Subject 3 Showing Time Elapsed, Distance Covered along Y Axis, Distance Covered along Z Axis, and the Hand Speed Attained along Y Axis

Trial	Time (seconds), Trials 1, 2, 3	Distance Y (cm), Trials 1, 2, 3	Distance Z (cm), Trials 1, 2, 3	Speed Y (cm per s), Trials 1, 2, 3
1	7.22	1892	721	262
2	7.28	2102	376	289
3	7.51	2078	512	277

Computing the average 'Y' speed for subject 2 for all three trials yields 276 cm.

The three subjects went through several tests and were picked from among a preliminary seven subjects after they showed the most consistency in their performances. Given that, a therapist can predict what a recovering patient should attain in terms of speed attained during the cup exercise. The results obtained here suggest a value of around 250 cm per seconds, which is the average of the three subjects' speeds. But looking at subject's 2 values for speed, the first trial yielded a considerably lower value than that subject's 2 proceeding trials. Taking subject's 2 average for speed to be that of the last two trials only and incorporating that into the final average for the three subjects gives an average 'Y' speed of 262 cm per seconds. Nevertheless, a therapist can determine the target to be attained by a patient given that patient's initial case. In all cases, a range of 190 cm per second and above could be taken as a measure of sound progress and the patient can be assessed against that range.

Distance Accumulated over the Z Axis

The 'Z' distance for each subject for the three trials is shown in tables 4.1 to 4.3. Distance accumulated over the Z axis shows the deviation from the vertical path of motion. A therapist can view a decrease in 'Z' distance as a sign of improvement. The three healthy subjects show an average deviation from the vertical path of 434, 523, and 536 cm for subjects 1, 2, and 3 respectively. Taking the average might give a therapist a target of 498 cm to be attained with a patient. However, seeing that each subject's vertical covered distance was different from the other two subjects, a percentage of the 'Z' distance of the 'Y' distance covered might be a more accurate measure to assess improvement of a patient.

Table 4.4 'Z' Distance for Cup Exercise as Percentage of 'Y' Distance for all Three Subjects for Three Trials

Subject	'Z' Distance as Percentage of 'Y' Distance for Trials 1, 2, 3	Average Percentage
1	16.7, 22.1, 22.4	20.4
2	28.3, 26.3, 22.0	25.3
3	38.1, 17.9, 24.6	26.9

Table 4.4 depicts the percentages. Taking the average of the values in the last column of table 4.4 gives 24.2 %, which might be a target set by a therapist.

Finger Idle Time

Charts 4.1 to 4.3 show the idle time for each finger during the cup exercise for each of the three trials for subject 1.

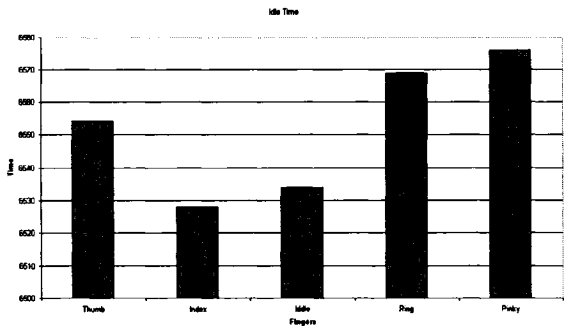


Chart 4.1 Finger Idle Time, Cup1, Subject 1

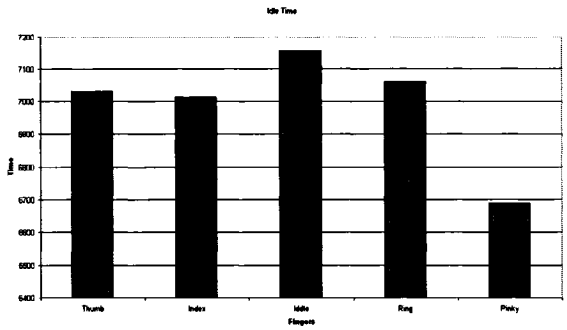


Chart 4.2 Finger Idle Time, Cup2, Subject 1

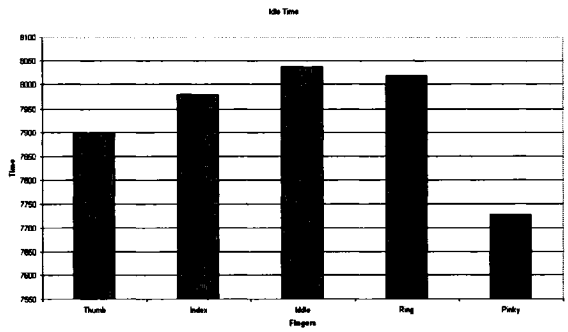


Chart 4.3 Finger Idle Time, Cup3, Subject 1

Table 4.5 summarizes the information conveyed for subject 1.

Table 4.5 Finger Idle Time (seconds) for Cup Exercise for each Finger for Subject 1 for Three Trials

Finger	Trial 1	Trial 2	Trial 3
Thumb	6.55	7.03	7.90
Index	6.53	7.01	7.98
Middle	6.53	7.16	8.04
Ring	6.57	7.06	8.02
Pinky	6.58	6.69	7.73

As the cup exercise involves moving the cup vertically, the grip is expected to be firm since no there is no manipulation of the cup otherwise. The values for finger idle time show that these values were very similar for each trial performed by subject 1. A low difference in idle time among the fingers can be a target set by a therapist when treating a patient.

Charts 4.4 to 4.6 depict the finger idle time for subject 2 for the three trials respectively.

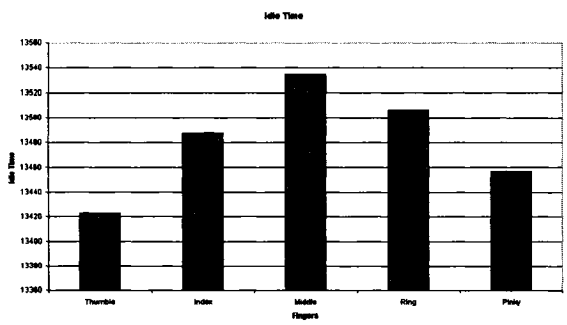


Chart 4.4 Finger Idle Time, Cup1, Subject 2

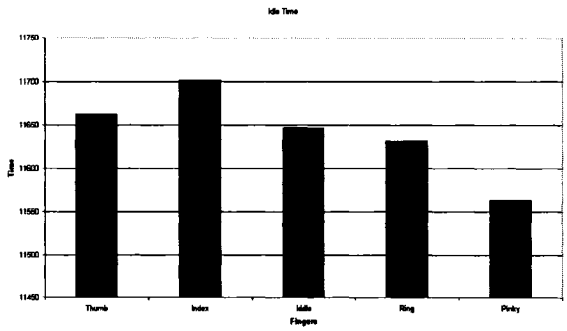


Chart 4.5 Finger Idle Time, Cup2, Subject 2

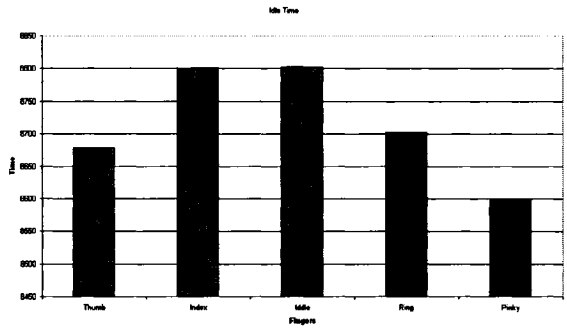


Chart 4.6 Finger Idle Time, Cup3, Subject 2

Extracting the information from these charts gives table 4.6 below.

Table 4.6 Finger Idle Time (seconds) for Cup Exercise for each Finger for Subject 2 for Three Trials

Finger	Trial 1	Trial 2	Trial 3
Thumb	13.42	11.66	8.68
Index	13.49	11.70	8.80
Middle	13.54	11.65	8.80
Ring	13.51	11.63	8.70
Pinky	13.46	11.56	8.60

Subject 2 exhibits higher finger idle times that subject 1, but this is due to that subject taking longer to perform the exercise. Nevertheless, it can be seen how small the difference in idle time between the five fingers is for each trial.

Finally, charts 4.7 to 4.9 depict the finger idle time for subject 3 for trials 1, 2, and 3 respectively.

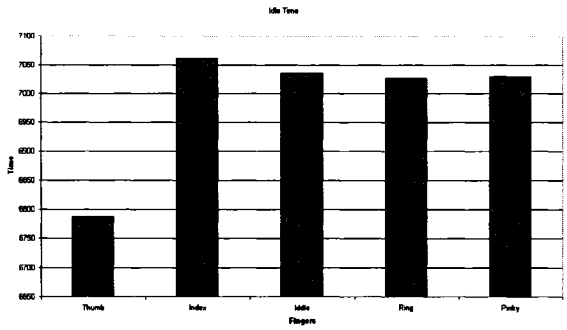


Chart 4.7 Finger Idle Time, Cup1, Subject 3

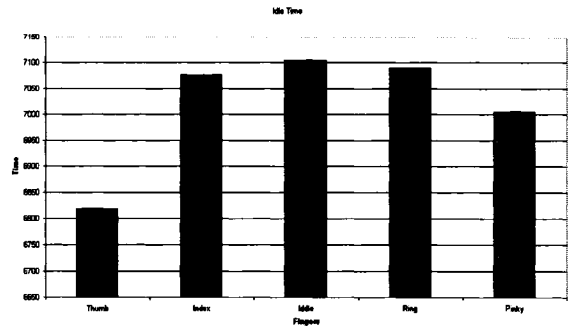


Chart 4.8 Finger Idle Time, Cup2, Subject 3

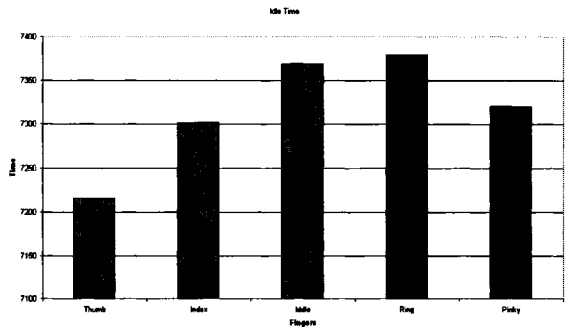


Chart 4.9 Finger Idle Time, Cup3, Subject 3

Putting the information from the charts into a table gives table 4.7 below.

Table 4.7 Finger Idle Time (seconds) for Cup Exercise for each Finger for Subject 3 for Three Trials

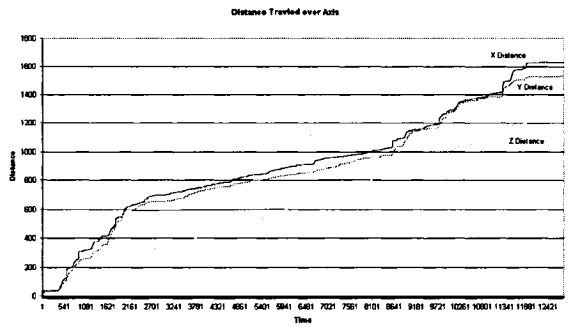
Finger	Trial 1	Trial 2	Trial 3
Thumb	6.79	6.82	7.22
Index	7.06	7.08	7.30
Middle	7.04	7.10	7.37
Ring	7.03	7.09	7.38
Pinky	7.03	7.01	7.32

Yet again, a small difference in idle time occurs among the five fingers for each trial. A therapist can safely use the criteria of fingers exhibiting smaller differences in their idle times during the course of rehabilitation while performing the cup exercise as a sign of improvement.

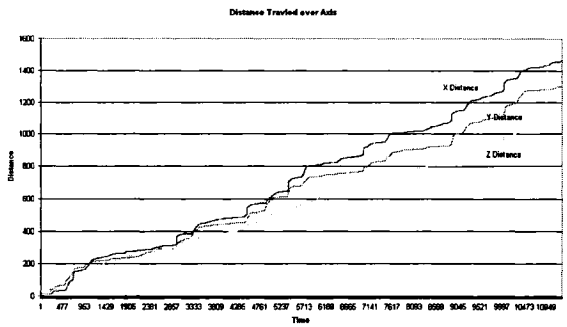
4.2.2 Cubes Exercise

The cubes exercise requires movement along all axes. Since the objective is to arrange the eight cubes in one area of the virtual space, it is expected that an efficient performance by a healthy subject should yield close values for the total distances covered along each axis. The speed of performing the exercise is also a measure of a good performance. Finger idle time is not critical in assessing performance as this exercise requires lots of movement and involves picking up the cubes and then releasing them.

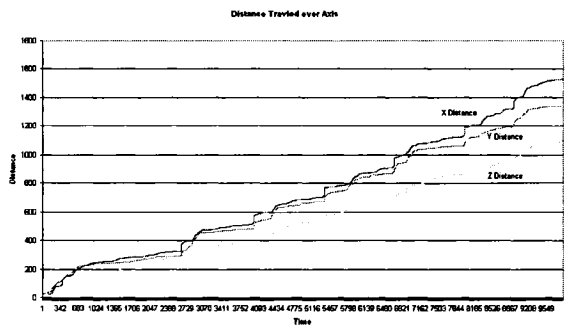
Graphs 4.10 to 4.12 show the distance covered along each axis for three trials for subject 1.



Graph 4.10 Distance Traveled, Cubes1, Subject 1



Graph 4.11 Distance Traveled, Cubes2, Subject 1



Graph 4.12 Distance Traveled, Cubes3, Subject 1

Information about the time elapsed during the exercise, the distance accumulated over each axis, and the total average speed along each axis for all three trials is tabulated in 4.8.

Table 4.8 Cubes Exercise for Three Trials for Subject 1 Showing Time Elapsed, Distance Covered along all Axes, and Average Hand Speed Attained along all Axes

Trial	Time (seconds)	Distance X (cm)	Distance Y (cm)	Distance Z (cm)	Average Speed over all Axes (cm per second)
1	12.81	1632	1533	1216	114
2	11.29	1469	1303	1097	114
3	9.86	1536	1345	1094	134

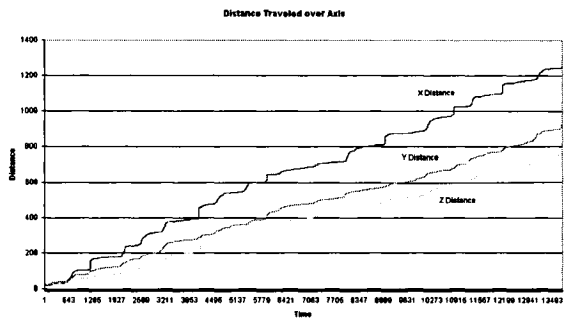
The last column can be used to compute the average hand speed of subject 1 during the cubes exercise; around 121 cm per seconds.

Another observation that can be useful to a therapist is the difference in distance covered between each axis for a particular patient. For this exercise, that difference can be an indication of the efficiency of performing the exercise. A lower value signals a better performance since it indicates a more compact manner of going about performing the exercise. The tests for subject 1 show that the 'X' distance was always the greatest with 'Z' distance being the lowest. Taking the difference in distance covered between the highest and lowest values for each trial for each subject yields table 4.9 below. An average difference of 410 cm is averaged in this case.

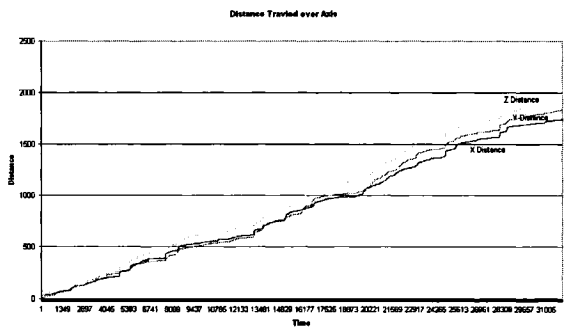
Table 4.9 Difference between Greatest and Lowest Distance for Cubes for Subject 1 for Three Trials

Trial	Difference between Greatest and Lowest Distance (cm)
1	416
2	372
3	442

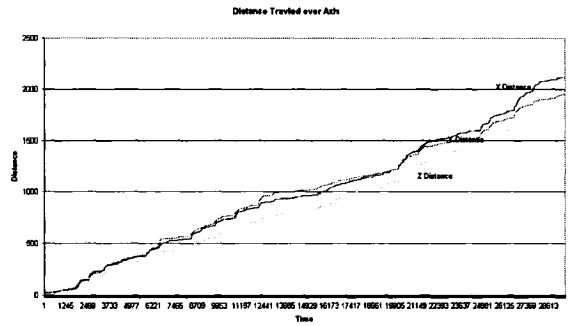
Graphs 4.13 to 4.15 convey the finger idle times for subject 2.



Graph 4.13 Distance Traveled, Cubes1, Subject 2



Graph 4.14 Distance Traveled, Cubes2, Subject 2



Graph 4.15 Distance Traveled, Cubes3, Subject 2

Table 4.10 below summarizes the information conveyed in graphs 4.13 to 4.15 below.

Table 4.10 Cubes Exercise for Three Trials for Subject 2 Showing Time Elapsed, Distance Covered along all Axes, and Average Hand Speed Attained along all Axes

Trial	Time (seconds)	Distance X (cm)	Distance Y (cm)	Distance Z (cm)	Average Speed over all Axes (cm per second)
1	13.76	1254	925	770	71
2	32.00	1755	1842	1933	58
3	29.54	2118	1949	1924	68

Computing the average speed over all axes attained gives 66 cm per second, which is considerable lower than that of subject 1 (121 cm per second). A look at the results of subject 3 for the cubes exercise can better determine what target a therapist should put for

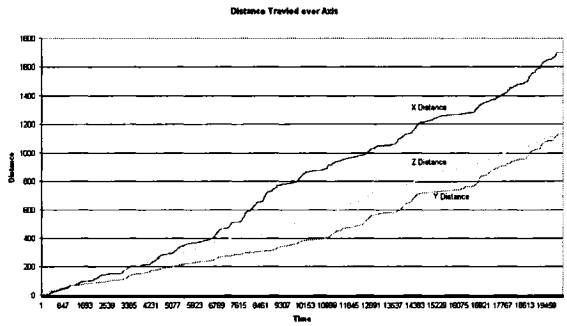
total average speed when treating a patient. But before, a similar analysis to that done on subject 1 on the differences in distance covered along each axis is presented in table 4.11.

Table 4.11 Difference between Greatest and Lowest Distance for Cubes for Subject 2 for Three Trials

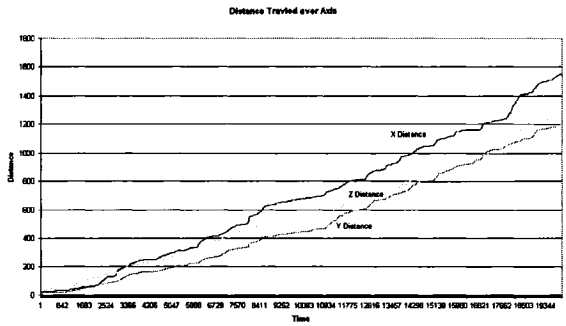
Trial	Difference between Greatest and Lowest Distance (cm)
1	484
2	178
3	194

Except for trial 2 where 'X' distance was smallest and 'Z' greatest, the 'X' distance was greatest for the other two trials, the 'Z' being smallest, like with subject 1. An average difference of 285 cm was computed. As shown earlier, subject 1 exhibited an average of 410 cm.

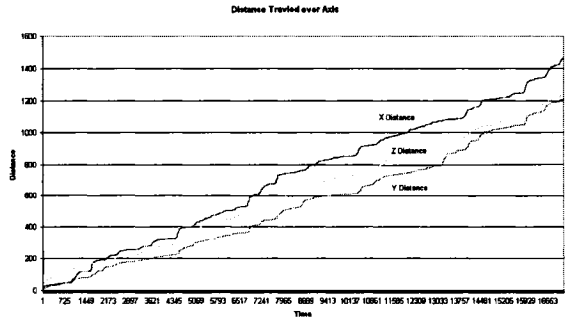
The last subject's (subject 3) performance can be viewed in graphs 4.16 to 4.18.



Graph 4.16 Distance Traveled, Cubes1, Subject 3



Graph 4.17 Distance Traveled, Cubes2, Subject 3



Graph 4.18 Distance Traveled, Cubes3, Subject 3

Table 4.12 summarizes the information in the graphs.

Table 4.12 Cubes Exercise for Three Trials for Subject 3 Showing Time Elapsed, Distance Covered along all Axes, and Average Hand Speed Attained along all Axes

Trial	Time (seconds)	Distance X (cm)	Distance Y (cm)	Distance Z (cm)	Average Speed over all Axes (cm per second)
1	20.08	1703	1138	1138	66
2	19.97	1556	1205	1262	67
3	17.17	1468	1210	1235	76

Using the last column to compute the average speed of subject 3 along all axes gives a value of around 70 cm per second. Recalling the values for total average speed for subjects 1 and 2 (121 and 66 cm per second), an average of these three values may give a therapist a target to set for a patient when performing the cubes exercise. But noticing that subject 1 showed a considerably higher value than subjects 2 and 3, the average of the latter two subjects could be a more realistic target: 68 cm per second.

Finally, we analyze subject 3's results in terms of difference between the greatest and lowest distance covered across all axes. Table 4.12 shows like subject 1 and trials 1 and 3 for subject 2, subject 3's hand covered more distance on the 'X' axis in all trials, with the 'Z' distance coming last in all those trials. The difference in distance between 'X' and 'Z' is shown in table 4.13 below. An average difference of 391 cm can be computed.

Table 4.13 Difference between Greatest and Lowest Distance for Cubes for Subject 3 for Three Trials

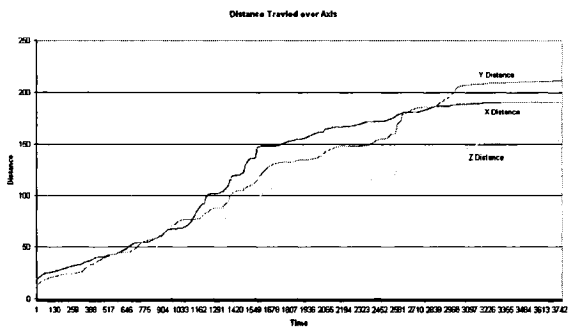
Trial	Difference between Greatest and Lowest Distance (cm)
1	565
2	351
3	258

Recalling the average for subjects 1 and 2 (410 and 285 cm respectively), a therapist can set a target of 362 cm for a recovering patient.

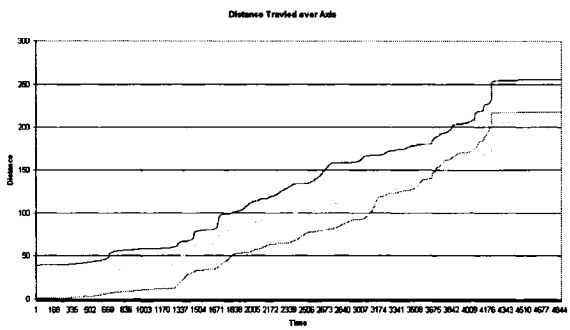
4.2.3 Maze Exercise

As this exercise restricts the subject to navigating within the narrow path of the maze, the distance covered is the least among all the exercises.

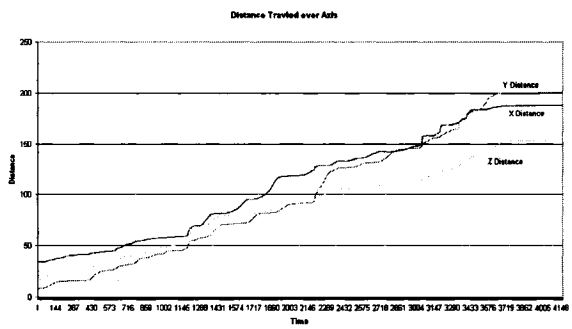
Graphs 4.19 to 4.21 show the distance covered along each axis for three trials for subject 1.



Graph 4.19 Distance Traveled, Maze1, Subject1



Graph 4.20 Distance Traveled, Maze2, Subject1



Graph 4.21 Distance Traveled, Maze3, Subject1

The results for subject 1 are shown in table 4.14.

Table 4.14 Maze Exercise for Three Trials for Subject 1 Showing Time Elapsed, Distance Covered along all Axes, and the Hand Speed Attained along all Axes

Trial	Time (seconds)	Distance X (cm)	Distance Y (cm)	Distance Z (cm)	Average Speed over all Axes (cm per s)
1	3.76	191	211	153	49
2	4.86	256	218	209	47
3	4.16	188	201	154	43

Since the maze is of a rectangular shape with the X dimension slightly greater than the Y, one would initially expert a higher X value for distance, but this is not true because the subjects had to move up and down the maze and hence covered a little more on the Y axis than the X.

The last column in table 4.14 is used to compute the average hand speed of subject along all axes: 46 cm per second. This value will be used later with the averages for subjects 2 and 3 to set a target for a therapist for a recovering patient performing this exercise.

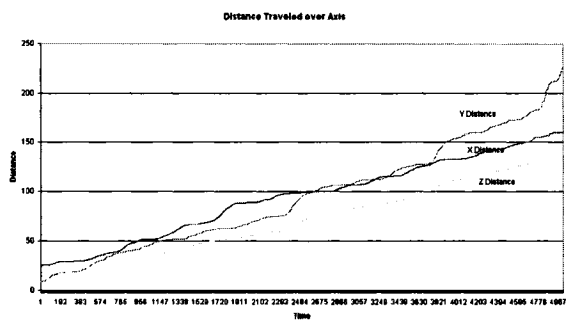
Since navigating the maze requires movement in the XY plane, any distance accumulated over the Z axis decreases the efficiency of performing the exercise. Table 4.15 shows 'Z' distance as Percentage of 'X' and 'Y' for subject 1 for all three trials.

Table 4.15 'Z' Distance for Maze Exercise as Percentage of 'X' and 'Y' Distances for Subject 1 for Three Trials

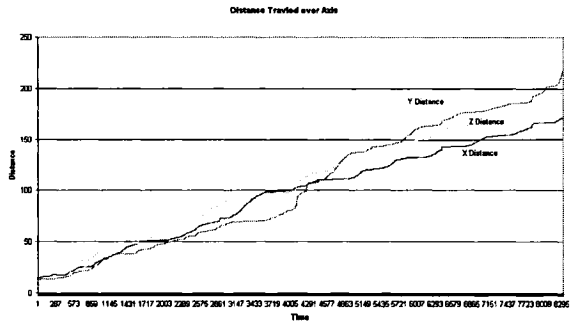
Trial	'Z' Distance (cm) as Percentage of 'X' Distance (cm)	'Z' Distance (cm) as Percentage 'Y' Distance (cm)
1	80.1	72.5
2	81.6	95.9
3	81.9	76.6
Average	81.2	81.7

An average of 81.2 % for 'Z' distance: 'X' distance and 81.7 % for 'Z' distance: 'Y' distance can be computed.

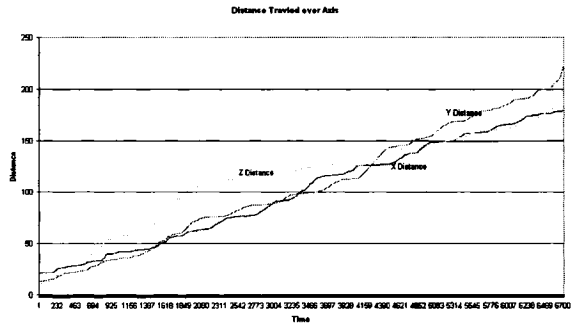
Graphs 4.22 to 4.24 show the results of subject 2 when performing the maze exercise.



Graph 4.22 Distance Traveled, Maze1, Subject2



Graph 4.23 Distance Traveled, Maze2, Subject2



Graph 4.24 Distance Traveled, Maze3, Subject2

Table 4.16 lists the results extracted from graphs 4.22 to 4.24.

Table 4.16 Maze Exercise for Three Trials for Subject 2 Showing Time Elapsed, Distance Covered along all Axes, and the Hand Speed Attained along all Axes

Trial	Time (seconds)	Distance X (cm)	Distance Y (cm)	Distance Z (cm)	Average Speed over all Axes (cm per s)
1	5.02	160	227	142	35
2	8.31	172	218	197	23
3	6.71	179	223	191	29

The average speed over all axes for subject 2 can be found by averaging the results in the last column of table 4.16: 29 cm per second; less than that for subject 1 (46 cm per second).

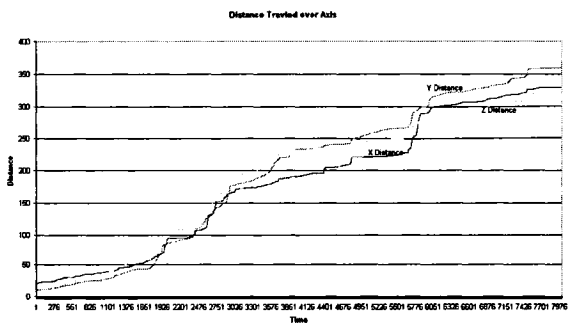
Looking at the 'Z' distance: 'X' distance and 'Z' distance: 'Y' distance ratios is easier when tabulating, as in table 4.17.

Table 4.17 'Z' Distance for Maze Exercise as Percentage of 'X' and 'Y' Distances for Subject 2 for Three Trials

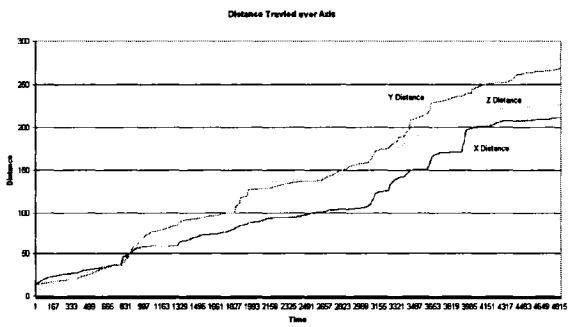
Trial	'Z' Distance as Percentage of 'X' Distance	'Z' Distance as Percentage 'Y' Distance
1	88.8	62.6
2	114.5	90.4
3	106.7	85.7
Average	103.3	79.6

An average of 103.3 % for 'Z' distance: 'X' distance and 79.6 % for 'Z' distance: 'Y' distance can be computed. Results from subject 3 can better determine a target to be set by a therapist.

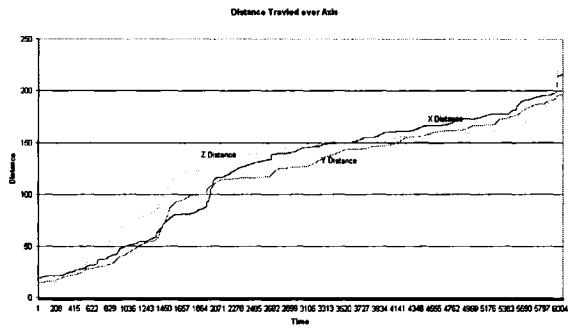
Graphs 4.25 to 4.27 show the performance of subject 3. Table 4.18 summarizes the performance.



Graph 4.25 Distance Traveled, Maze1, Subject3



Graph 4.26 Distance Traveled, Maze2, Subject3



Graph 4.27 Distance Traveled, Maze3, Subject3

Table 4.18 Maze Exercise for Three Trials for Subject 3 Showing Time Elapsed, Distance Covered along all Axes, and the Hand Speed Attained along all Axes

Trial	Time (seconds)	Distance X (cm)	Distance Y (cm)	Distance Z (cm)	Average Speed over all Axes (cm per s)
1	8.00	329	360	321	42
2	4.82	211	269	227	49
3	6.03	216	196	221	35

The average speed over all axes for subject 3 is computed to be 42 cm per second; less than that for subject 1 (46 cm per second), but more than that for subject 2 (29 cm per second). A therapist may set a target for a patient by taking the average of the three subjects: 39 cm per second.

Finally we look at the 'Z' ratio to 'X' and 'Y' distances in table 4.19 below.

Table 4.19 'Z' Distance for Maze Exercise as Percentage of 'X' and 'Y' Distances for Subject 3 for Three Trials

Trial	'Z' Distance as Percentage of 'X' Distance	'Z' Distance as Percentage 'Y' Distance
1	97.6	89.2
2	107.6	84.4
3	102.3	112.8
Average	102.5	95.47

As can be seen, the averages are 102.5 % and 95.47 % for 'Z' distance: 'X' distance and 'Z' distance: 'Y' distance respectively. Recalling those values for subjects 1 and 2 (81.2, 81.7 & 103.3, 79.6 respectively), a therapist may set a target of 95.67 % and 85.59 % for 'Z' distance: 'X' distance and 'Z' distance: 'Y' distance respectively.

This chapter showed a number of ways of looking at the data and analyzed it to extract information that can help a therapist better assess a patient's rehabilitation course. When applying the methods described here to analyze a patient's data, care should be taken to consider many factors; age, sex, and mood of the patient are important factors to be looked at. A patient in a bad psychological state can be less motivated to perform the exercises, thus affecting the accuracy of analysis.

CHAPTER 5

CONCLUSION

In this thesis, a haptic-based framework in a virtual reality environment has been developed and studied to help improve the hand rehabilitation process following a stroke. The framework included four components: sensory system, haptic software simulation system, client application, and haptic/behavioural data system.

The sensory component of the system, embedded within the haptic and visual interfaces, has been described. The CyberGrasp station including CyberGrasp, CyberForce, and CyberGlove has been presented and its limitations outlined.

The haptic software component that simulates the complex calculations involved in the haptic rendering process loop, maintains synchronization with graphic rendering, and records haptic behavioural data for further analysis has been presented. The problems faced during this phase of the framework have been explained.

The haptic/behavioural data system component that processes data files obtained from subjects' trials has been described. The data files showing several physical attributes that were recorded such as the time stamp that establishes the start and end of each exercise, the 3D world coordinates of the virtual position, and the joint angles for the middle phalange of each finger have been analyzed to extract additional information,

like the total distance covered and average velocity along the X, Y, and Z axis. The analysis has been depicted in the form of graphs and tables.

The client application that features three virtual reality exercises has been presented and the exercises used have been described. The exercises involved lifting a cup vertically, arranging a set of cubes according to a pre-determined pattern, and navigating a maze with a stick.

The results have been analyzed to provide therapists with information that may help them set targets for their recovering patients during the course of rehabilitation. For example, a patient at the end of recovery may be expected to attain a speed of 190 cm per seconds when performing the cup exercise, while exhibiting a deviation on the Z axis of around 25% of the vertical distance covered (on the Y axis). As for the patient's grip, a firm grip would see the patient display almost identical values for the time the fingers remain idle during the exercise. When performing the cubes exercise, a recovering patient aims to decrease the difference between the distance he/she accumulates on the 'X' axis and that that he/she accumulates on the 'Z' axis. A difference of around 360 cm has been found to be a target to be attained. As for the maze exercise, an average speed of 39 cm per second over all axes has been found to constitute a target for a recovering patient. Also for the maze exercise, a recovering patient would be expected to cover distance on the Z axis that is around 96% of that covered on the X axis and around 86 % on the Y axis at the end of rehabilitation.

The next step that follows after the effort that has been presented in this thesis would be to tackle the problems with the hardware and acquire latest technologies to start testing with post-stroke patients aiming to recover their hand functions. Care should be given to adapting patients to the CyberGrasp. Immersing patients into the virtual environment should be thorough and an individual patient's case should be studied to determine the right time to start performing the exercises and gathering data. Data will ideally be analyzed for each session and the rapidness of recovery monitored to determine the efficiency of performance and hence assess the effectiveness of the exercises in the

rehabilitation process. The resulting analysis will be stored in a database and will be accessible by the therapist to track the patient's case. The extensive data base that will materialize will allow the therapist to alter tasks in exercises intelligently according to his/her patient's performance. Comparing results of different patients will also be possible and is useful in giving the therapist a broader picture of the overall rehabilitation system and its effectiveness.

The described framework can be incorporated into a robotics system that trains muscles for the whole arm. Adjustments can be made to allow for arm sensors and actuators to provide force feedback.

Other areas like authentication can also benefit from the extensive data base formed. Information about the manner a certain person performs certain tasks can be combined together to deduce unique patterns that identify a person.

APPENDIX: CLASSES USED IN IMPLEMENTATION

HapticApp Class

This is the main class of the framework. It is responsible for creating the GUI and initializing the haptic interface and is quite simple to use. One must first create an instance of it, call the Init() function (which returns TRUE or FALSE depending on its success), add the objects, and then call the Start() function to start the simulation (collision detection). Afterwards, simply call Update() to update the rendering of the objects. It is important to note that all objects which will have a haptic presence must be added to the Haptic Scene Graph (HSG) before calling Start(). This is because when the collision engine is created, an optimized geometry of all objects in the haptic scene graph will be created to facilitate collision detection. Therefore, all objects must be added before creating this engine (adding objects will be covered shortly).

There are currently 6 minor functions which can be of use: toggleShowCollisionCount(), toggleShowStiffness(), toggleShowWeight(), showCollision(), showStiff(), showInertia(). The first three functions toggle the showCollisionCount, showStiffness, and showWeight flags respectively. The flags are initialized to "false". The last three functions take a bool parameter and set the appropriate flag, showCollisionCount, showStiffness, and showWeight flags respectively, to that bool value.

Adding objects can be done in multiple ways:

1- AddGraphicalObject() will add an object which will be rendered on-screen, but will not be added to the HSG. The function takes a pointer to an instance of IDrawableObject (more on this later).

2- AddToHaptics() will add an object which will have a haptic presence but will not be rendered on-screen. The function takes a pointer to an instance of IDrawableHapticObject (more on this later).

3- AddObject() has two parameters, the object and a bool value. If the bool value is "true", the object will be added to the haptic scene graph and be rendered on-screen; "false" will have the object rendered on-screen only. The function takes a pointer to an instance of IDrawableObject for the first parameter (the object), and if the second parameter (the bool value) is true, the object will be cast to an IDrawableHapticObject pointer.

Example of use

```
HapticApp app;

if( !app.Init())
{
  //Add error code here
}

//Create object here

//Add an object to the haptic scene graph and visual scene graph (on-
screen rendering)
app.AddObject(&object, true);

//Start simulation - Starts haptics collision detection
app.Start();

//Show the collision count, stiffness and weight. The last on toggles
//the flag to on because it is initialized to false.
app.showCollision(true);
app.showStiff(true);
app.toggleWeight();

while(!done)

{
      //Handle messages from windows
      if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
      {
            if(msg.message == WM_QUIT)
                  done = true;

            TranslateMessage(&msg);
            DispatchMessage(&msg);
      }
```

```
        else
        {
                //No messages to process- let's draw our scene
                app.Update()
        }
}
```

Note that the line of code:

```
app.AddObject(&object, true);
```

can be replaced with:

```
app.AddGraphicalObject(&object);
app.AddToHaptics(&object);
```

AppGUI class

This is an OpenGL GUI class. The user should not use this class. The HapticApp class takes care of creating an instance of this class and supplying the objects for rendering. However, it is quite simple to use if desired. One must first call Init(), which takes a char* as an argument (the title). Afterwards, when one wants to render a scene, one calls ClearScene() which clears the screen. Then one can render objects using RenderObjects() which takes a std::vector of IDrawableObject pointers as a parameter. Finally, FlipToScreen is called to swap the back buffer and forward buffer, thus drawing the image to the screen.

AppGUI also allows writing text to the screen. This function is DrawText() which takes a Cvector3 object as a parameter, which is the position to draw the text, followed by the text to be written to the screen. It should be noted that after the vector, the parameters are similar to those of sprintf. This means that it can take an infinite amount of text parameters and/or numbers. It should be noted that the formatting rules are the same as for sprintf. For example, if one wants to write "Age: 35" to the screen, the parameters would look like this: DrawText(position, "Age : %d", age);. To add variables to text, the proper %d, %f, ... must be added to the text.

AppGUI has a menu which is found in the AppGUIMenu.rc file. It is a simple menu that allows the user to set the filename for a data file, select when to start and end recording data, select skin tones, and quit the application. This menu also uses a dialog box with an edit box to get the file name. Messages for this dialog box are processed in NewFileDlgProc() which is found in AppGUI.cpp.

There is one function that is found in the AppGUI.cpp which should be of use to the user: AppGUIWndProc(). This function handles the windows messages for the window created in the AppGUI class, as well as the message related to the menu. This function should be modified to suit the application's needs.

Example of use

```
AppGUI gui;
std::vector<IDrawableObject*> objects;

if (!gui.Init("Application Title"))
{
      //Error code here
}

//create and add objects to list here

...

void renderScene()
{
      //Cvector3 positionVector(0.0, 0.0, 0.0);

      //clear the screen
      gui.ClearScene();

      //rendering code

      //render the list of objects
      gui.RenderObject(objects);

      //more rendering code

      //show FPS
      gui.DrawText(positionVector, "FPS: %f", fps);

      //Bring the image to the screen
      gui.FlipToScreen();
      }
```

Haptics class

This class is the main Haptics class. It wraps up the initialization of the haptic devices, updating the HSG, and holds an instance of the Simulation class. Like the GUI, HapticApp is responsible for creating this object (instance of the Simulation class) and calling its functions. To properly use this class, one must first call Init(), which returns TRUE or FALSE depending on its success. Afterwards, one must add objects to the HSG using AddToHSG() which takes an IDrawableHapticObject pointer as a parameter. After all objects have been added, the simulation must be started by calling CreateEngine(). To render the hand, RenderHand() should be called. RenderHand() takes a row-major matrix as a parameter, which should be the camera matrix. RenderContactPatches() and/or CollisionEngineRender() can also be called during rendering to render the contact patches and/or the collision information (bounding spheres, collision points), respectivly.

Methods are provided for updating and retrieving information. If there is movement in the visual scene graph (i.e. translation), UpdateHSG() should be called. This method updates the haptic scene graph. It currently takes a Cvector3 object representing the movement done. GetTransform() retrieves the haptic scene graph matrix of the object given in the parameter. getHand(), getRoot(),getTracker(), getGlove, getEngine(), and getSimulation() all retrieve a pointer to their respective haptic object. There is also a getComponent() which has an int as parameter, the int being the index of the component. This function returns a pointer to the component at the specified index. SetSkinColor(), which takes three int (red, green, blue) as parameters, sets the color of the hand. The default is 213,179,140. Finally, there are get/setStiffness(), get/setInertia(), to get or set the Stiffness and Inertia of the contact patch, and getOOCollisionCount() which will retrieve the amount of Object-Object collisions.

Example of use

```
Haptics haptic;

if (!haptics.Init())
{
      //error code here
```

```
}

//create object(s) here

//Add the object to the Haptic Scene Graph
haptic.AddToHSG(&object);

//Create the engine and start the Simulation
haptic.CreateEngine();

…

haptic.setStiffness(0.5f);
haptic.setInertia(10);
…

void RenderScene()
{
      //some rendering code

      //Get the transform of the object
      haptic.GetTransform(&object);

      //Render object here

      //Render hand, contact pactches, and collision info
      haptic.RenderHand(cameraMatrix);
      haptic.RenderContractPatches();
      haptic.CollisionEngineRender();

      //More rendering code

}
```

Simulation class

This class is extremely important for the haptics. Its main purpose is to create the collision engine and handle the collisions and is fairly simple to use. An instance of this class is created and passed on to the vhtEngine object. The constructor for the Simulation class takes a pointer to a Haptics instance because some methods require access to the haptics devices. The Haptics class takes care of this, as well as wrapping its member methods such as CollisionEngineRender(). The vhtEngine instance will take care of calling the function to handle the collisions.

This class also allows data to be written to a file (currently only tracker position, timestamp, and the proximal joint angle of each finger). To record data, one must simply

call CreateDataFile(), which takes a char* as a parameter (the filename without an extension). This function creates the file in TXT format. Then setRecordFlag(), which takes a bool as a parameter, must be called, giving "true" as a parameter to start writing and "false" to stop. It should be noted that if setRecordFlag() is called before CreateDataFile() or that no data file has been opened, setRecordFlag() will open one with the default name "data.txt". The user will be warned of this.

Example of use

```
//Create engine
vhtEngine* engine = new vhtEngine;

//Register the vhtHumanHand instance
engine->registerHand(hand);

//Create a simulation instance
Simulation* sim = new Simulation(this);

//Create a data file called RecordData.txt
sim->CreateDataFile("RecordData");

//Set the simulation
engine->useSimulation(sim);

//Start simulation (detecting collision)
engine->start();

//Start recording data
sim->setRecordFlag(true);

...

//Stop Recording data
sim->setRecordFlag(false);
```

IDrawableObject base class / interface

This is a base class / interface. Therefore, no instance of it can be created. Its purpose is to facilitate adding support for various object types (currently only 3DS objects are supported). To use the framework with a specific object type, one simply has to create a class that inherits from this class, implement GLRender() (to render the object in OpenGL), and set the "numShapes" variables. "numShapes" is an integer that stores the

amount of meshes, or mesh equivalents, in the object. The rest of the framework will take care of setting the index value and updating its transform matrix.

Example of use

```
class Object: public IDrawableObject
{
	// class variables and methods definitions here

	void GLRender()
	{
		//Render model in OpenGL here}
	}

	void LoadObject()
	{
		//load object here

		numShapes = numMeshes;
	}
}
```

IDrawableHapticObject base class / interface

This base class / interface is very similar to IDrawableObject. More precisely, it extends IDrawableObject. Its purpose is to facilitate adding support for different object types (currently only 3DS objects are supported) that will not only be drawn on-screen, but also have a haptic presence. Using this interface is quite simple. If the object type will be used for haptics, a class is created that inherits from IDrawableHapticObject instead of IDrawableObject. The method GLRender() is defined and the variable "numShapes" is set, then the method MapToHaptics() is implemented. MapToHaptics(), in simple terms, recreates the mesh into a vhtShape3D object, which it then returns. The vhtShape3D is an object used by the VHT library and cannot be changed. The framework takes care of setting the "componentID" variable.

Example of use

```
class Object: public IDrawableObject
{
	// class variables and methods definitions here
```

```
        void GLRender()
        {
                //Render model in OpenGL here}
        }

        void LoadObject()
        {
                //load object here

                numShapes = numMeshes;
        }

        vhtShape3D* MapToHaptics(int shape)
        {
                //copy mesh into vhtShape3D object
        }
}
```

C3DSModel class

This class loads and renders 3DS models. 3DS models can be created in modeling programs, thus not requiring the object to be written vertex by vertex in OpenGL. This class inherits from IDrawableHapticObject, therefore, any instance of this class can be used for haptics and be rendered on-screen. Using this class is very simple; an instance is created, Load3DS() is called, which takes the filename as a parameter and which returns TRUE or FALSE depending on its success. To render the model, GLRender() is called, which is what the framework (more precisely AppGUI) does in RenderObjects(). Unload3DS() can be called to unload the model, but this is done also by the destructor, so it is optional.

Note that since C3DSModel is rendered in OpenGL, the framework must be initialized before loading a 3DS file (since textures are created, which use OpenGL commands).

Example of use

```
//Init framework here

//Create object
C3DSModel object;

//Load 3DS file
object.Load3DS("aiplane.3ds");
```

```
//Add object to framework
app.AddObject(&object);

...

void RenderObjects()
{

      //Rendering code here

      //Render object
      objects.GLRender()
      //More code here
}
```

Cvector3 Class

This is a simple vector class. Its purpose is to perform vector math, and hold vector information. It should be noted that it is not a fully functional vector class; its methods were added as needed. To use it, an instance of it is created, and its method is used as needed.

Example of use

```
//Create 3 vectors
Cvector3 normal, vect1, vect2

//Set vect1
vect1.x = 1.0;
vect1.y = 2.0;
vect1.z = 3.0;

//Set vect2
vect2.x = 3.0;
vect2.y = 4.0;
vect2.z = 5.0;

//Calculate the normal by doing the cross product then
//normalizing the vector
normal = crossProduct(vect1,vect2);
normal.Normalize();
```

REFERENCES

[1] American Heart Association, "Heart disease and stroke statistics — 2005 update," Dallas, Texas. 2005.

[2] Avizzano, Solis, Frisoli, Bergamasco, "Motor Learning Skill Experiments Using Haptic Interface Capabilities," Proc. of IEEE International Workshop on Robot and Human Interactive Communication. Berlin, Germany. 2002.

[3] Bergamasco, Degl'Innocenti, Bucciarelli, "A Realistic Approach for Grasping and Moving Virtual Objects," Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems. Munich, Germany. 1994.

[4] Boian, Deutsch, Lee, Burdea, and Lewis, "Haptic Effects for Virtual Reality-Based Post-Stroke Rehabilitation," Proc. of 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. 2003.

[5] Boian, Sharma, Han, Merians, Burdea, Adamovich, Reece, Tremaine, and Poizner, "Virtual Reality-Based Post Stroke Hand Rehabilitation," Proc. of Medicine Meets Virtual Reality Conference. Newport Beach, CA. 2002.

[6] Broeren, Georgsson, Rydmark, and Sunnerhagen, "Virtual Reality in Stroke Rehabilitation with the Assistance of Haptics and Telemedicine," Proc. of 4th International Conference on Disability, Virtual Reality & Associated Technologies. Veszprém, Hungary. 2002.

[7] Burdea, "Key Note Address: Virtual Rehabilitation-Benefits and Challenges," 1st International Workshop on Virtual Reality Rehabilitation (Mental Health, Neurological, Physical, Vocational). Lausanne, Switzerland. 2002.

[8] Burdea, Popescu, Hentz, and Colbert, "Virtual Reality-Based Orthopaedic Telerehabilitation," IEEE Transactions on Rehabilitation Engineering. 2000.

[9] Connor, Wing, Humphreys, Bracewell, and Harvey, "Errorless Learning Using Haptic Guidance: Research in Cognitive Rehabilitation Following Stroke," Proc. of 4th International Conference on Disability, Virtual Reality & Associated Technologies. Veszprém, Hungary. 2002.

[10] Fagan, Matsuoka, and Klatzky, "Feedback Distortion for Rehabilitation: Gauging Perceived Physical Effort," Proceedings of 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. 2003.

[11] Glantz, Rizzo, and Graap, "Virtual Reality for Psychotherapy: Current Reality and Future Possibilities," Psychotherapy: Theory, Research, Practice, Training. 2003.

[12] Jack, Boian, Merians, Adamovich, Tremaine, Recce, Burdea, and Poizner, "A Virtual Reality-Based Exercise Program for Stroke Rehabilitation," Proc. of Fourth ACM SIGCAPH Conference on Assistive Technologies. Arlington, Virginia. 2000.

[13] Jack, Boian, Merians, Tremaine, Burdea, Adamovich, Recce, and Poizner, "Virtual Reality-Enhanced Stroke Rehabilitation," IEEE Transactions on Neural and Rehabilitation Engineering. 2001.

[14] Jebsen, Taylor, Trieschmann, Trotter, and Howard, "An Objective and Standardized Test of Hand Function," Archives of Physical Medicine and Rehabilitation, pp. 311-319. June 1969.

[15] Mathiowitz, Volland, Kashman, and Weber, "Adult Norms for the Box and Blocks Test of Manual Dexterity," The American Journal of Occupational Therapy, pp. 386-391. 1985.

[16] McLaughlin, Chen, Park, Zhu, and Yoon, "Recognizing User State from Haptic Data," International Conference on Cybernetics and Information Technologies, Systems, and Applications. Orlando, Florida. 2004.

[17] McLaughlin, Rizzo, Jung, Peng, Yeh, Zhu, and the USC/UT Consortium for Interdisciplinary Research, (accepted for presentation), "Haptics-Enhanced Virtual Environments for Stroke Rehabilitation," Proc. of the IPSI. Cambridge, MA. 2005.

[18] McLaughlin, Sukhatme, Peng, Zhu, and Parks, "Performance and Co-Presence in Heterogeneous Haptic Collaboration," Haptics Symposium, IEEE VR. Los Angeles, CA. 2003.

[19] MedicineNet.com, Definition of Neuroplasticity. http://www.medterms.com/script/main/art.asp?articlekey=40362. Last accessed May 1, 2006.

[20] Merians, Jack, Boian, Tremaine, Burdea, Adamovich, Reece, and Poizner, "Virtual Reality-Augmented Rehabilitation for Patients Following Stroke," Case Report, Journal of the American Physical Therapy Association, vol. 82, no. 9. Sep. 2002.

[21] Mirtich, "V-Clip: Fast and Robust Polyhedral Collision Detection," ACM Transactions on Graphics, 1998.

[22] Misra and Agah, "Design of a robotic exoskeleton arm for rehabilitation," Technical Report ITTC-FY2003-TR-28940-01, Information and Telecommunication Technology Center (ITTC). University of Kansas. Lawrence, Kansas. Nov. 2002.

[23] Nudo, "Neural Substrates for the Effects of Rehabilitative Training on Motor Recovery after Ischemic Infraction," Science, vol. 272, pp.1791-1794. 1996.

[24] Orozco, Asfaw, Adler, Shirmohammadi, El Saddik "Automatic Identification of Participants in Haptic Systems," IEEE IMTC, Ottawa, Ontario. 2005.

[25] Orozco, Shakra, and El Saddik, "Adaptive Haptic Framework," Proc. of IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems. Giardini Naxos, Italy. 2005.

[26] Orozco, Shakra, and El Saddik, "Haptic: The New Biometrics-embedded Media to Recognizing and Quantifying Human Patterns," Proc. of 13th Annual ACM International Conference on Multimedia. Singapore. 2005.

[27] Piron, Tonin, Trivello, Battistin, and Dam, "Motor Tele-Rehabilitation in Post-Stroke Patients," Medical Informatics and the Internet in Medicine. 2004.

[28] Popescu, Burdea, Bouzit, and Hentz, "A Virtual-Reality-Based Telerehabilitation System with Force Feedback," Information Technology in Biomedicine, IEEE Transactions, 2000.

[29] Prisco, Avizzano, Calcara, Ciancio, Pinna, and Bergamasco, "A Virtual Environment with Haptic Feedback for the Treatment of Motor Dexterity Disabilities," Proc. of the IEEE International Conference on Robotics & Automation. 1998.

[30] Qu, El Saddik, and Adler. "Dynamic Signature Verification System Using Stroke Based Features." MSc. thesis. University of Ottawa. Ottawa, ON. 2004.

[31] Reachin Technologies. http://www.reachin.se. Last accessed May 1, 2006.

[32] Rijken and Dekker, "Clinical Experiences of Rehabilitation Therapists with Chronic Diseases: A Quantitative Approach," Clinical Rehabilitation, vol. 12, pp. 143-150. 1998.

[33] Rizzo, Buckwalter, and Neumann, "Virtual reality and Cognitive Rehabilitation: A Brief Review of the Future," The Journal of Head Trauma Rehabilitation, 1997.

[34] Rizzo, McLaughlin, Jung, Peng, Yeh, Zhu, and the USC/UT Consortium for Interdisciplinary Research, "Virtual Therapeutic Environments with Haptics: An Interdisciplinary Approach for Developing Post-Stroke Rehabilitation Systems," Proceedings of International Conference on Computers for People with Special Need. 2005.

[35] Rizzo, Schultheis, Kerns, and Mateer, "Analysis of Assets for Virtual Reality Applications in Neuropsychology," Neuropsychological Rehabilitation, 2004.

[36] Schuyler, Mahoney, and Hove, "Standardized Assessment Tests for the Human Robot System," Proc. of the 5th International Conference on Rehabilitation Robotics. 1997.

[37] Sensable Technologies. http://www.sensable.com. Last accessed May 1, 2006.

[38] Shakra, Orozco, El Saddik, Shirmohammadi, and Lemaire, "Haptic Instrumentation for Physical Rehabilitation of Stroke Patients," Proc. IEEE Workshop on Medical Measurement and Applications. Benevento, Italy. 2006.

[39] Shakra, Orozco, El Saddik, Shirmohammadi, and Lemaire, "VR-Based Hand Rehabilitation Using a Haptic-Based Framework," Proc. IEEE Instrumentation and Measurement Conference. Sorrento, Italy. 2006.

[40] Stereo3DTM. http://www.stereo3d.com. Last accessed May 1, 2006.

[41] Sveistrup, "Motor Rehabilitation Using Virtual Reality," Journal of Neuro-Engineering and Rehabilitation. 2004.

[42] Taub, Uswatte, and Morris, "Improved Motor Recovery after Stroke and Massive Cortical Reorganization Following Constraint-Induced Movement Therapy," Physical Medicine and Rehabilitation. Clinics of North America. 2003.

[43] USC/UT Exploratory Center for the Interdisciplinary Study of Neuroplasticity and Stroke Rehabilitation. http://www.usc.edu/schools/medicine/departments/cell_neurobiology/research/isnsr. Last accessed May 1, 2006.

[44] Van Den Bergen, "Collision Detection in Interactive 3D Environments," Playlogic Game Factory, Breda, the Netherlands. 2003.

[45] Virtual Technologies, INC, 2000-2001. CyberForce: Reference Manual, version 1.31.

[46] Virtual Technologies, INC, 2001. Virtualhand: User and Programmer Guides, version 2.5.

[47] Whalen, Petriu, Yang, Petriu, and Cordea, "Capturing Behaviour for the Use of Avatars in Virtual Environments," CyberPsychology and Behaviour. 2003.

[48] Yamaguchi, "The hyper hospital - virtual reality based medical system on the computer network- its concept and user-configurable virtual world creating system," Technical report, Department of Bio-Medical Engineering, School of High Technology for Human Welfare. Tokai University. Japan. 1996.

[49] Yang, Petriu, Whalen, and Petriu, "Hierarchical Animation Control of Avatars in 3-D Virtual Environments," IEEE Transactions on Instrumentation and Measurement. 2005.

[50] Youdin, Sturm, and Youdin, "Understanding Rehabilitation Engineering," 1988, Proceedings of the Fourteenth Annual Northeast Bioengineering Conference. 1988.

[51] Zhang, Beatriz, Seale, Masel, Christiansen, and Ottenbacher, "A Virtual Reality Environment for Evaluation of a Daily Living Skill in Brain Injury Rehabilitation: Reliability and Validity," Archives of Physical Medicine Rehabilitation, vol. 84, no. 8, pp. 1118-24. August 2003.

[52] Zhou, Shen, Shakra, El Saddik, and Georganas, "XML-based Representation of Haptic Information," Proc. of the fourth IEEE International Workshop on Haptic Virtual Environments and their Applications. Ottawa, ON. 2005.

[53] Zimand, Anderson, Gershon, Graap, Hodges, and Rothbaum, "Virtual Reality Therapy: Innovative Treatment for Anxiety Disorders," Primary Psychiatry. 2003.