ABSTRACT

A VIRTUAL TRAINING ENVIRONMENT FOR WHEELCHAIR MOUNTED ROBOT

by
Elizabeth Leichtnam

A model of a robotic manipulator designed to respond to different inputs from users with different skills or conditions is presented. This model can be adapted to different input variables and converted into joint angles which relate to movement in space of the manipulator's links. The algorithm allows for real time response and the features of the virtual environments (i.e., 3D design, stereoscopic vision) gives the designers the opportunity to use this model for training and usability evaluation according to the skills and different conditions that users could present.

Two types of virtual environments were created aiming to be used as training tools for the user in the operation of the controllers. The first type is intended to develop complex daily tasks; the second one was designed to be used as an evaluation of usability. A relationship between the input given by the human motor system of the potential user and the response obtained from the model is evaluated by using Fitts' Law. Several works have been written using it as a model of prediction of human performance for Human-Computer Interaction. This work uses structured virtual environments where the variables required for Fitts' Law application are known and controlled. The evaluation of the response of the manipulator to the inputs from the operator have given information about the feasibility of controlling the movements of the robot according to the skills of the user.

A VIRTUAL TRAINING ENVIRONMENT FOR WHEELCHAIR MOUNTED ROBOT

by
Elizabeth Leichtnam

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Biomedical Engineering

Department of Biomedical Engineering

May 2008

APPROVAL PAGE

A VIRTUAL TRAINING ENVIRONMENT FOR WHEELCHAIR MOUNTED ROBOT

Elizabeth Leichtnam

4/30/08

Dr. Richard Foulds PhD, Thesis Advisor — Date
Associate Professor of Biomedical Engineering, NJIT

4/30/08

Dr. Sergei Adamovich PhD, Committee Member — Date
Associate Professor of Biomedical Engineering, NJIT

4/30/08

Dr. Bruno Mantilla M.D., PhD, Committee Member — Date
Special Lecturer of Biomedical Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Elizabeth Leichtnam

Degree: Master of Science

Date: May 2008

Undergraduate and Graduate Education:

- Master of Science in Biomedical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2008

- Bachelor of Science in Biomedical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2006

Major: Biomedical Engineering

To my mom for believing I could do anything

ACKNOWLEDGMENT

I would like to express my sincerest gratitude and appreciation to my advisor, Dr. Richard Foulds, who provided invaluable guidance throughout my undergraduate and graduate studies at New Jersey Institute of Technology. I would also like to thank my committee members, Dr. Sergei Adamovich and Dr. Bruno Mantilla for their assistance. I would also like to recognize my family. Without their continued love and support, I could not have reached the goals I have today. Thank you to all my fellow colleagues in the lab for their support and indispensable knowledge throughout my endeavor. Lastly, I would like to thank the National Institute on Disabilities and Rehabilitative Research for their funding through the Rehabilitation Engineering Research Center Grant #H133E050011-06. Thank you.

TABLE OF CONTENTS

Chapter	Page
1 BACKGROUND	1
1.1 Introduction	1
1.2 Motivation	1
1.3 Assistive Robotics	2
1.4 Wheelchair Mounted Robotics	5
1.5 Virtual Reality	8
1.6 Stereoscopic Vision	9
1.7 Fitts' Law	11
1.8 Functional Testing	14
2 SOFTWARE AND SETUP	16
2.1 Interfaces	17
2.1.1 Immersion Probe and Personal Digitizer	17
2.1.2 Spaceball® 5000	18
2.2 Software Applications	20
2.2.1 Virtual Reality Toolbox	20
2.2.2 VRML	20
2.2.3 Virtual Reality Editors	21
2.2.4 Virtual Reality Viewers	22
2.2.5 Robotics Toolbox	24
3 DESIGN	25
3.1 World Building	25

TABLE OF CONTENTS
(Continued)

Chapter	Page
3.2 Control Algorithms	29
3.2.1 Collision Detection	29
3.2.2 Object Rotation	36
3.2.3 Toggle Control	37
3.2.4 Wheelchair Movement	39
4 IMPLEMENTATION	41
4.1 Experimental Design	41
4.2 Data Collection	45
4.3 Results	46
5 DISCUSSION AND FUTURE WORK	55
APPENDIX A SAMPLE TRAJECTORIES FROM FITTS LAW TESTING	57
APPENDIX B SAMPLE TRAJECTORIES FROM BOX AND BLOCKS TESTING.	61
APPENDIX C SCREEN SHOTS FROM VR WORLDS	64
REFERENCES	66

LIST OF TABLES

Table		Page
4.1	Analysis of Variance for MT for all input devices.....	47
4.2	Mean Times per Task and ID's.....	47

LIST OF FIGURES

Figure		Page
2.1	Immersion Probe	17
2.2	Spaceball and Directions of Force	19
2.3	User interface for Spaceball	19
3.1	Coordinate Systems For Graphical Objects In Hierarchy	26
3.2	Robotics Tollbox Robot and Coordinate System	32
3.3	Transformation from VRML to Matlab Coordinates	33
3.4	Simulink Code For Moving Many Objects	35
3.5	Virtual World Where Robot Can Move Chess Pieces	36
4.1	Orientations for Fitts' Law Tapping Experiments	44
4.2	Interval Plot for Fitts' Law of MT over Condition, ID, and Task	48
4.3	Interval Plot for Fitts' Law of MT over Task Only	49
4.4	Interval Plot for BBT of Task Only	53
A.1	X, Y, Z Plot of Finger Movement	57
A.2	X, Y, Z Plot of Stylus	57
A.3	X, Y, Z Plot of VR Stylus	57
A.4	X, Y, Z Plot of Spaceball	57
A.5	X, Y Plot of Finger	58
A.6	X, Y Plot of Stylus	58
A.7	X, Y Plot of VR Stylus	58
A.8	X, Y Plot of Spaceball	58

LIST OF FIGURES
(Continued)

Figure		Page
A.9	X, Y, Z vs. Time of Finger	59
A.10	X, Y, Z vs. Time of Stylus	59
A.11	X, Y, Z vs. Time of VR Stylus	59
A.12	X, Y, Z vs. Time of Spaceball	59
B.1	X, Y, Z Plot of Finger Movement	60
B.2	X, Y, Z Plot of VR Stylus	60
B.3	X, Y, Z Plot of Spaceball	60
C.1	Virtual Kitchen with Movable Utensils and Jell-o Cubes	61
C.2	Virtual Office with Computer, Desk, etc	61
C.3	Sample Fitts' Law World Used for Experiments	62
C.4	Virtual Box and Blocks World used for Experiments	62

CHAPTER 1

BACKGROUND

1.1 Introduction

Robotics as Assistive Technology is in need of improvements before mass market appeal. The excessive price and limited usefulness of most assistive robots are two glaring factors in the limited success of assistive robotics. Wheelchair mounted robots offer a good option for those whose abilities make it necessary for nursing aid and/or robotic assistance. They offer independence in unstructured environments but are sometimes too slow. The expense of these systems and the danger of a large machine working so closely with humans, and especially children, motivated the development of a three dimensional virtual training system. The design of this system is presented in conjunction with an evaluation of the system using Fitts' Law to verify the design and aid in future redesigns of algorithms for the training system and new interfaces.

1.2 Motivation

There are many conditions and situations where a person's motor ability may be severely diminished either permanently or temporarily. The New Jersey Institute of Technology Rehabilitation Engineering Research Center (NJIT RERC) on Technology for Children with Orthopedic Disabilities identified as its goal to improve the quality of life of children with cerebral palsy, muscular dystrophies, arthrogryposis, contracture due to burns, congenital and traumatic amputations, polio and bone diseases. These conditions can require the use of wheelchairs and can also significantly impair the ability of their

arms. These impairments can be accompanied by a limited range of motion, diminished strength, and loss of coordination. If these conditions reduce a person's ability to manipulate objects and perform necessary daily tasks, the options of lifestyle are limited. Current available options consist of assistive technologies or near fulltime nursing aid. For several individuals in this situation many of the existing assistive technologies lack in practicality and usability. The technology is either not real-world-ready, the device is too simplistic or only conditionally useful, or the use is too slow to be practical in everyday circumstances. Because of these reasons, many endure the cost of nursing aid and the negative impact on their quality of life due to the lack of their independence.

1.3 Assistive Robotics

Assistive rehabilitation robotic technology is categorized by the three words themselves. The definition of robotics differs from source to source but can be narrowed to two main aspects, programmability and multi-functionality. The assistive or rehabilitation aspect's main aim is to improve function. So any device that is programmable and improves function can be considered an assistive robot. The assistive robots of the last thirty years can be grouped into several categories: Fixed site or workstation robots, task specific robots such as power feeders, mobile assistive robots, wheelchair-mounted manipulators, orthotics, therapeutic robots, and educational robots (Hillman, 2003). These types of robots can again be grouped into the type of function of which the device seeks to restore, physical or social. Orthotics, therapeutic robots, and educational robots seek to restore the actual physical function of a person where as workstation robots, power feeders, mobile robots, and wheelchair mounted robots seek to give the user the ability to perform

certain tasks that they would otherwise not be able to perform on their own, which provides improved social functionality by means of increasing their independence. Hillman (2003) argues that the impact of any of these technologies is only significant if they become commercially available and successful. Devices that improve social functionality in the past have been most successful by this definition. Cost and functionality of these devices seem to be the driving factors of success of these devices where they usually sacrifice one factor for the other.

The simplest and least expensive technology in this field is the task or multitask specific robot which sacrifices function for form and cost. These devices come in the form of power feeders or grooming machines. The disadvantages to this type of system are obvious. Not only do these devices serve one or only a few functions, but they also require quite a great deal of preparatory work by the care giver to set up. The care giver must prepare the food into tiny bits and place them in the tray or place the grooming devices in their proper places (Topping et al, 1999). They must also place the user and robot in working positions. The major selling point to these devices is a significantly reduced price in comparison with other assistive robots available to the disabled public. This technology is not completely dismissible. For those with severe physical disability, their opinion of their own quality of life might be greatly improved due to the ability to do at least one task on their own without the reliance of their care giver (Topping et al, 1998). The labor saving ability of these devices and the independence brought to their users is questionable, but the value in giving the user independence in one or a few tasks has had a convincing effect on the users' self-esteem.

The concept of this technology can be extended from a task independence into a situational independence. Where power feeders provide task independence, workstation robots provide situational or environmental independence. With workstation robots the system is fixed and can only assist the user in the fixed environment. The environment must be semi-structured (Van der Loos et al, 1998), meaning the system can not handle any situation and any variable that presents itself. Independence and assistance is only provided to the disabled user when they are positioned inside the room in which the robot is mounted. The potential to the improvement of the quality of life and self-esteem is immense. The implementation of this technology in a person's life might enable them to return to work or enjoy a hobby that they thought was impossible. Although this restoration of function is impressive and appreciated if adequate to the user, it does not maximally restore social function where other tasks or environments are concerned.

Mobile robots are another type of assistive robotic technology that has gotten quite a bit of research attention in the past few decades. Mobile robots are usually designed to be capable of autonomous operation in the unstructured environment. They usually sacrifice manipulability for mobility, where task specific and workstation robots sacrifice mobility for efficacy of manipulation in specific situations. Mobile robots usually employ vision systems, and/or a complex series of integrated sensors that help the robot chose its path and avoid obstacles (Yoshiyuki et al, 2003). These components and the complex algorithms created to use them to control the robot add immensely to the final cost to the user. The main goal of these robots is to reduce the amount of assistance from a human care giver required by the handicapped user. The fetch and carry use limits the ability of the system to give maximal independence to the user. The user

remains still while the robot identifies an object and brings it to the fixed position of the user. Therefore the user must remain fixed and wait for the robot to carry out its predefined commands. This fetch and carry approach also assumes that once the object is brought to the user that he or she has enough ability to use or manipulate on their own the object that was brought to them. For a user who is too disabled to use a wheelchair to navigate, it is unlikely that he/she will have the ability to manipulate what ever is carried to them by the robot. Along with expense and impracticality of the time of delivery, the technology is also lacking. These systems are not usually an out of the box system. It not only requires a great deal of customization in the programming of the algorithms of the robot but occasionally the rooms or buildings also need to be outfitted with markers (Evans, 1994) and if using a vision system the house/rooms need to be plainly decorated so that the robot's vision system is not confused. These are not trivial inconveniences and can cost a great deal in labor, time and disturbances to daily life. This chain of assumptions made in the design of these types of robots and the limitations of their use restricts the population and circumstances where these robots will be useful and worth the investment of time and money.

1.4 Wheelchair Mounted robots

The wheelchair mounted robot design idea attempts to bridge the gap between mobility and manipulation by involving the user. This type of robot is technically a telemanipulator which means it relies solely on the user for its commands to move and generate task goals. A group from the Netherlands developed and released on the market a wheelchair-mounted assistive robot called the Manus arm. It is a six degree of freedom

robot with a gripper which operates in the direct space of the disabled end user. The six degrees of freedom allow the robot to achieve any position and orientation in the space of the robot which does not restrict the activities or space in which the robot will function properly. The Manus allows for a human-like range of motion and acts somewhat as a third arm. The Manus has no complex system of sensors in which it tries to obtain information about its environment or the task performance at hand (Driessen et al, 2001). It relies on the user as the sensory information so that it can operate in an unstructured world. Some robotic systems try to define the environment in which it operates, which limits the use of the robot and the independence of the user. Other systems use complex sensory systems which are not only expensive but have flaws and complications that do not have solutions as of yet.

The Manus is not perfect though, by any means. When designing a device specifically for the physically disabled it is important to keep in mind how a device is going to interface with a population with not only a special need but also vastly varied abilities. Creating an interface that controls all six degrees of freedom and making it manageable by users with varying mobility is a difficult task. The Manus design team focused mainly on developing interfaces that accommodate different abilities and not those that alleviate the difficulty of operating the 6+1 degrees of freedom of the robot. When delegating the responsibility of the movement of the robot to the user the designer should make every effort to ease that burden. The interfaces available for the Manus include a keypad, a two degree of freedom joystick, a foot peddle, etc. which force the user to switch between several modes to control all joints and motions of the robot (Driessen et al, 2001) which can be a long and arduous task. One of the main advantages

to specifically the Manus robotic arm over other available wheelchair mounted robots, is that it comes with software and settings that make interfacing and control of the robot adaptable and updateable so that if a new device or technology is found to be more effective, it is possible to provide it to the end user. For these reasons, and after review of existing technologies, the NJIT RERC on Technology for Children with Orthopedic Disabilities embarked on a task of improving the usefulness and marketability of the Manus Arm.

The approach to this includes the development of new human-machine interfaces and the conversion of existing devices to perform unstructured tasks in a near real-time manner, as well as a virtual reality-based training environment. Before implementing the control mechanisms in the actual manipulator, these mechanisms are designed and tested in a 3D virtual environment in a scaled model from the actual one; this virtual environment is improved by featuring stereoscopic perception through a pair of special goggles combined with a software/hardware arrangement. The 3D model is necessary to aid the clinicians and users determine which input interfacing device best suits the user's requirements and conditions since each of the users has their own unique level of ability and their own preferences. Offering multiple interfaces that can operate multiple degrees of freedom at once and in a time efficient manner is an important aspect of the system. Also, operating multiple degrees of freedom at once can be confusing and the learning curve is possibly quite steep. Therefore, the 3D environments were created for the users to familiarize themselves with their new assistant. This will be a safer and less costly alternative to training patients directly to operate the Manus ARM. Training before using

the Manus ARM is necessary because there is risk of damaging a very expensive piece of equipment and a possibility for the user to harm themselves or people around them.

1.5 Virtual Reality

The term virtual reality refers to computer displays that are interactive and give the illusion of being in another location. The techniques and technology of modern evolved from vehicle simulation technology of the 1960's (Ellis, S.R. 1994). Today VR is used for laparoscopic surgical training, teleoperation, planetary surface visualization, and advanced gaming. Media in the early stages of the technology gave the erroneous illusion that the technology was more advanced and realistic than it actually is. Although modern applications of virtual reality employ very advanced technologies such as haptics, and highly advanced visual and auditory techniques, the scope of this project does not include them. Instead virtual reality is used as a tool to create a system conducive for learning how to operate a specific wheelchair mounted robot under realistic conditions.

Virtual reality usually refers to and implies an immersive 3D experience. For the purposes of this document virtual reality, virtual worlds, and virtual environments all refer to the same entity, which is the creation of a world or environment in which a person can see and interact with the world itself or objects placed inside of it but is not completely immersive. The technology created/used is most aptly comparable to that of a video game, where input from a user is detected and employed to guide the visual output. Although, the dissimilar aspect of the type of virtual reality used as compared to normal computerized gaming, is the visual representation is usually first person and the environment is not story or goal directed. The tools used to create the necessary visual

representation of the robot and the objects with which it interacts are the Virtual Reality Modeling Language or VRML, 3D modeling software which can create 3D objects in the VRML file type (.wrl) such as VRealm Builder®, 3D Studio Max®, ProEngineer®, SolidWorks® etc., Matlab® and Simulink® for the creation of control algorithms, Internet Explorer® (IE) with an IE plug-in such as Blaxxun Contact® for VRML viewing in an IE window, and a computer with a graphics card capable of 3D stereoscopic viewing.

1.6 Stereoscopic Vision

Stereoscopic vision is any technique capable of producing three-dimensional visual information or creating the illusion of depth in an image. It can most easily be described as the three dimensional viewing similar to what one can experience in movie theaters and Imax around the globe when the show is advertised in "3D". Although the three dimensional image is created by a different technique for movies and other media, the principal is similar. The illusion of depth is created by presenting each eye with different components or perspectives of the same image and letting the brain reconcile the two into the third dimension. There are several techniques for creating a three dimensional image including red-blue anaglyph, linear polarization, circular polarization, and shutter glasses. Shutter glasses are used to create the 3D image for the environments described in this document. They are made of a material that contains a polarizing filter which becomes dark when voltage is applied, but is otherwise transparent. These glasses are connected to the computer's video card through a sync. The sync causes the glasses to darken over one eye, and then the other in synchronization with the refresh rate of the monitor. The

monitor, likewise, alternately displays different perspectives for each eye synchronized with the eye which is given the transparent side of the eyeglasses. The setup employed uses a CRT monitor with a refresh rate of 120 Hz, meaning each eye is exposed to 60 Hz so that image flicker is difficult to detect and image ghosting is kept to a minimum.

The purpose behind creating a virtual environment with 3D stereoscopic capabilities is to mimic as closely as possible real situations. If the purpose of the system is to train individuals for real world situations, and tasks in the real world require manipulation in three dimensions, then the training simulation should reproduce those tasks in a virtual world that not only has those three dimensions but are perceivable and are not just mathematically in existence. The stereoscopic display technology allows you to place objects at different depths and allows those depths to be perceived inside and outside of the plane of the monitor. Realism is not the only benefit to a stereoscopic display. Research done in 1989 (Drascic et al) showed that with simple tasks, there is no learning curve associated with a stereoscopic display. After repetition of the task, subjects performed the task with the same level of proficiency at the first and last trials meaning that stereo assisted their movements so that no improvement was necessary. The experiment also showed that the proficiency of the task with monoscopic vision approached the proficiency of stereoscopic vision with repetition. The same study performed another experiment with a Fitts' Law paradigm. Using Fitts' Law to change the difficulty of a task, they found that stereoscopic vision was more beneficial when the difficulty of the task was increased. For simpler tasks the difference between the two displays is less noticeable. The real world requires manipulation in all degrees of freedom and complex motions in combination. The difficulty of the studied tasks is

small and measurable in comparison to real world tasks. Using this principal provides for the logic that stereoscopic vision would allow for users to perform tasks more proficiently at the onset of use, contrary to monoscopic where some instructional and training time would be required for complex real world tasks. Eliminating this time allows for users to transfer more quickly to the ultimate goal of control of the real wheelchair mounted robot. These claims were supported in 1991 by Draper et al. Their experimental set-up differed from the first study but confirmed that stereoscopic vision allowed for tasks with higher difficulty (Fitts' Law index of difficulty) to be completed at faster rates. They also performed an experiment measuring the length of time it took for subjects to complete an unstructured, more realistic tasks. They found that on average the task was performed 65% faster using a stereoscopic vision system over a monoscopic view. Due to these results, stereoscopic vision was determined to be an important and necessary aspect of the training system, because decreased task time and learning curve allow users to train on the system with minimal frustration and maximal success. Also, training time could focus on learning the how input device(s) control the movement of the robotic manipulator in all degrees of freedom.

1.7 Fitts' Law

Fitts' Law was developed in 1954 as an extrapolation of information theory. Information theory quantifies the capacity of the transmission of information, measured in bits, through a non-ideal channel perturbed by noise. Paul Fitts drew an analogy of the transmission of data though a medium such as a copper wire to the movement of information through the human motor system. The Fitts experiments were aimed at

evaluating the behavior of the receptor-neural-effecter system of the human motor system in developing specific repetitive tasks, tapping, disc transfer, and pin transfer (Fitts, 1954). The experiments yielded a measurement of information transmission capacity of human motor system in controlling amplitude of movement. As a task becomes more difficult the time taken to execute the task increases, in addition, as more accuracy is demanded the time taken to execute the task also increases (Fitts, Peterson, 1964). The mathematical formula for this law is most commonly found as:

$C = \frac{ID}{MT}$ Where C is the capacity of the human neuromuscular system

ID is the index of difficulty

MT is the mean movement time.

$ID = \log_2 \frac{2A}{W}$ A is the amplitude of distance moved

W is the target width

and linear regression produces

MT = a +b*ID where a and b are arbitrary task based constants

Since Fitts' first paper was published many researchers have tried to improve the model's accuracy, to account for nonzero intercepts, or to account for low values of ID. Many formulations have been published which try to account for these problems with Fitts' original formulation. These competing models have used different dimensional variables such as effective width and effective amplitude. They have taken into account approach angle when the experiment is expanded into a second dimension. Other models have used linear or power functions to describe the phenomena. Among all these models Fitts' original equation describes best what physically happens and keeps intact the analogy of

information transmission through the human neuromuscular system with the exception of Shannon's formulation (MacKenzie, I. S., 1992).

$$MT = a + b \log_2 \frac{A+W}{W}$$

Shannon's formulation is only significantly different from Fitts' Law when the ID approaches zero. Although this exception is more accurate in the low region of ID bits, it should be noted that the original information theory equation was only intended for situations with high signal to noise ratio, and that assumption carried over to Fitts' Law is that it should only be used for larger values of ID.

Fitts' Law was verified as a tool in evaluating tasks involved with human-computer interaction. MacKenzie (1992) did an across study comparison of six different employments of Fitts' Law with computer input devices. These studies compared mice, isometric and displacement joysticks, trackballs, touch pads, eye tracker, and foot pedal. Because of the diverse and large amount of variations introduced within and between studies a consistent model was not found. Although a model predicting performance in human-computer interfaces is not concrete, recommendations for experimental design were given. MacKenzie postulates that more reliable within study results can be obtained by "adopting a wide and representative range of A-W conditions" and "adopting the Fitts paradigm for serial task or discrete tasks offers the benefit of a simple experimental setup and invites access to a large body of past research." Therefore, the Fitts paradigm was adapted for the verification of the design of the virtual world. Similar experiments to those done for mice and joysticks will be performed inside the virtual 3D world. The major difference between these experiments and the setup described in this document is that the human movement is not directly correlated to the virtual movement. Subjects

are asked to use the isometric force controller or a large position controller to displace the end-effecter of the virtual model of the wheelchair mounted robot, and reach specific targets located in specific points in space (Exact experimental design described in Chapter 4). These experiments will help identify any improvements that need to be made to the algorithms, and to also assist in determining which interface is most suitable for a disabled user's specific needs.

1.8 Functional Testing

Fitts' Law has been used as a way of quantifying human motion of limbs by describing it as information transfer through the neuromuscular system. Previous Fitts' Law studies have used only healthy subjects for the purpose of assessment of the model. This study proposes the use of Fitts' paradigm as a tool for comparison between healthy and disabled subjects. Because the design of the system is aimed at giving function to those who have little to no motor function in their upper limbs the goal is to make a comparison to their physical abilities and the effective function that can be restored due to the implementation /use of the input systems and the wheelchair mounted robot. There are many qualitative and quantitative measures of physical upper extremity being used by physical therapists and researchers around the world. Many of the published general functional tests are listed below.

- the Minnesota Rate of Manipulation (MRM) test
- the Upper Extremity Function Test (UEFT)
- the Purdue Pegboard test
- the Jebsen test of hand function
- the Nine-Hole Peg test
- the Smith hand function evaluation
- the Box and Block Test (BBT)

- the Physical Capacities Evaluation of Hand Skill (PCE)
- the Action Research Arm (ARA) test
- the Sollerman hand function test
- the Standardized Object Test (SOT)

For a more extensive review of these and other such tests see van Tuijl et al, 2002.

These tests are aimed at giving a measure to the function that exists with a person with less than healthy function in their upper extremities. They are most often employed to ascertain whether or not function has been gained after some therapeutic intervention. Most of these tests include both gross and fine motor movements. These tests consist of tasks like picking and placing objects of different sizes and weights, placing pegs in holes, stacking objects, writing, and other simulated activities of daily living. Out of these and other functional tests the Box and Blocks Test (BBT) was chosen as a simple test that could be used as an experiment to compare the real world with the virtual worlds. It was chosen for several reasons, first because the Test apparatus is available to the NJIT RERC, it is easy to perform and takes very little time, published normative data is available on healthy adults, and finally because the virtual version was easy to design and render. The Box and Block test is made up of a larger wooden box with a partition directly in the centre creating two equal sides. A number of small multi colored wooden blocks are placed in one side of the box. The subject being tested is required to use the dominant hand to grasp one block at a time and transport it over the partition and release it into the opposite side. The subject is given 60 seconds in which to complete the test, and the score of the test is reported as the number of blocks transported to the other side. This test was originally developed to evaluate the gross manual dexterity of adults with cerebral palsy but has since been used on children and patients with other disorders (Mathiowetz et al, 1985).

CHAPTER 2

SETUP, REQUIREMENTS, AND PREVIOUS WORK

As of now two input devices are implemented with the 3D training system and two more are in development. The first is a position sensing six degree of freedom stylus called the Immersion Probe®. The other operational device is an isometric force input ball called the Spacemouse 5000® that interprets forces and torques in six degrees of freedom at once. There are two more devices that can be used as inputs and which interfaces are in development. The first one is the Flock of Birds® which uses a magnetic field in a transmitter and sensor system to attain position and orientation in space (6 DOF). The other one is a simpler three degree of freedom position recognition which implements accelerometers. The development of these devices is important because it allows the ability to cater to the specific special needs of our intended users.

Whichever device is implemented it is used to attain the joint angles of the robot. These joint angles found through standard inverse kinematics and singularity avoidance algorithms for the Spaceball or a simple conversion for the stylus. These angles are then used as input for the 3D display. All the computation and control algorithms are written in Matlab. The 3D worlds are created in VRML 97 programming language using VRealm Builder software as the developing platform. Matlab's Virtual Reality Toolbox allows for the interaction of the written algorithms and the VRML visualizations through an internet plug-in. This plug-in allows for stereoscopic vision if the hardware conditions are provided (i.e., video card and stereo devices).

The 3D worlds have been developed to mimic real life situations which will be helpful for the users to be trained on for real life applications. The worlds created include

simple worlds which components are created to be part of functional tests. The design of the worlds may include put and place tasks, stacking tasks, and a fine dexterity in the form of a chess game. There are more complex environments where once the user has mastered the movement of the joints of the robot he/she can simulate moving through as in the wheelchair. These environments include a kitchen and an office. The user can move through them and manipulate certain objects as they command by using the input devices. These environments are helped in their realism through the use of an advanced graphics card which allows for stereoscopic vision display. With this aspect users can perceive depth and closely corresponds to their perception of space as if they were in their wheelchair with the real manipulator attached to it.

2.1 Interfaces

2.1.1 Immersion Probe and Personal Digitizer

The Immersion Probe™ which will be referred to as the stylus, is a six-degree-of-freedom tool with six revolute joints and three of them located in its end-effecter. In this way its shape is similar to that of the robotic Arm that it is used to control, but the link lengths and offsets differ which does not allow a direct one to one correlation. The shape can be seen below.

Figure 2.1 Immersion Probe

Because of its similar shape this device controls the ARM manipulator through mapping the position and orientation of the tip of the stylus to that of the end-effecter of the ARM. Although it requires quite a bit of movement and control from the user, the advantage of this device is that it uses intuitive movement that is similar to what the user would expect if they were to manually move the end-effecter of the ARM itself. Although these movements are not a directly scaled mapping it requires less imagination and learning of the movements the ARM makes. A secondary advantage is that it requires less processing of the input in terms of the response obtained from the model.

The stylus uses optical encoders at each of the joints and custom circuitry along with RS-232 serial port to communicate with the PC (Immersion Corporation). The communication protocol was previously established by another member of the NJIT RERC on Technology for Children with Orthopedic Disabilities and more information on how it was established, what the components of the system are, see Ramirez, 2007.

2.1.2 Spaceball® 5000

The Spaceball® 5000 is a device that receives forces and torques as inputs in six degrees of freedom. Pressure can be applied directly to the device in six different directions and the output is received as either positions or velocities. The directions of the inputs are shown below, which are three axes of force (red arrows) in right/left, fore/aft, and up/down and torques (blue arrows) around those axes. Figure 2.2, shown below, illustrates the Spaceball device and demonstrate the forces and torques used to operate it.

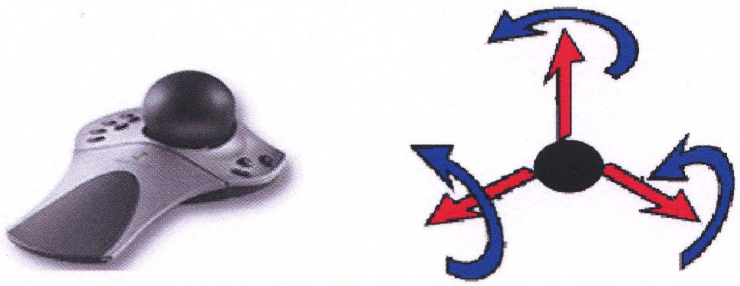


Figure 2.2 Spaceball and Directions of Force

The device uses optical sensors to gather the input information and communicates with the PC through USB (universal serial bus). The device drivers are already available with the use of Matlab's Virtual Reality toolbox so communication is simple. The input is gathered as a Simulink source and the parameters of the Spaceball are adjustable through a Simulink source parameter block GUI (graphical user interface), which is shown in figure 2.3 below. These parameters include Mode, enabling axes, sensitivity and others which can be changed by either entering values or checking boxes.

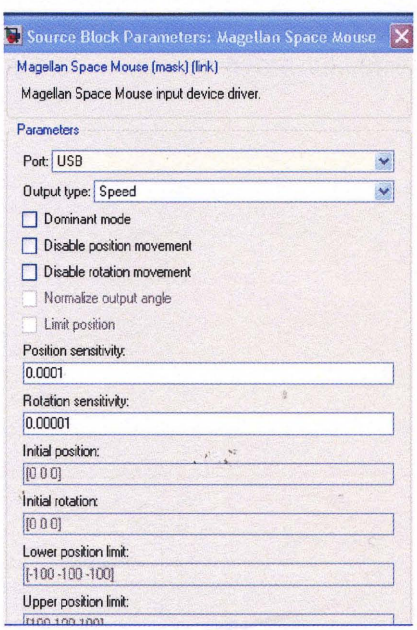


Figure 2.3 User Interface for Spaceball

2.2 Software Applications

2.2.1 Virtual Reality Toolbox

The Virtual Reality Toolbox is a means for modeling active systems in virtual reality. It broadens the capabilities of Matlab and Simulink to include virtual reality graphics. The Virtual Reality Toolbox features include tools for creating and visualizing virtual reality models. These features are Matlab functions and Simulink function blocks useful for programming dynamically changing graphic scenes, and also software and installation packages for creating, editing and viewing the virtual scenes. The software tools are VRML support, VRealm Builder® which is software for building 3D worlds in the VRML syntax, and Blaxxun Contact® IE plug-in for VRML viewing in an IE window.

2.2.2 VRML

VRML is an acronym for the Virtual Reality Modeling Language. VRML is a 3D interchange format and can be thought of as a 3D analog to HTML. The creation of the language stemmed from an effort to enhance the content of Web pages with advanced three-dimensional graphics and interaction with those graphics. VRML is an open and flexible platform for creating interactive three-dimensional scenes. The use of 3D graphics has become more relevant outside of the traditional forums of art and gaming due to the improvement of graphical and computational ability of computers. Technical and scientific uses for 3D visualizations have become more attainable through the help of the Virtual Reality toolbox by MATLAB (Humusoft s.r.o. and The Math Works, Inc, 2001).

VRML "defines most of the commonly used semantics found in today's 3D applications such as hierarchal transformations, light sources, viewpoints, geometry, animation, fog, material properties, and texture mapping" (Carey, R., Bell, G., 1997). Virtual worlds are created in VRML using what is called hierarchal scene graph. The scene graph is composed of nodes. VRML defines 54 different types of nodes which contain information about the properties of the entities inside the virtual world. These properties include geometry, appearance, sound, and others. Fields are the data storage mechanism for nodes and are subordinate to nodes in the hierarchal structure. There are 20 types of fields which can be of varying data types (Carey, R., Bell, G., 1997). In this structure nodes can have "children" meaning it contains other node(s) or nodes may have more than one "parent" and is hierarchally underneath one or more nodes. This structure allows for the creation of complex objects or worlds from simpler subparts. VRML files (.wrl extension) contain the header, the scene graph, prototypes, and event routing. The prototypes and event routing were not used in this document and will not be described further. After a complete world is built the VRML files are then processed by a browser for viewing.

2.2.3 Virtual Reality Editors

VRML files use a standard text format and therefore any common text editor can used to create or edit VRML files. V-Realm Builder is a software application used to generate VRML (.wrl) files, similar to the way text or software can generate HTML files for web pages (Humusoft s.r.o. and The Math Works, Inc, 2001). V-Realm Builder provides a platform for creating worlds without a deep understanding of the language. It ensures the correctness of the syntax of VRML file and provides graphical feedback for the design

decisions made. There are other 3-D editors, which have different formats and abilities that have the capability to export their format to the VRML format. These packages include cad packages such 3D Studio and ProEngineer. V-Realm Builder is an editor that uses VRML as its native format and it is included with the Virtual Reality toolbox from Matlab. It is considered one of the best VRML native editors, but it was found to have a steep learning curve by this author. Therefore, ProEngineer (a software package with which this author had preexisting expertise) was used to create the more complex geometries to cut down on the time that would otherwise be spent learning the software and building. The non-native geometries were then imported into the geometry field of a node in a world created in V-Realm Builder.

2.2.4 Virtual Reality Viewer

Once a VRML file is built and saved it can be manipulated dynamically through Simulink blocks, and Matlab m-files. These manipulations can be also viewed dynamically through the use of a VRML Viewer. The Virtual reality toolbox includes two viewers for viewing virtual worlds, an internal and an external viewer. The internal viewer is rigid and does not allow for stereoscopic viewing, so the external viewer was used for both development and experimentation. The external viewer has to be installed through Matlab and set as the default viewer. Once set as the default, any world opened for viewing though Matlab or Simulink will be opened in the external viewer. The viewer used is called Blaxxun Contact and it is technically a plug-in which allows VRML viewing through either Internet Explorer or Netscape internet browsers. In order for Blaxxun to work properly several adjustments need to be made to the host computer. The virtual reality has known bugs when dealing with newer versions of Microsoft Internet

Explorer (IE) and operating systems (Humusoft s.r.o. and The Math Works, Inc, 2001). If the system is using the version of IE 5.5 or later and the operating system is older than Windows XP Service Pack 1 (most machines are using Service Pack 2) Microsoft Java Virtual Machine (JVM) must be installed on the PC and to avoid complications Sun's version of java should be uninstalled. Due to a court settlement, Microsoft JVM stopped being distributed after 2004, and in 2007 was no longer supported, so versions must be obtained by a third party site. After assuring all the software is installed correctly, the default network security setting must changed before using the Blaxxun Contact to ensure that the virtual scene is updated appropriately. Navigate the menu system of IE to enable network access to all address by the steps below.

From the Tools→ Internet Options → Local Intranet →Security →Custom Level→ Microsoft VM→ Java permissions →Custom→Java Custom Settings → Edit Permissions →Run Unsigned Content → Access to all Network Addresses → Enable → Click OK

These steps must be followed in order for the virtual world to be seen dynamically. Other steps must be taken to assure correct stereoscopic display. The virtual reality toolbox works in conjunction with Blaxxun, the computer's graphics card, monitor and vision system (shutter glasses and sync) to produce a stereoscopic display. In order for stereo to be displayed, the settings for bit depth, monitor size, resolution, and refresh must be matching in the monitor properties menu, the stereo driver menu, and the preferences menu for Blaxxun plug-in.

2.2.5 Robotics Toolbox

The robotics toolbox was developed by Peter Corke, the research director of the Autonomous Systems laboratory in the CSIRO ICT Centre in Australia. The toolbox is available to be downloaded for free from his website and contains numerous Matlab functions to which source code is accessible and editable. The Toolbox provides many functions that are useful in robotics (Corke, 1996). The functions implemented include manipulator definition and modeling functions and the kinematic functions to model the manipulators motion. Also, employed quite frequently are the homogeneous transform functions provided due to the fact that both robotics and 3D graphics use the same linear algebra theory to define positional relationships between different parts of their structure. In robotics, the links are defined by its reference frame, or coordinate system whose z axis is perpendicular to the surface to which the next link is attached. In VRML graphics, the reference frame of a child object is the result of the translations and rotations of its parent. So the final description of the child with respect to the VRML base frame is the following product:

$$^{base}T_{child} = {}^{base}T_{parent} * {}^{parent}T_{child}$$

Similarly, in robotics the description of the second joint with respect to the base is:

$$^{base}T_{joint2} = {}^{base}T_{joint1} * {}^{joint1}T_{joint2}$$

The robotics toolbox provides functions for creating these transformation matrices which has been a useful time saving too when dealing with different reference frames and going between robotics and graphical domains.

CHAPTER 3

DESIGN ELEMENTS OF THE TRAINING ENVIRONMENT

3.1 World Building

Worlds should be built with care and should follow a few simple rules in order to make the manipulation of the objects in that world by the robotic manipulator easier and intuitive to the programmer. The first rule is to keep the fields (like geometry etc.) of parent node of complex objects blank, especially rotation if possible. This is necessary to simplify the location process for complex objects or complex worlds. When an object is designed in the VRML hierarchal structure, each new transform added as a child of a precedent node will have a coordinate system relative to its parent. So if the parent is translated, the child's origin is the translation point of the parent. Likewise, if the parent is rotated its children will have a base coordinate system that is rotated in the same direction. This parental rotation therefore changes the axes in which the children are translated, so locating them mathematically becomes less trivial than a simple node with no children. For example, in Figure 3.1 the cylinder to the right is a child of the cylinder to the left. The parent cylinder is rotated 90 degrees about its z axis and is translated 2 units in the positive y direction, sphere marking the origin. The parent cylinder, since it does not have any parents itself, has the general coordinate system of VRML worlds which is shown. The child cylinder (right) has a coordinate system that is rotated 90 degrees about its z axis of the general VRML coordinate system as shown.

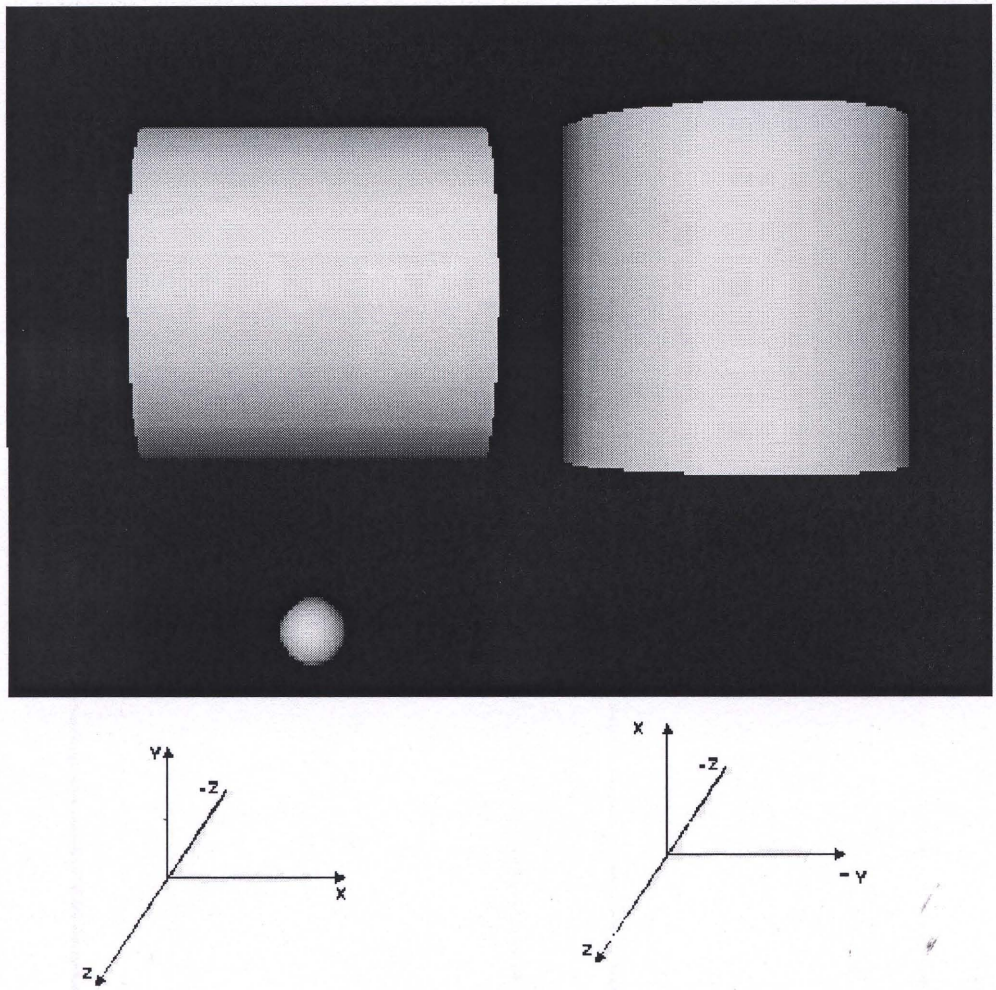


Figure 3.1 Coordinate Systems for Graphical Objects in Hierarchy

It is translated 2.5 in the negative y direction of its coordinate system which is the positive x direction of VRML. In order to find the object to interact with it, the object's position must then be converted to some base frame, for convenience use the VRML coordinate frame. In this simple example it is easily surmised that child cylinder was shifted up 2 and to the right 2.5, which is $[2.5_x, 2_y, 0_z]$ in VRML coordinates. Mathematically to arrive at this and more complex locations, translation matrix of the parent multiplied by the rotation matrix of the parent multiplied by the translation matrix of the child. The mathematical expression becomes:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

These locations can only be found by knowing the hierarchal structure and knowing the rotations and translations of all the objects that a parent to the object. The translations and rotations can be found by accessing the fields of each of the objects which leads one to believe that these equations can be automated but there is no simple way for accessing the hierarchal structure. There are no values held in the "children" field of any node; it is an empty array which acts as a place holder for the object's geometry, appearance, and other node children. Since the structure can not be determined dynamically, the true position of subordinate objects can not be determined either. This means that knowledge of the structure of the world must be present and/or hard coded in order for the objects' position to be obtained and altered. For ease of design, worlds should be created with subordinate nodes only when the nodes contribute to the same object and will move and be treated identically to the parent.

Another good practice when designing worlds is to separate objects into groups of objects that will be interacted with, and those that are there for ambiance. By categorizing the objects within the world that will be interacted with, you can eliminate calculations and make the code more efficient. This is most useful when calculating the collision detection for objects with the end effecter of the robot. To make use of this advantage, a naming standard was created and used in the worlds so that the collision detection algorithm could be used without edit in all worlds. All objects that are intended to be moved by the robot's gripper should be named with the first letters of the name

being "move_" (or any convention useful) so that they could be detected, and put into an array so that they can become a variable that can be dealt with as a whole. With the use of built in Matlab functions (get, strncmp, find, etc.) the world as a whole vrnode data type can be converted into and array of numbers with each row representing the position in space of an individual object. The code to accomplish this task is shown below. This is not a necessary piece of code but it can be made into a function that can used by all programs instead of writing an individual program for each new world that is produces and hard coding the objects that need to be moved into the separate program.

```
myworld= vrworld('world.wrl');
open(myworld)
allobjects=get(myworld, 'Nodes');
x=get(allobjects,'Name')
t='move_'
tf=strncmp(x,t,5);
A=find(tf);
movableobjects=[];
for i=1:length(A)
    z=A(i);
    movableobjects=[movableobjects;allobjects(z,:)];
end
positions=getfield(movableobjects,'translation');
positions1=[];
for i=1:length(positions)
   positions1=[positions1; positions{i}];
end
```

The final guideline to follow when creating a world is to name every node and field. Neither a node nor the data that is contained within the node is accessible and its properties are not editable, unless it is named. It is important to name all of the editable properties to ensure that all properties can be changed dynamically. A naming convention should also be used when naming the properties of the node. When using VRealm Builder object nodes usually have a geometry and appearance as child nodes. The particular naming convention adopted is up to the programmer but naming the appearance and shape nodes "ParentNodeName_appearance" was found to be a

convenient practice. The names should be distinguishable because the simplest way of accessing a particular field is through use of its handle. The handle notation is the same for all nodes no matter where they fall in the hierarchy. Their fields are all accessed in the same fashion which is: "MyWorldName.NodeName.FieldName". Since there is no simple way of identifying the parent(s) and making dynamic decisions based on the parent information of the nodes the naming should be unique, but also systematic so that hard coding is avoidable and multifaceted function writing is simplified.

3.2 Control algorithms

There are a few algorithms that went into the functions for controlling the visual representations of the dynamic virtual environments. Not all environments implemented all of the functions but a comprehensive final environment would encompass all of these and maybe more.

3.2.1 Collision Detection

A person familiar with graphic design might wonder why it would be necessary for someone to write their own collision detection algorithm because the VRML standard and others provide a collision detection system. In VRML this system is composed of the *Collision* node, *collideTime, bboxCenter* and *bboxSize.* The collision node only handles collisions between the user and the world; it does not detect collisions between arbitrary objects in the world. General object-to-object collision detection is not specified in VRML. The specified collision detection detects geometric collisions between the user's avatar and the scene's geometry, and prevents the avatar from 'entering' the geometry or the regions of the world that are not intended to be. An avatar is a computer user's

representation of himself or herself in the form of a three-dimensional model used in games and other computer based graphic systems. The VRML viewers implemented with the robotic manipulator have the option of viewing an avatar or just in a first person view. The VRML default (if there are no Collision nodes specified in a world) collision is detected with all objects during navigation. The *bboxCenter* and *bboxSize* fields specify a bounding box that encloses the Collision node's children. Those fields are also found in all transform geometries and can be used to the advantage of the programmer attempting to implement their own object to object collision detection algorithm. Since the robotic manipulator is an object in the world, and not an avatar, such an algorithm is necessary. The default value of the *bboxSize* field is (-1, -1, -1) unless otherwise changed when building the world. For complex objects this is a good tool to use to dynamically find the area occupied by the object, although it must be determined and entered manually while the world is being built. For simple, one-shape objects it is not necessary to use this field, because a bounding box can be created from the dimensions of the geometry which are stored in the fields: *radius*, for a sphere, *radius* and *height* for a cylinder, and *size* for a box. Assuming that the center field of the transforms are set to zero, a bounding box is formed by reading either the geometry fields of the object or the *bboxSize* field, taking those read values, dividing them by two, and adding and subtracting those values to the values read from the transform field of the object. Sample code for a (named) box becomes:

```
myworld= vrworld('world.wrl');
open(myworld)
i=myworld.BoxLF.size;    %assuming the name of the geometry node is known
j=myworld.LeftFrontBox.translation; %assuming the name of the box node is known
k=j+(i/2);
l=j-(i/2);
```

This gives you the arrays j and k in [x y z] which define the maximum and minimum values of x, y, and z which will define a collision with the object. The object that we are concerned with colliding with movable objects is the gripper of the robotic manipulator. To find the position of the end effecter of the robot we must calculate the transformation matrix which describes the conversion from the gripper reference frame with respect to the reference frame of the base of the robot. To mathematically achieve this, a series of transformation matrices representing the transformations from one link to the next would be multiplied in order to find the transformation from the end effecter frame to the base frame. This fortunately is an unnecessary calculation with the use of the robotics toolbox. Since a robot object is already created, and the end point is at which the gripper is located (through the inverse jacobian calculation), these calculations can used to an advantage. Using the already defined robot, and the calculated angles as input, calling the function fkine from the robotics toolbox returns the [4x4] transformation matrix necessary. This transformation matrix defines the gripper in the frame of the robot base, which is the Matlab graphics coordinate system. In the coordinate system the first link of the robot extends from the origin up the z axis the length of the link as shown in figure 3.2. The red line is not part of the robot, it just indicates the origin of the coordinate system and the base of the robot.

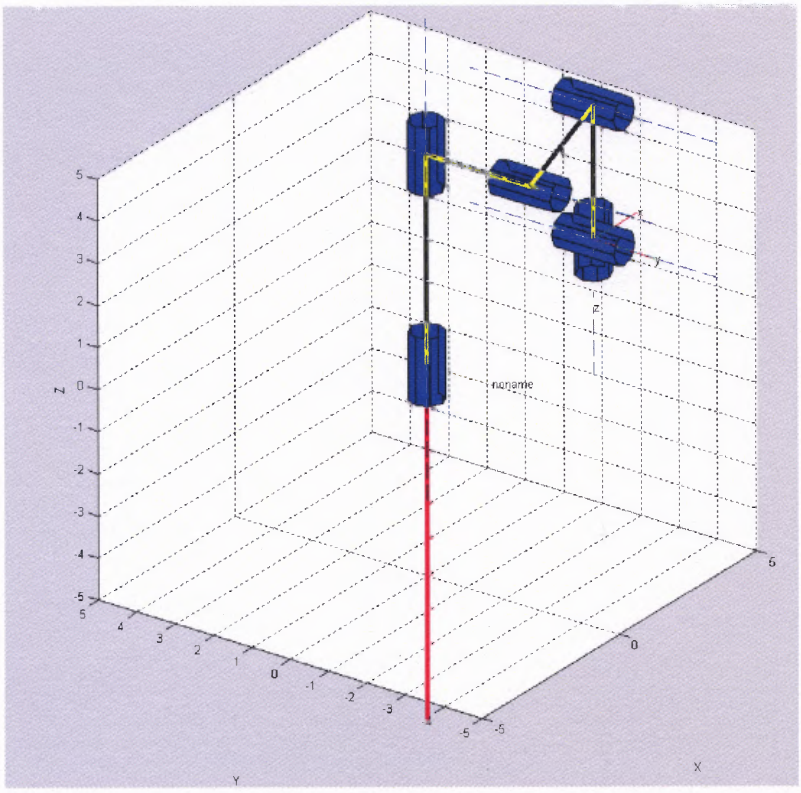
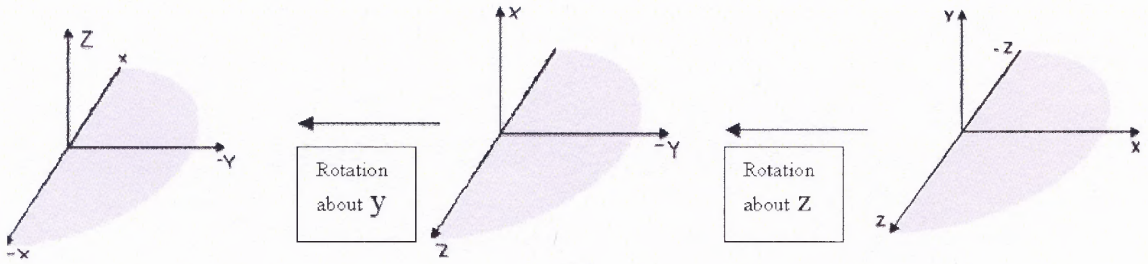


Figure 3.2 Robotics Toolbox Robot and Coordinate System

The Matlab coordinate system differs from the VRML coordinate system. In order to detect the collision between the robot and the graphical objects the coordinate system from one must be converted into the other. These differing coordinate systems can be thought of as different frames, similar to the different frames at each of the joints of the robot. Converting between the two therefore uses the same concepts of homogeneous transformation matrices. To transform the robot's coordinates into VRML coordinates, the robot's coordinates must be rotated ninety degrees around the z axis and then rotated again ninety degrees around the y axis. Figure 3.3 shows these rotations.



Matlab coordinate VRML coordinate

Figure 3.3 Transformation from VRML to Matlab Coordinates

The transformation matrices from these rotations can be made from some simple functions in the robotics toolbox, roty and rotz. Using these functions and the aforementioned inputs and functions, the code for converting the robot's position into VRML coordinates becomes:

forward=fkine(robot_def(urobot),urobot);

transform=rotz(pi/2)*roty(pi/2)*forward;

```
position=transform(1:3,4);
translate=[position(1); position(2)+.5; position(3)];
```

After obtaining the position of the robot in VRML coordinates, these coordinates can be compared to the translations of all the movable objects, and their proximity can be detected, either by the bboxSize field or the size fields of the geometry nodes as stated previously. Also, an array of nodes of movable objects has been created, (section 3.1) so that array makes for the rest of the collision detection to be done by a simple loop.

```
bounds=getfield(movableobjects,'bboxSize');
bounds1=[];
for i=1:length(bounds)
   bounds1=[bounds1; bounds{i}];
end
i=1;
while i<length(positions1)
  difference(i,:)=abs(positions1(i,:)-translate');
  if difference(i,1)<bounds1(i,1) && difference(i,2)<bounds(i,2) &&...
```

```
    difference(i,3)<bounds1(i,3)
     movepiece=i;
     i=length(positions1)+1;
   else
     i=i+1;
     movepiece=0;
   end
end
```

This set of code finds the index of the object with which the robot arm has collided and sets the variable called *movepiece* equal to it. If the robot has not collided with any of the movable pieces, then the variable is set equal to zero. This variable is then used as a flag to trigger another section of code which sets the translation of the object equal to that of the robot end effecter and that code which is a simple if statement shown below.

```
if movepiece~=0
     x=get(movableobjects(movepiece),'Name');
     y=strcat('myworld.',x,'.translation=translate');
     eval(y);
end
```

In this way object identification, collision detection, and object movement can be done with knowing very little about the world itself. These algorithms are useful when programming solely with Matlab because the handle graphic commands and the graphic refresh command, *vrdrawnow*, are ineffectual when calling a Matlab function from a Simulink simulation. Interacting worlds were developed in Simulink because a preexisting communication protocol with the Spaceball device was available with the Virtual Reality toolbox when development of this project was first embarked. Since then a Matlab based communication protocol has been made available with a new version of the toolbox. The developed Simulink programs require a great deal of hard coding. Due to the "sink" functionality of Simulink, values need to be outputted to the sink for every calculation cycle of the simulation. For this reason, any field that might be edited at any

point, such as translation or color, needs to always have a value assigned to it. To accomplish this, arrays were created of the editable variables, usually one row of the array per variable. The value of the array from previous iteration is saved and any variable that is edited in the present iteration is written over its row in the array. Then the array is exported and split, and then sent to its matching sink port. A sample of such a Simulink simulation is show in the figure below. As you can see each movable object, in this case each chess piece needs its own sink port.

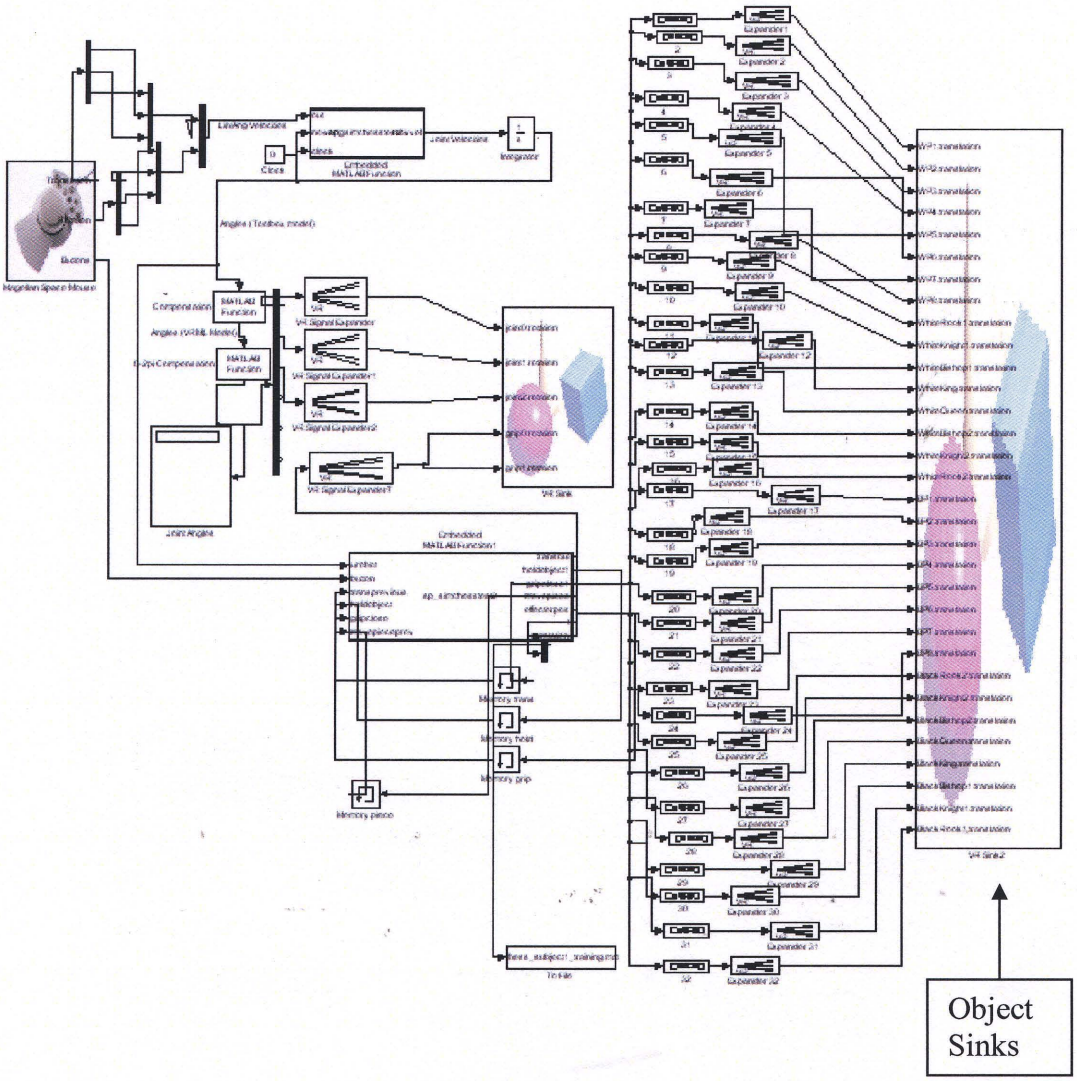


Figure 3.4 Simulink Code For Moving Many Objects

The code above controls the world shown below. The large sink block exists so that all of the chess pieces are movable by the robot. For more virtual worlds see Appendix C.

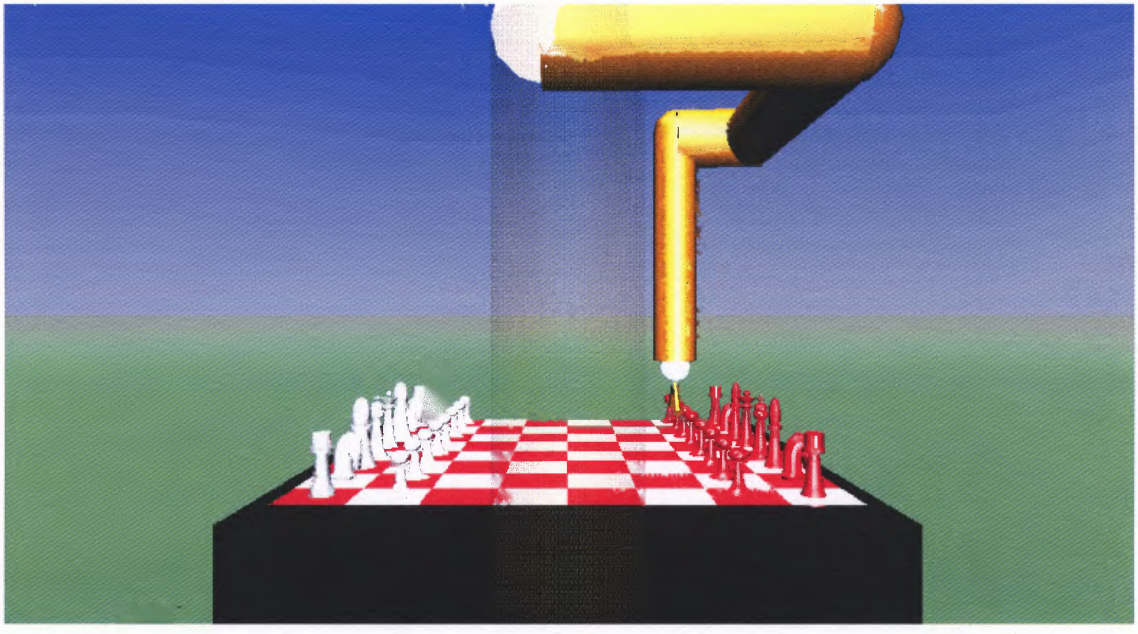


Figure 3.5 Virtual World Where The Robot Can Move Chess Pieces

3.2.2 Object rotation

An object can be rotated with the changes in orientation of the robotic manipulator's end effecter. This algorithm has not been implemented successfully due to some inconsistencies in the end-effecter rotation. This algorithm when implemented is executed alongside the collision detection and movement algorithm. When an object is picked up its translation is set equal to the position of the robot's end effecter. This can be done with rotation in the same fashion with a few exceptions. The rotation matrix inside the transformation matrix has to be converted into a different sort of array that VRML recognizes. Luckily there is a Matlab function and a Simulink function block in the VRML toolbox to handle this conversion. The other difference is that the rotation of

the object can not be directly set to the rotation of the object. If this is done, as soon as the object is detected it will jump from its original rotation to the present orientation of the gripper. Instead the difference between the orientation of the gripper in the previous iteration and the present iteration should be added to the present orientation of the object. In this way the orientation is adjusted from its own orientation and does not just mirror the robot's orientation.

3.2.3 Toggle control

An aspect that was important for functionality and realism was to create a working gripper that toggled from open to close. It is a simple aspect but an important one nonetheless. Creating a working gripper also adds to calculation efficiency. The efficiency is improved by preventing unnecessary calculation from being executed. For instance, collision detection only needs to be performed when the user intends to pick up a movable object. This intention can be presumed when the gripper goes from open to close. By creating a set of variables as flags and a system of nested if statements, calculations will be performed only when necessary. The variables created represent the state of the gripper in reference to the movable objects. The variables are "button" which represents whether or not the button was pushed, "gripclose" which represents whether or not the gripper is closed, and "holdobject" which represents whether or not an object is held and its translation is being altered along with the position of the robot's end effecter. These variables are boolean where a value of one/true indicates that the action indicated by the name of the variable is true (ex. gripclose equal to one indicates that the robot griper is closed). The toggle is enacted as a result of a boolean button on both of the implemented user interfaces. The buttons read a value of one when the button is pressed

down and a value of zero when released. This is used as criteria for determining the state of the environment. Depending on the state of the environment, different actions need to happen. If the gripper is open and the button is pushed, the gripper should close. Also, the collision detection algorithm should execute and if an object is collided its position should be manipulated and the "holdobject" variable should be changed to true. If the button is pushed and the gripper is closed, the only action that needs to be taken is to open the gripper, which require no calculation only a handle set command. If the button is not pushed, the variables are detected from the previous loop execution and an object is moved or not moved based on the values of "holdobject" and "movepiece". The code for this is shown below.

A toggle was also created to switch between a simulated wheelchair motion and control of the robotic manipulator. This feature was only implemented with the Spaceball interface. For this control a second boolean button was used and only one more variable needed to be created, named "wcactive". When the wheelchair is active, the robotic manipulation is turned off. This is a safe and convenient feature to have so that the user can focus on one task and have only one interface to use when implementing on a real wheelchair. There is pseudo code below to show this toggle.

```
if button==1;
  pause(0.01)
    if wcactive==0
      wcactive1=1;
    else
      wcactive1=wcactive
    end
else
  wcactive1=wcactive
end
if wcactive1==0
      urobot=robot_alg(mouse_input)
```

```
else
[robot_base, base_orientation]= chair_alg(mouse_input)
end
wcactive=wcactive1
```

3.2.4 Wheelchair movement

For the wheelchair movement effect, the user utilizes the fore/aft force direction to navigate through the world and the torque about the y axis to turn the chair right or left. None of the other degrees of freedom are available when the wheelchair is active because movement in these degrees of freedom is not realistic. In this way the Spaceball is converted into a two degree of freedom joystick similar to those employed on most automatic wheelchair controllers. The algorithm for this is not overly simple. Since the coordinate system for the virtual world and Spaceball are fixed, there has to be a way of accounting for the rotation of the robot. Keeping the fore/aft (x) direction of the Spaceball mapped to the forward movement of the virtual robot, which when rotated is in terms of x and z in the VRML coordinates system, requires some calculation. The angle of rotation about the y is taken from the Spaceball input and is used to calculate the resulting x and z values, which are velocity vectors. Sine and cosine can be used for this but using the roty function accomplishes the same goal and is available so it was employed. After the new velocity vectors are found, they are integrated and sent back into the function as positions. The rotation transformation has to be performed on the velocities because if the positions are rotated, the coordinates of the base will be shifted when a rotation is performed. The code which accomplishes this is shown below and is executed when the toggle sets wcactive to true. Be aware that this function is nested

inside a Simulink simulation and that the variable newvel is integrated in Simulink and sent back as newpos.

```
chairorient=[0; 1; 0; newpos(6)];
linvel2=roty(mouse(6))*[mouse(1); mouse(2); mouse(3); 1];
linvel=mouse;
linvel(1)=linvel2(1);
linvel(2)=linvel2(2);
linvel(3)=linvel2(3);
basepos(1)=newpos(1);
basepos(2)=newpos(2);
basepos(3)=newpos(3);
```

The chairorient and basepos variables are output to the sink of the robot's base rotation and translation fields respectively. If this code were to be executed in a Matlab program, the integration would have to be executed differently and some handle commands would have to be used to set the robot's base rotation and translation fields.

Another component to the system is that the viewpoint of the user moves with the robot. This is accomplished by making the Viewpoint node, a child of the base of the robot. Doing this allows you to set the position of the robot base as a function of Spaceball input and keep the viewpoint at a fixed distance from the base of the robot without altering any of the fields of the viewpoint node. This makes the visual representation more realistic and gives a first-person navigation through the virtual world as if the robot was attached to a real chair (at a fixed position) and the chair is moving through space.

CHAPTER 4

IMPLEMENTATION

In order to quantify the quality of the virtual training environment or the potential for useful implementation of the product, a series of experiments were completed. The goal of the experiments were fourfold; 1) to determine how well an non-disabled subject can control a virtual robot, 2) to evaluate how well (the discrepancies) an non-disabled subject can learn to use two different types of input interfaces, 3) to compare the types of movements made in real life (with actual human arm) to the types of movements made by a virtual robot with two different interfaces, 4) and to determine the maximum amount of function that can be provided to eventual users of the robot and one of the new interfaces, or in other words create an able bodied baseline for future comparison with data from disabled users. Upon receipt of statistically relevant data, further studies could be undertaken to show increase in social (not physical) function in users whose physical ability is too small to measure in most standard functional tests.

4.1 Experimental Design

To investigate these aims a series of experiments were planned and executed to achieve the answers to these questions. Two types of experiments were done, a box of blocks function test, and a Fitts' law paradigm test. Both of these tests were completed in a real life environment and in a virtual reality environment. The virtual reality portions of each of the experiments were performed by each subject 24 hours after a 15 minute training session on each input device. This thirty minute training session was meant to introduce

the subjects to the interfaces and to give them a feel for the movement of the robot. The end effecter of the robot moves in X,Y,Z planes according to the input from the user. The orientation degrees of freedom of the robot were turned off for the testing so that the motor task that needed to be learned could be simplified. These tactics were chosen because the task of controlling the robot's three degrees of freedom was deemed difficult enough to require learning. The training was meant to induce a fast learning stage in which a vast amount of skill is acquired in a little time, which has been shown to happen in the acquisition of motor skills (Ungerleider et al., 2002) and hopefully the results would have implications for the ability of a person who has overlearned the task. Motor training sessions have also been shown to have a greater impact when evaluation takes place 24 hours after the session as opposed to immediate post training measurements (Ungerleider et al., 2002), which explains the reasoning for the delay of testing.

The experiments were conducted on 6 subjects, 4 female, 2 male. Five of the subjects were right handed, but all subjects used their dominant hand for all parts of the experiments. The subjects were considered healthy with no known motor or neurological disorder whose ages ranged from 23-31 with a mean age of 26.6. The subjects were subjected to the same two day regiment with the only difference among the subjects being which of the tests were performed first.

The first experiment day consisted of all of the real life experiments, and two fifteen minute training sessions in virtual reality, one session per user interface (Stylus and Spaceball). The real life experiments consisted of two Fitts' Law tapping paradigm tests and a box and blocks test with three replicates. The Fitts' Law tapping tests consisted of 3 Indexes of Difficulty and 4 orientations of motion, resulting in 12 trials per

subject per paradigm. The same 12 trials were executed by each subject using just their finger and also using the stylus (the same stylus used as the VR interface) to tap the targets. All subjects performed the 12 trials for each tapping interface in the same random order created by a DOE design in the statistical software Minitab. Three subjects performed the tapping tests first with their finger and the other three subjects performed the tapping test first with the stylus. After the Fitts' Law real life trials, the subjects performed the box and blocks test three times. After all the real life trials were run, the subjects were taken to the VR station and instructed to play virtual reality chess with the robot manipulator. They were given a short set of instructions on how the interfaces were to be operated, and told to get accustomed to the way that the robot's end effecter moved. The two fifteen minute training sessions for each of the input devices were performed back to back in no particular order. The entirety of day one testing took about ninety minutes per subject.

Twenty-four hours later the subjects were asked to perform a second session of testing. The second session consisted of duplicates of the real life tests in virtual reality. The subjects performed twelve trials of Fitts' Law tapping test with both of the interfaces, stylus and Spaceball. The twelve trials were done in a different but random order for each of the interfaces and again, half of the subjects performed the Fitts' Law tests with the Spaceball first and the other half started with the stylus. The virtual Fitts' Law paradigm consisted of tapping the tip of the gripper of the robotic manipulator to virtual boxes that were sized and positioned inside the virtual world representing the same three ID's and orientations that existed in real life.

The four orientations were chosen to see if there was a discrepancy in the learning of movement in certain directions. The purpose of the multiple orientations was to investigate whether a delay in movement time existed in real life and whether this delay is proportional in the virtual world or whether certain planes of motion are more difficult than others based on other factors. The stereoscopic vision is supposed to assist in the acquisition of the targets placed in the depth of the field of view, but the robotic motion might be more complex to control in certain positions in the field or in certain orientations. In some orientations the VR blocks had to be strategically placed so that the entirety of both blocks were inside the workspace of the robot and the stylus while also maintaining the intended ID of the trial. Below is a figure of the general orientations. In the real life experiments the subjects were centered between the targets.

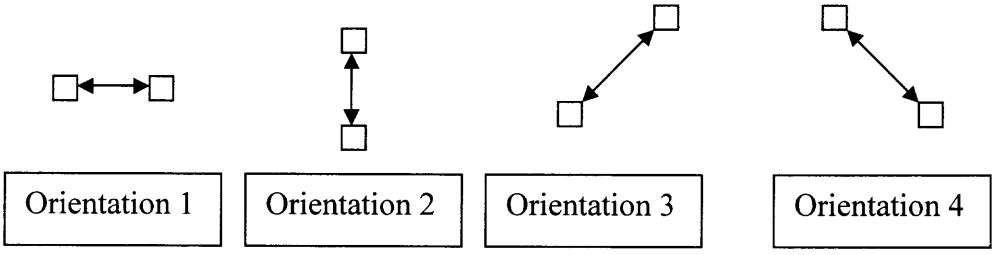


Figure 4.1 Orientations for Fitts' Law Tapping Experiments

The Indexes of difficulty for the experiments were also designed for ease of use in virtual reality then duplicated in real life. Making small targets in VR would make it extremely difficult to achieve the position of the robotic manipulator, since the robot is relatively bulky as compared to a person's finger. The dimensions of the robotic manipulator in the VR worlds are 100 mm per unit. The target sizes in VR are 1 by 1 and 2 by 2 squares, which are stationed at distances of 4 and 9 units creating the following ID's.

$$ID = \log_2 \frac{2A}{W} = \frac{2*4}{1} = 3 \qquad ID = \log_2 \frac{18}{1} = 4.16$$

$$ID = \log_2 \frac{18}{2} = 3.16$$

The ID's for the real life experiments used the same proportions, resulting in the same values for ID but both dimensions, target width and amplitude, were decreased by a factor of to for ease of reaching. The dimensions of the real life experiments then become 50mm and 100mm for target width and 200mm and 450mm for distance between targets. The targets were bright colored paper squares pasted at the necessary measured distances on white paper. One piece of paper was made for each ID and orientation combination, totaling twelve. At the beginning of the trial the paper with the targets was centered in front of the subject and taped down to the table.

4.2 Data Collection

The data for all the real life experiments were gathered using a Nest of Birds device along with a Matlab program that communicates with the PC by serial port. The Nest of Birds device has an electromagnetic transmitter and a sensor, the transmitter reads the six degrees of freedom, position and orientation, at a desired sampling rated. The program collects the data, filters, plots and saves it to file. The program was written by Katharine Swift, a member of the NJIT RERC. By attaching a sensor to the finger, stylus, and wrist for the finger tapping test, the stylus tapping test, and the box and blocks test respectively, the nest of birds tracks the trajectory of the hand moving through space through the duration of the test. Each Fitts' Law tapping trial was run for 5 seconds and timed by the program. The subject was instructed when to begin, tapped the targets as

fast as possible while maintaining accuracy, and was told when to stop. The time of each of the taps, was calculated from the extrema of the trajectories and the sampling rate of the nest of birds. The subjects were also instructed to start at the center of one of the targets so that the trajectory data could be determined by using the beginning data as the center of one target and the known values of the amplitude and width of the targets to ensure that all the taps fell within the limits of the targets. The Box and Blocks test was done in a similar fashion, where the duration was changed to sixty seconds and the subjects were instructed to move one block at a time over the barrier as quickly as possible.

The data from the virtual reality experiments were gathered from the programs controlling the visual representations themselves. The collision detection algorithm calculates the location of the end effecter of the robot for every iteration. That position is used to detect a hit on the virtual target and turn it green, and is also saved to file along with the CPU time. The saved files were processed using similar code to detect a hit and the CPU times were subtracted to find the time between hits.

4.3 Results

On completion of the experiments the data was analyzed in several ways. One of the main purposes of the experiment design was to evaluate the time for the Fitts' Law tapping paradigm. The table below shows the Analysis of Variance for the times. The significant sources of variance based on the P values were found to be task (a.k.a input device), the index of difficulty (ID), the condition or orientation of the targets, and interaction effects between task and ID, and task and condition. Based on the F values

the more significant sources are the Task, ID and the task/ID interaction. These significances are not surprising because it was anticipated that there would be differences between the input devices and Fitts' Law shows that the movement time is dependant on ID. It should be noted that the R squared value is relatively low meaning that there are other reasons besides the factors considered for the variance in the data.

Table 4.1 Analysis of Variance for MT for All Input Devices

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Task	3	4050.23	4050.23	1350.08	122.59	**0.000**
ID	2	286.42	286.42	143.21	13.00	**0.000**
Condition	3	96.41	96.41	32.14	2.92	**0.035**
task*ID	6	643.81	643.81	107.30	9.74	**0.000**
task*Condition	9	279.18	279.18	31.02	2.82	**0.004**
ID*Condition	6	62.93	62.93	10.49	0.95	**0.458**
task*ID*Condition	18	184.82	184.82	10.27	0.93	**0.540**
Error	240	2643.17	2643.17	11.01		
Total	**287**	**8246.97**				

S = 3.31861 R-Sq = 67.95% R-Sq(adj) = 61.67%

The most significant source of variation is the input device as expected. The table below shows the means of the data based on task and ID separately. There is more than a 28 fold increase in movement time between the finger task and the Spaceball, but there is only a two fold increase in movement time between the smallest and largest ID. With greater differences in ID these means would have been different but they still would not have been as great a difference as with the tasks.

Table 4.2 Mean Times per Task and ID's

Task	Task Means
Finger	0.3376
RL Stylus	0.4231
VR Stylus	2.3657
Spaceball	9.4976

ID	ID Means
3.00	2.2121
3.17	2.7203
4.17	4.5354

The Interval plot below shows the range of values based on the three factors. It is quite obvious that the range of values for the real life experiments fall within the same region and the effect of virtual reality is apparent in the difference between the real life (task 1 and 2) and the stylus experiments (3). The disparity between the fourth task and the third can be attributed to the misunderstood and misinterpreted motion of the robotic end effecter but also the disparity in the speed of movement.

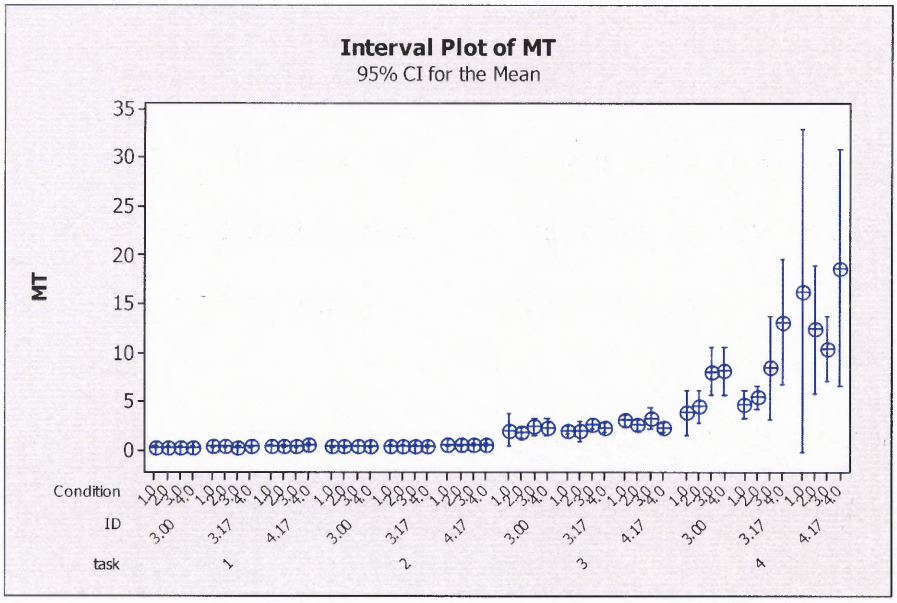


Figure 4.2 Interval Plot for Fitts' Law of MT Over Condition, ID, and Task

A simpler interval plot is show on the next page, in figure 4.3, indicating the intervals based solely on the input device.

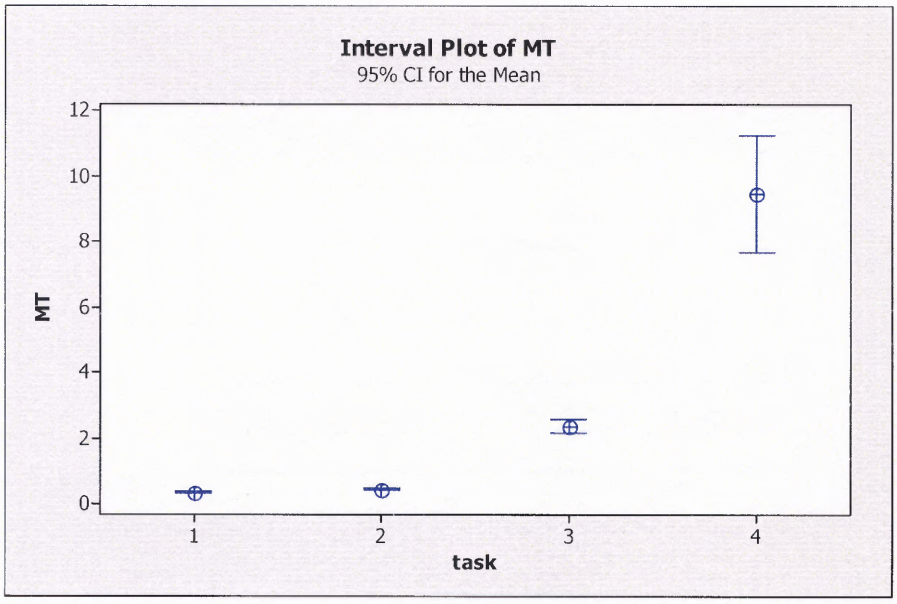


Figure 4.3 Interval Plot for Fitts' Law of MT Over Task Only

The motion of the Spaceball is limited by the sensitivity setting in the Spaceball menu but the motion of the stylus is only limited by the ability of the subject to move and interpret the 3D screen. This sensitivity setting sets a maximum value that can be read by the algorithms and interpreted as velocity. For all testing the value of the sensitivity was arbitrarily set to 0.0001. Although this value has no particular interpretation it keeps the visual scene seamless and looking realistic. It is not without its reasoning either. The maximum velocity of the robot for which this model is created is approximately 9.9 cm per second. Using this as a guide, the largest ID is virtually 90 cm apart. This means that in real life the robot would only be able to reach the targets at a minimum of 10 seconds. This suggests that for realism sake, the sensitivity should be decreased. But for practicality sake, decreasing the speed of the movement of the robot by the Spaceball would add to an already frustrating task of learning these counterintuitive motions.

To further understand the motions and how the subjects interpreted the different input devices the trajectories of each of the Fitts' Law tapping tests were analyzed. Sample trajectories for each of the tests from trials with the same ID and orientation can be found in Appendix A and are labeled according to the device used. Take note that the Real life dimensions are in inches and the VR simulations are in 100 mm. The trajectories are shown in 3 different forms, a 3D plot, a 2D plot showing the plane of motion in which the targets are situated, and a position versus time plot showing x, y, and z trajectories versus time or sample number. Each plot shows something different about how the users use the different devices. In the 3D plots, there are several observations to be made. The height trajectory of the finger tapping reduces over time which indicates improvement a reduced distance that the finger is traveling. No such improvement is present in the stylus tapping. This indicates that the device itself or the act of transferring movement inhibits this type of optimization of movement. Comparing the VR stylus plot to the finger and stylus plots shows that the types of movements made by the robot's end effecter with the stylus are similar to those naturally made by the human arm. This is an encouraging result because robotic movement is only a mapping of the angles of the stylus device and therefore the movement made by the human arm using this device should be similar. But when comparing the movements made by the robotic arm when controlled by the Spaceball to the other three devices there is an evident disparity in the movement. The movements are not in the same arch form and it is difficult to determine where the targets were placed. Also it is easy to see that the subject overshoots the target in all directions and must correct the motion. It is also important to point out that the movements are more boxy and less smooth which could indicate that the user interprets the trajectories

and/or the corrections that need to be made in one or two degrees of freedom at a time instead of all three at once.

The planar movement also assists in understanding the movements and how they are interpreted by the user. The planar real life movements are very similar, the only difference being that the finger is a little more accurate in reaching the center of the target which is not a surprising result. In the VR stylus planar movement there is more deviation in the trajectories from the center line from target to target, meaning the subjects do not follow an exact and efficient strait line movement which should be their motor goal. The two dimensional plot of the Spaceball reveals the discrepancies of movement in a more manageable manner than the 3D case. By studying this plot it can be seen that the subject makes one sweeping movement in both degrees of freedom to approach the target then a series of corrections and overshoots are made until the target is reached. There are several possible explanations for this type of motion. One is that the exact location is not perceived well in the virtual environment and a general trajectory is planned and then corrected more significantly. This is supported by the fact that the stylus trajectory is looser that the real life trajectories. Another explanation is that the inertia of the Spaceball is not easily predictable, and so the user overshoots even when they are trying to correct their motion. This inertia is an inherent property of the Spaceball and cannot be changed without changing the device drivers. A further explanation is that simply the movements of the robot from the forces on the Spaceball were not fully understood by the subjects and the trajectories suffered from a sustained lack of understanding.

These conclusions can also be reached by investigating the dimensions versus time. In the real life trials, the maximum and minimum values of the x and y values (blue and red) indicate the point of a target. The z value (green) is the vertical value and reaches a maximum value in the middle of a motion and the same minimum value when both targets are reached. These trajectories are smooth and show very little hesitation on the target. The VR stylus is not as smooth as the real life trajectories but is most significant in the vertical value (red). This value shows local extrema where the subject is assumed to be hitting the targets, which indicates some significant overshoot corrections in this direction in almost every target approach. This trend is present in all three values of the Spaceball trajectories. These corrections are so significant in the Spaceball trajectories that it is very difficult to even see a maximum and minimum sinusoidal trend. It is this investigators personal opinion, through observations made throughout testing and personal exposure to the use of the Spaceball to control the robotic manipulator, that vast improvements can be made in timing and trajectory through more training and exposure to the Spaceball and the virtual simulations. Over time this exposure could lead to Spaceball values approaching those of the virtual reality stylus trials.

The Box and Blocks Test (BBT) purpose was to investigate a more realistic type movement which could restore social function. The virtual BBT incorporates complex motion in three dimensions but also incorporates the use of a button to grasp and release the blocks. The trajectories of these trials show similar results to the Fitts' Law except that the motion is less predictable because in both real life and VR the blocks are located in different places and the subject must move in a wider area to move the blocks. A

simple analysis of the means tells us a great deal of information. Below is an interval plot of the three tasks. The variable measured is the number of blocks moved in one minute.

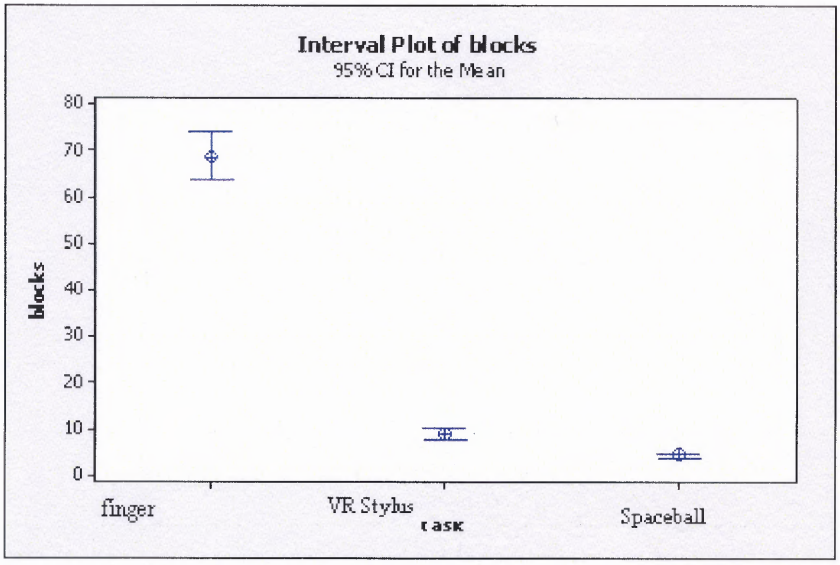


Figure 4.4 Interval Plot for BBT of Task Only

What is noteworthy about this simple mean comparison is the difference between the Fitts' Law test and BBT. The real life trials ended up moving on average seven and half times more blocks than the VR stylus which in turn moved double the amount of blocks as the Spaceball. In terms of time, it took on average 0.873 seconds to move one block in real life while the VR stylus and Spaceball took 9.27 and 13.67 seconds respectively. This translates into a seven and a half fold increase in time from real life to VR which is approximately the same as the Fitts' tapping but the increase from real life to Spaceball was cut from 28 fold to 15 fold or between VR stylus and space ball the increase was cut from 4 fold to 2 fold. This can mean one of two things. Since the BBT tests were run after all the Fitts' tapping tests, that learning occurred from the beginning of the sessions to the end. The other option is that with more complex series of motions or tasks, that are

closer to real life activities, the awkward and less natural motion makes less of an impact. Either option is encouraging result and gives reason to investigate an overlearned condition on subjects with the Spaceball. The trajectories in Appendix B also support this conclusion. Similar to the Fitts' Law finding, the trajectories from real life are very similar to those from controlling the robot in VR with the stylus. The Spaceball trajectories show improvement in the coordinated three dimensional motion, meaning the subject trajectories are more curved and natural looking an move in all three directions at once. Also, the ability to locate a specific position in space is improved. The spikes in trajectory located near the zero point on the z axis indicate where the blocks are located. There is very minimal searching and trajectory correction as there was in the Fitts' Law studies. The lack of searching and improved the curvature affirm the belief that a user can learn to use this interface and possible others to control the wheelchair mounted robot.

CHAPTER 5

DISCUSSION AND FUTURE WORK

The NJIT RERC has many more things to do to help successfully deliver assistive robotics to a wider range of users whose activities of daily living would be simplified and whose self esteem would be increased by the realization of this technology into their lives. One of the most significant contributions to the science and populations that seek to develop this sort of technology, is to continue to produce new technologies and creatively implement existing technologies to create a wide range of user interfaces to control the robot. One of the most challenging aspects of rehabilitation engineering is that there is not one answer that is right for everyone. It is the challenge of the scientific and clinical communities to come up with new and innovative solutions to enhance the lives of those who may be in need of some sort of assistance. By creating a variety of options for control of the robot through different types of interfaces, the amount and types of users that can be accommodated will be increased.

The results from the human motion studies on six healthy subjects gave some encouraging feedback for the work done on assistive robotics. The Spaceball translates isometric force and torque inputs into velocities. These velocities are then translated into a visual representation of movement of a virtual model of a Manus ARM robot. This type of isometric motion transfer is not found in many applications and is therefore not a previously learned skill or a natural or intuitive skill. Continuing the research of this paper by investigating subjects who have more training on the device and have time to learn the skill sufficiently, would hopefully give a better idea of the value of the maximum speed and functional ability of the ARM robot controlled by the Spaceball.

The conclusions drawn from observations made on experiments with subjects who have an overlearned level of skill with the Spaceball will give credence to the work of developing a wider range input devices for users with varied physical abilities.

After the confirmation of the belief that subjects can learn the skills involved with new and counterintuitive devices, steps can be taken toward implementing these new interface technologies with an actual physical Manus ARM. The implementation will not be without its challenges but, after it is complete a training regiment can be designed for real subjects whose physical ability would warrant a system of this nature. This regiment would include an evaluation session which would ascertain which input device fits the needs of the user. Followed by a series of training sessions within several entertaining worlds in which the user would attain an expert level of control of the robotic manipulator. Then the user would move to a series of distance sessions with the ARM mounted a safe distance away from the user. The robot would also be mounted near a simulated real life situation training environment such as a kitchen. After the subject can perform tasks efficiently and safely in the simulated environment then steps can be taken to provide a customized wheelchair mounted system.

The goals the NJIT RERC has set are realistic and attainable in the relatively near future. The downfall of other assistive robotic technologies is that the technology is too young in the development process and too advanced to be realistically implemented in actual patients' lives. The steps proposed would improve the usability of an existing technology and deliver a device system that can be used by patients in the near future and not discarded for being inefficient or too difficult to use.

APPENDIX A

SAMPLE TRAJECTORIES FROM FITTS' LAW TESTING

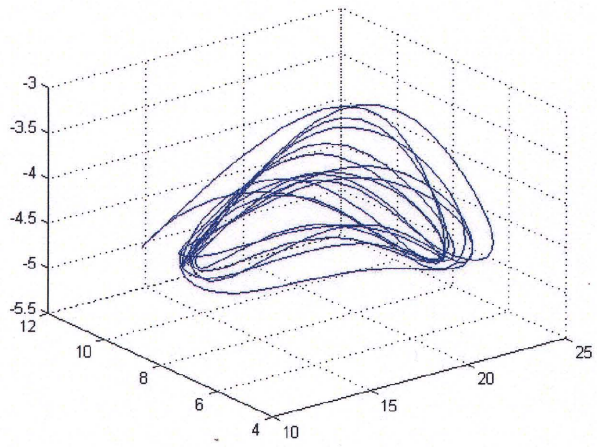


Figure A.1 X, Y, Z Plot of Finger Movement

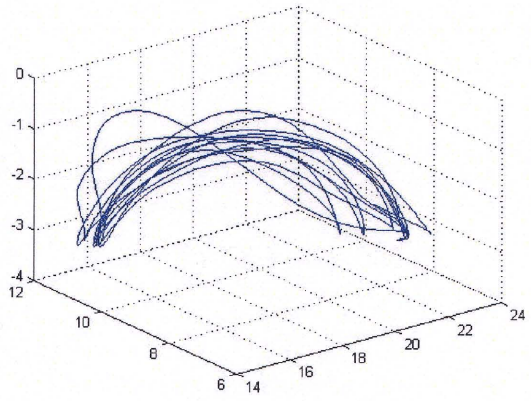


Figure A.2 X, Y, Z Plot of Stylus

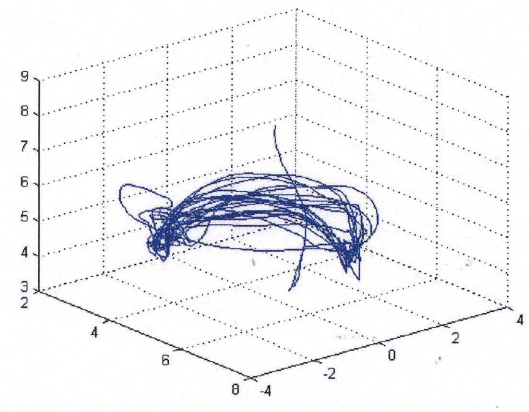


Figure A.3 X, Y, Z Plot of VR Stylus

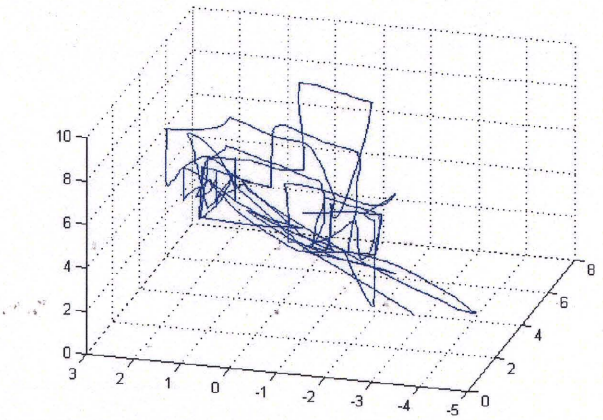


Figure A.4 X, Y, Z Plot of Spaceball

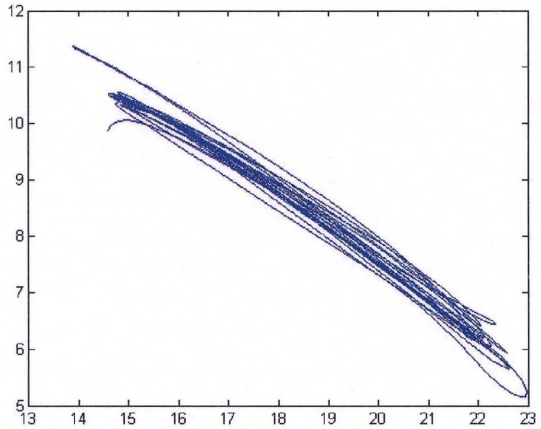


Figure A.5 X, Y Plot of Finger

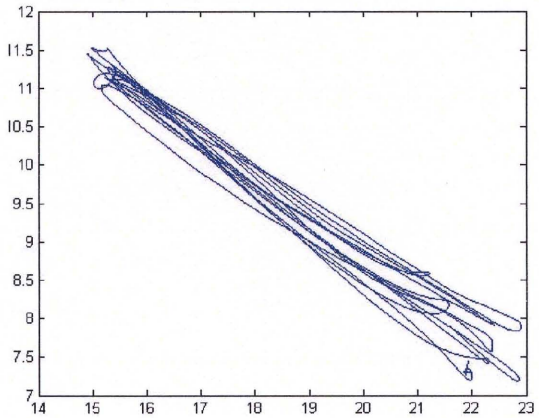


Figure A.6 X, Y Plot of Stylus

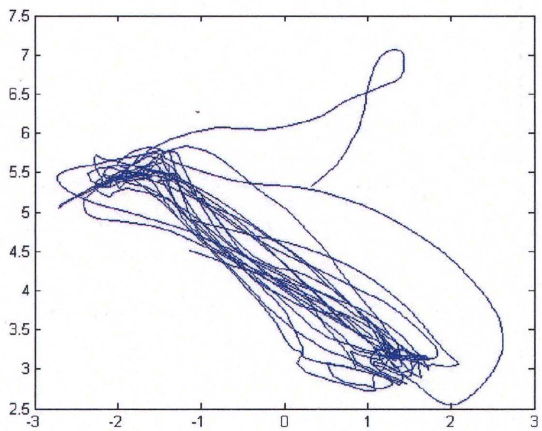


Figure A.7 X, Y Plot of VR Stylus

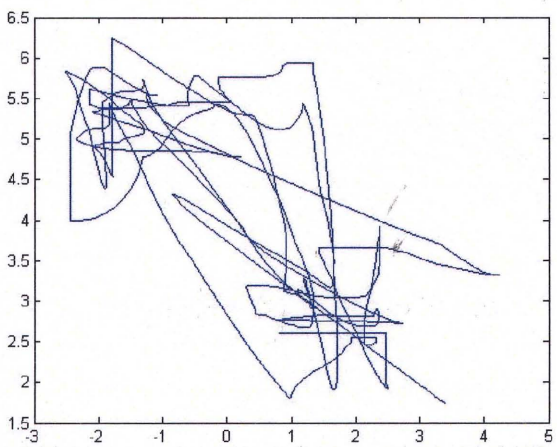


Figure A.8 X, Y Plot of Spaceball

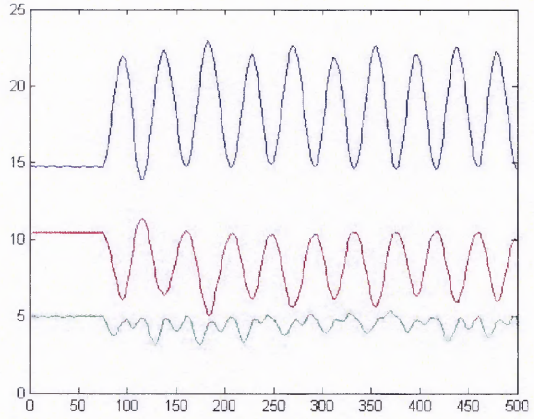


Figure A.9 X, Y, Z vs. Time of Finger

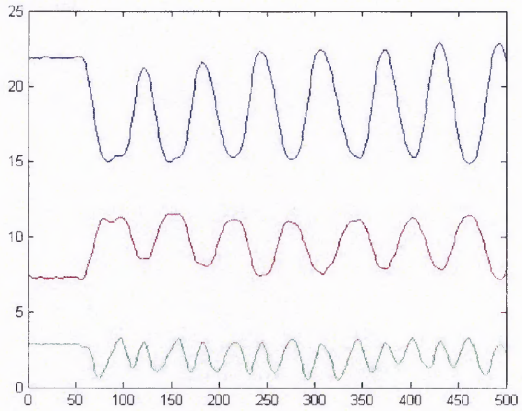


Figure A.10 X, Y, Z vs. Time of Stylus

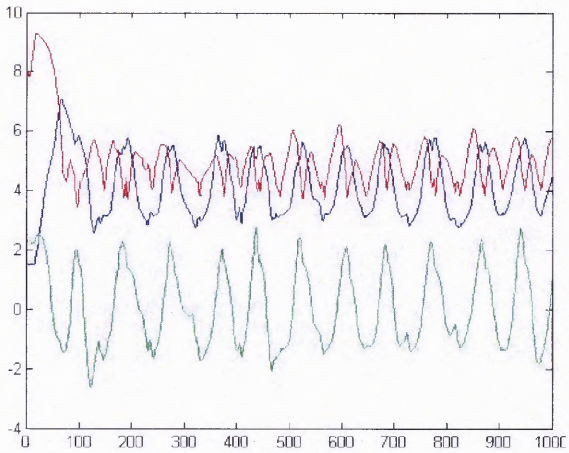


Figure A.11 X, Y, Z vs. Time of VR Stylus

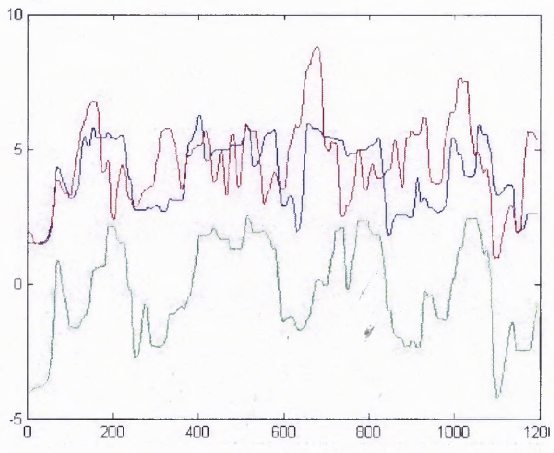


Figure A.12 X, Y, Z vs. Time of Spaceball

APPENDIX B

SAMPLE TRAJECTORIES FROM BOX AND BLOCKS TESTING

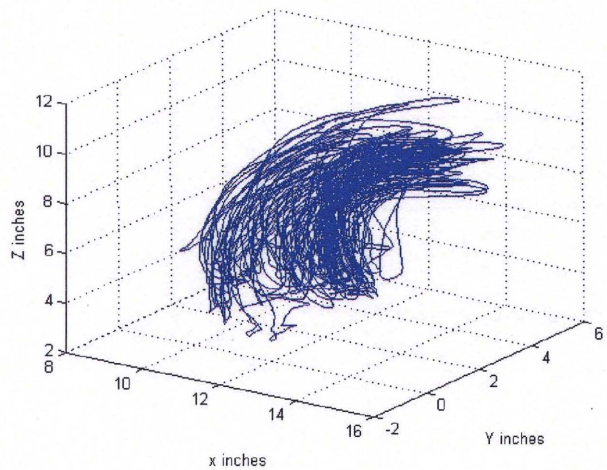


Figure B.1 X, Y, Z Plot of Finger Movement

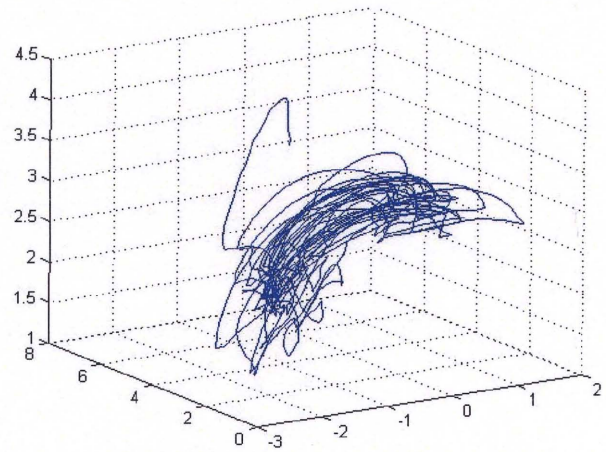


Figure B.2 X, Y, Z Plot of VR Stylus

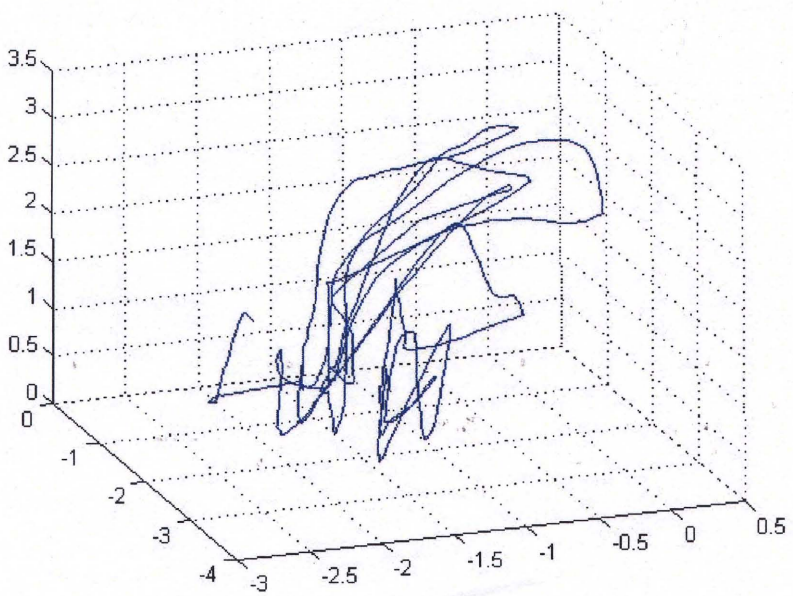


Figure B.1 X, Y, Z Plot of Spaceball

APPENDIX C

SCREEN SHOTS OF VR WORLDS

Figure C.1 Virtual Kitchen with movable utensils and jell-o cubes.

Figure C.2 Virtual Office with computer, Desk, etc.

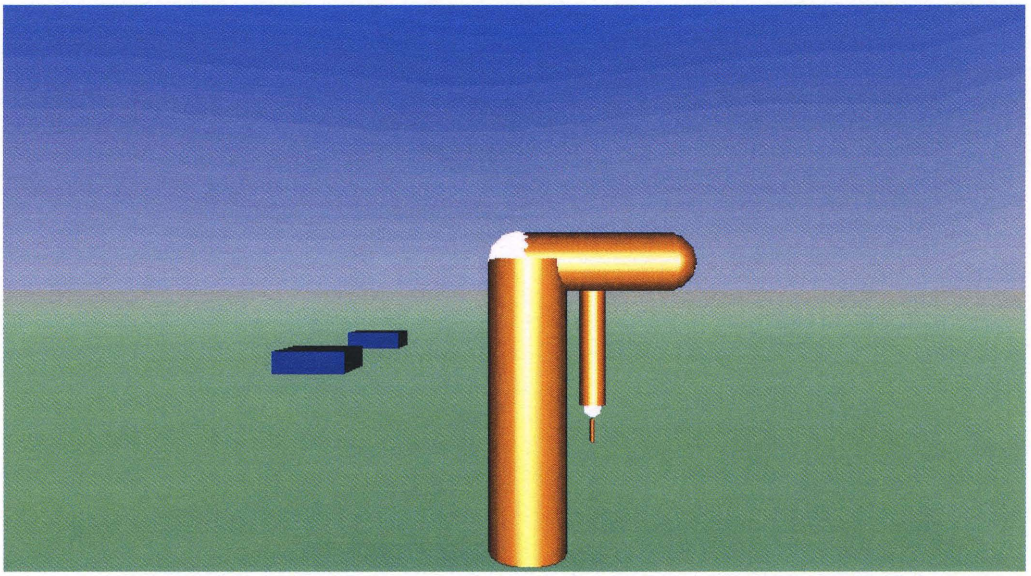


Figure C.3 Sample Fitts Law World used for Experiments

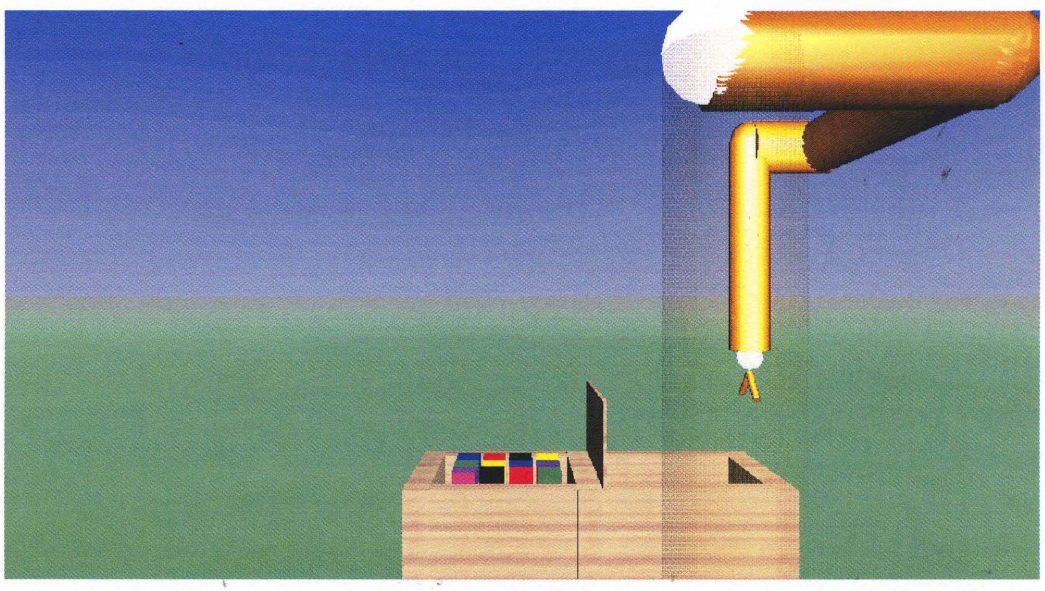


Figure C.4 Virtual Box and Blocks World used for Experiments

REFERENCES

1. Hillman, M. "Rehabilitation robotics from past to present – a historical perspective." *Proceedings of the ICORR 2003, The Eight International Conference of Rehabilitation Robotics*, April 23-25, 2003.

2. M. Topping and J. Smith. "The development of Handy 1: A robotic system to assist the severely disabled." *Technology & Disability*, 1999, Vol. 10 Issue 2, p95.

3. M. Topping, H. Heck, G. Bolmsjo, D. Weightman. "The Development of RAIL (Robotic Aid to Independent Living)" *Proceedings of the Third TIDE Congress.* 1998.

4. H.F.M. Van der Loos, J.J. Wagner, N. Smaby, K. Chang, O. Madrigal, L.J. Leifer, O. Khatib. "ProVAR Assistive Robot System Architecture." *Proceedings from the IEEE International Conference on Robotics and Automation.* May, 1999.

5. Evans, J.M.; "HelpMate: an Autonomous Mobile Robot Courier for Hospitals." *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, 'Advanced Robotic Systems and the Real World.'* September 12-16, 1994. Volume 3, Page(s):1695 – 1700.

6. Yoshiyuki Takahashi, Takashi Komeda, and Hiroyuki Koyama. "Development of the Assistive Mobile Robot System: AMOS – To Aid in the Daily Life of the Physically Handicapped." *Advanced Robotics.* November, 2003.

7. Wiegner, A.W., Taylor, B., Sheredos, S.J. "Clinical evaluation of the Helping Hand Electromechanical Arm." *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE.* Oct 31 - Nov 3, 1996. Volume: 2, On page(s): 541-542

8. Farahmand, F.; Pourazad, M.T.; Moussavi, Z.; "An Intelligent Assistive Robotic Manipulator." *27th Annual International Conference of the Engineering in Medicine and Biology Society*, 2005. Page(s):5028 – 5031

9. Driessen, B. J. F., Evers, H.G., Woerden, J.A. v, "MANUS – a wheelchair-mounted rehabilitation robot." *Proc Instn Mech Engrs.* 2001. Volume 215. Part H.

10. Ellis, S.R. "What are Virtual Environments." *IEEE Computer Graphics and Applications.* January 1994. Volume 14, Issue 1, Page(s):17 – 22.

11. Draper, J.V.; Handel, S.; Hood, C.C.; Kring, C.T. "Three Experiments With Stereoscopic Television: When It Works And Why." *IEEE International Conference on Systems, Man, and Cybernetics.* October 13-16, 1991. Volume 2, Page :1047 – 1052.

12. Drascic, D.; Milgram, P.; Grodski, J. "Learning effects in telemanipulation with monoscopic versus stereoscopic remote viewing" *IEEE International Conference on Systems, Man and Cybernetics.* Nov. 14-17, 1989. Volume 3, Page:1244 – 1249.

13. Wai-keung Fung; Wang-tai Lo; Yun-hui Liu; Ning Xi. "A Case Study of 3D Stereoscopic vs. 2D Monoscopic Tele-Reality in Real-Time Dexterous Teleoperation." *IEEE/RSJ International Conference on Intelligent Robots and Systems.* Aug 2-6, 2005. Page 181 – 186.

14. Fitts, P. M., "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement." *Journal of Experimental Psychology.* June, 1954. Volume 47, No. 6. 381- 391.

15. Fitts, P. M., Peterson. "Information Capacity of Discrete Motor Responses." *Journal of Experimental Psychology.* February, 1964.Volume 67, No. 2. Page 103 – 111.

16. MacKenzie, I. S. "Fitts' Law as a Research and Design Tool in Human-Computer Interaction." *Human-Computer Interaction.* 1992. Volume 7, Page 91-139.

17. van Tuijl, J.H., Janssen-Potten, Y.J.M., Seelen, H.A.M. "Evaluation of Upper Extremety Motor Function Tests in Tetraplegics." *Spinal Cord* January 29, 2002. Volume 40, 51-64.

18. Mathiowetz, V. Volland, G. Kashman, N. Weber, K. "Adult Norms for the Box and Block Test of manual dexterity." *American Journal of Occupational Therapy.* 1985. Volume 39, Page 386-391.

19. Immersion Probe™ and Personal Digitizer™. "Programmer's Technical Reference Manual: Immersion Probe and Personal Digitizer." Revision 2.0a. Immersion Corporation, Santa Clara, CA.

20. Ramirez, Diego. "Multi-Degree of Freedom Telemanipulation in an Unstructured Environment." *Masters Thesis*, NJIT. January, 2007.

21. Humusoft s.r.o. and The Math Works, Inc. "Virtual Reality Toolbox: For use with Matlab® and Simulink®." User's Guide. August 2001.

22. Carey, R., Bell, G. "The Annotated VRML97 Reference Manual." http://www.cs.vu.nl/~eliens/documents/vrml/reference/BOOK.HTM. Feb. 2008 Copyright © 1997 by Rikk Carey and Gavin Bell.

23. Corke, P. I. "A Robotics Toolbox for Matlab (Release 7)." *IEEE Robotics and Automation Magazine.* March 1996.Volume. 3. Number 1. Page 24-32.

24. Ungerleider, Leslie G., Doyon, Julien, Karni, Avi. "Imaging Brain Plasticity during Motor Skill Learning." *Neurobiology of Learning and Memory.* 2002. Volume 78, 553-564.