ABSTRACT

CREATING NEW VISUALIZATION AND HUMAN INTERFACE DEVICES FOR THERAPEUTIC VIDEO GAMES

by
Kunal Jayant Doshi

Virtual reality (VR) gaming environment as a tool for rehabilitation of patients with upper extremity disorders is fast gaining momentum. VR based motor training systems provide an engaging, motivating and adaptable environment where the motion of the limb displayed in the virtual world is a replication of the motion produced in the real world by the patient's extremity.

The goal of this thesis was to create a generic gaming system which can be interfaced to a number of different Human interface devices (HID) and produce rich graphics to create a virtual environment which closely resembles the real world. This would overcome the current limitations of the 'HANDS UP' game developed by the Neuromuscular lab which accepts only a web camera input and uses color marker detection to recreate the limb movements in simple two dimensional environment.

Three dimensional worlds designed in Virtual Reality Modeling Language (VRML) were controlled using SIMROBOT and Virtual reality toolbox in MATLAB to create better visualization. The Human Interface devices currently used for Virtual Reality video games are expensive and cannot be afforded for home use. Various new HID's like the Flock of Birds, Nintendo Wiimote and IMU 6 DOF V3 were tested for their use in the virtual gaming environment. Each device presented their own set of advantages and problems. The thesis work involved understanding and resolving these problems and interfacing the devices with the gaming system.

CREATING NEW VISUALIZATION AND HUMAN INTERFACE DEVICES FOR THERAPEUTIC VIDEO GAMES

by
Kunal Jayant Doshi

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

January 2008

APPROVAL PAGE

CREATING NEW VISUALIZATION AND HUMAN INTERFACE DEVICES FOR THERAPEUTIC VIDEO GAMES

Kunal Jayant Doshi

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

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

Dr. Lisa Simone, Committee Member — Date
Research Professor of Biomedical Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Kunal Jayant Doshi

Degree: Master of Science

Date: January 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,
  Mumbai University, Mumbai, India, 2006

Major: Biomedical Engineering

To my beloved family and my late grandmother, 'Jayaben Doshi'

ACKNOWLEDGMENT

I would like to thank Dr. Richard A Foulds, my thesis advisor for guiding me throughout my Master's at NJIT. He has always provided me with novel ideas, constant support and encouragement whenever I was stuck. His confidence and belief in me was the guiding force towards my achievement.

I would like to thank Dr. Sergei Adamovich for all his valuable advice. He was kind enough to let me use his lab facility for my research and on my committee. I would also like to thank Dr. Lisa Simone for being on my committee.

Special thanks to Qinyin Qiu and Abraham Mathai who are working in my lab for the innumerable suggestions and help that I received from them. Finally, I would like to thank my parents for their continued words of advice.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

1.1 Background

Stroke and Cerebral palsy (CP) are two leading causes of upper extremity disorders. The patients with these disorders normally suffer from muscle weakness, loss of range of motion and reduced motor control. These individuals are unable to perform everyday tasks like grabbing objects, walking, eating and speaking. This limits the ability of such patients to perform their daily activities and live an independent life. In individuals with cerebral palsy, the stronger limb tends to compensate for the weaker limb rather than involving the weaker limb. This leads to muscle contracture. Muscle contracture is defined as shortening of the muscle fiber due to reduction in the number of sarcomeres. It occurs when the muscles are not used through their normal range of extension and flexion. Recent studies have shown that intensive and repetitive therapeutic intervention can lead to improved sensory motor skills in stroke patients [1-5]. Initiating movements in the extremities that would otherwise not be used creates new patterns of neuronal activation. This leads to neural reorganization [7]. Intensive training encourages healthy neurons to assume role of the damaged cells resulting in improved sensory-motor function [8-10]. This ability of the brain to reorganize itself by forming new neural connections is called as neuroplasticity. In traditional therapy, patients are normally seen for half- hour sessions once or twice a day. This is decreased to once or twice a week for outpatient therapy [11]. As a result, these techniques fail to provide the required amount or intensity of practice needed to effect neural and functional change [11]. Also there is a

huge cost factor involved in this technique due to costly consulting fees and hospital facilities.

Virtual Reality (VR) as a tool for Rehabilitation of patients with upper extremity disorders is fast gaining momentum as it can be used as a tool for repetitive motor training in a more interactive way. VR based motor training systems provide an engaging, motivating and adaptable environment where the motion of the limb displayed in the virtual world is a replication of the motion produced in the real world by the patient's extremity. A sense of authority over the virtual hand is developed which increases the confidence among users about the ability of their hand. Due to the reduced intervention by the physiotherapist such systems can be installed at the patient's home enabling increased therapy time while reducing the commuting hassle for the patients. VR offers an opportunity to bring the complexity of the physical world into the controlled environment of the laboratory. Brain plasticity is more likely when the individuals voluntarily begin an attempt to control the movement. Neural learning occurs faster due to visual and proprioceptive feedback. A repeated use of the joints to the extremes of the compromised range of motion can help in increasing the range of motion of the limbs [12].

Denise Reid studied the effect of VR on children with CP. These individuals can use VR system to practice and try out new skills and movements without the worry of embarrassment or the risk of injury [13]. This in turn can lead to improved motor performance and a sense of personal control or self-efficacy. The study of influence of virtual reality on kids with cerebral palsy shows that VR creates an environment encouraging involvement, motivation and playfulness among children. Also, the aim of

the VR technology is to make the rehabilitation therapy more economical to the patients because the success of any rehabilitation technique for the masses would largely depend on the cost that is involved with it. Subject involvement is the key driving force behind the success of VR as a rehabilitation technique. It is a known fact that there is faster neuronal recovery if the subjects are involved in the task. The willingness of the subjects to complete the task leads to a shorter recovery period.

The use of the robotic manipulator for rehabilitation of subjects with stroke was demonstrated by Krebs et al. [14]. A combination of physical intervention with virtual reality was described by Jack et al. in which subjects with stroke interacted with virtual objects that were presented using computer graphics; a force reflecting glove was used for haptic feedback [11]. Krueger in his study of a low immersion virtual reality system called VideoPlace, had found high motivations among users to control movements of their silhouette to participate in these video games. He believed that individuals with disabilities could be motivated to push their capabilities in order to interact with a virtual environment.

Commercially available games using joystick or keyboard cannot be used for rehabilitation engineering because it requires very quick and precise movements which the patients with upper extremity disorder do not posses. Also these games do not provide the range of flexion and extension necessary to address the issue of muscle contracture. An ideal VR based gaming system should be able to use the available range of motion of the patient. Also it should motivate the user for their effort by techniques such as applauding on completion of each task and awarding points.

Researchers at Massachusetts Institute of Technology have developed the IVE (Interactive Video Environment) and ALIVE (Artificial Life IVE) systems which separates the user's silhouette from the background. The users see themselves in their physical surrounding along with the computer generated characters. Sony's Eye-toy is a low-immersion virtual reality gaming system interface to its playstation technology. It uses a USB camera to capture the user's image which is integrated into a software generated background. Current VR systems are expensive and require specialized equipment and software. Also they are available with only a limited number of games and simulations involving specific built-in tasks like playing the piano, stacking blocks, or catching birds. Repetition is found to be boring by the subjects once the specified tasks and simulations are mastered by them [15]. A VR system should therefore be able to be modified and made more difficult to prevent this boredom.

The objective of the Neuro-muscular lab at NJIT is to create an inexpensive, motivating gaming platform for upper extremity disorders which can be customized for every subject considering the available range of motion. Also the gaming console should be easily modifiable and upgradeable to meet different therapeutic targets. As an effort in this direction the lab has developed a game called 'HANDS UP'. This game was designed for children with orthopedic disorders. The game tracks the hand motion using web camera as the input. Using pattern recognition the game determines the position of the color markers on the hand. This virtual interaction system promotes cortical reorganization, as well as discourages unwanted changes in the muscle tissue that result in contracture. The behavior of the computer generated graphics can be modified to optimize the virtual environment for individuals with limited range of motion [6].

The game was designed using the SIMROBOT toolbox in MATLAB. However, the current system accepts only webcam as the input. The limitation of the webcam based system is that it necessitates the hand motion be done in front of the webcam. The game takes into account only the translation in two planes. Hence it gives no perception of depth or the rotations about the three axes. The SIMROBOT toolbox uses simple MATLAB graphics which does not create a very realistic virtual environment. Also since it has a refresh rate of only 5 frames/second the movement is jittery. Due to these limitations of the current system there exists a requirement for better generic gaming systems which can be interfaced to a number of different interfaces and provide more realistic graphics.

1.2 Objective

The advancements in Micro-Electro Mechanical Systems (MEMS) technology has led to the development of extremely small and highly precise sensors. These sensors are also relatively inexpensive. The human hand has six degrees of freedom, three translational and three rotational motions. A combination of accelerometers and gyroscopes can be used to measure these motions in 3-D space. The goal of this thesis was to evaluate the performance of devices made from these sensors as a possible input to the VR games. The relatively inexpensive nature of these sensors makes them favorable for use in development of an inexpensive VR rehabilitation gaming console for home use. The use of VRML (Virtual Reality Modeling Language) as a tool for producing better 3-D visualization of the VR world was evaluated. The goal of the thesis is to create a test bed

which can be used to explore the benefits of using high quality graphics and six degrees of freedom input.

1.3 Outline of the project

This thesis is organized as follows: Chapter 2 outlines the development of a new graphical interface using VRML. Chapter 3 involves evaluating the Flock of Birds [16] as an interface to test the new graphics and also as a gold standard for evaluating other interfaces. It also involves a survey of the current interfaces available in the market. Interfacing the Ninentendo Wii [18] and understanding and its practical limitations as an interface for the game is discussed in chapter 4. Chapter 5 describes the IMU 6DOF [17] and the working principle of its components. Problems faced while working with the IMU and work done towards rectifying them is discussed in Chapter 6. Appendix contains the code written.

CHAPTER 2

DEVELOPMENT OF A NEW GRAPHICAL INTERFACE USING VRML

SIMROBOT displays the environment using the plot command in MATLAB, this not only results in poor graphics but also the display appears jittery. This is because the plot command erases the entire previous plot and plots the new environment again. Depending upon the complexity of the program, the display of the environment is updated periodically. In case of complex algorithm there is large time gap between two plot commands resulting in a game where the objects tend to jump in-between moves rather than having a smooth transition. The refresh rate of the computer monitor screen is also another factor that affects the game display. With MATLAB graphics there is no depth perception possible as only two dimensional objects are used. Hence in order to provide a game display which is more immersive, realistic and allows representation of depth and rotations about the three axes, SIMROBOT toolbox is modified to call and control a three dimensional world developed in Virtual Reality Modeling language (VRML).

2.1 SIMROBOT

SIMROBOT (SIMulated ROBOTs) is an autonomous mobile robotics toolbox which allows the user to simulate the behavior of one or more mobile robots, moving in virtual environment. It was built as a graduate thesis at the Brno University of Technology, Department of Control Measurement and Instrumentation (Petrinic, 2005).Each robot can be equipped with several virtual sensors, like ultrasonic sensor and laser scanner, and can

be driven by its own control algorithm. It has a collision detection algorithm to detect the collision of the robots among themselves and with the environment.

2.2 VRML

VRML an acronym of Virtual Reality Modeling Language is a file format for describing interactive three-dimensional objects and worlds. VRML is a platform-independent language for describing three dimensional interactive virtual worlds linked with the World Wide Web. VRML was designed as a tool to minimize the size of files and to provide a means of modeling more complex objects with primitives, without having to overload the net with large files. Objects can be built from solid shapes, from text, or from primitive points, lines, and faces. Textures (2-D patterns) and optical material properties which affects how they interact with the lights in the world can also be applied to them. Objects can be grouped into more complex objects, used multiple times, translated, and rotated. They can also be used to trigger events, which can be routed to other events or to scripts written in higher languages like MATLAB, C, JavaScript and Java. Within VRML it is possible to trigger sounds, move objects along paths, and link to HTML or VRML targets.

In VRML, objects are called as nodes. Each node has fields like translation, rotation, scale and children with which its position, orientation, appearance in space can be controlled. VRML worlds have an extension, '.wrl'. It can be viewed in a web browser using free VRML viewers like Cortona ® VRML Client from Parallel graphics and Blaxxun contact from Blaxxun®.

2.2.1 V-Realm Builder 2.0

V-Realm Builder from Integrated Data Systems Inc. was used for creation of VRML world. It provides a more interactive way of creating virtual worlds compared to text editors. Hence more complex worlds can be created easily using V-Realm Builder. Also modification made to the fields can be viewed instantly. The figure below shows a view of the V-Realm Builder.

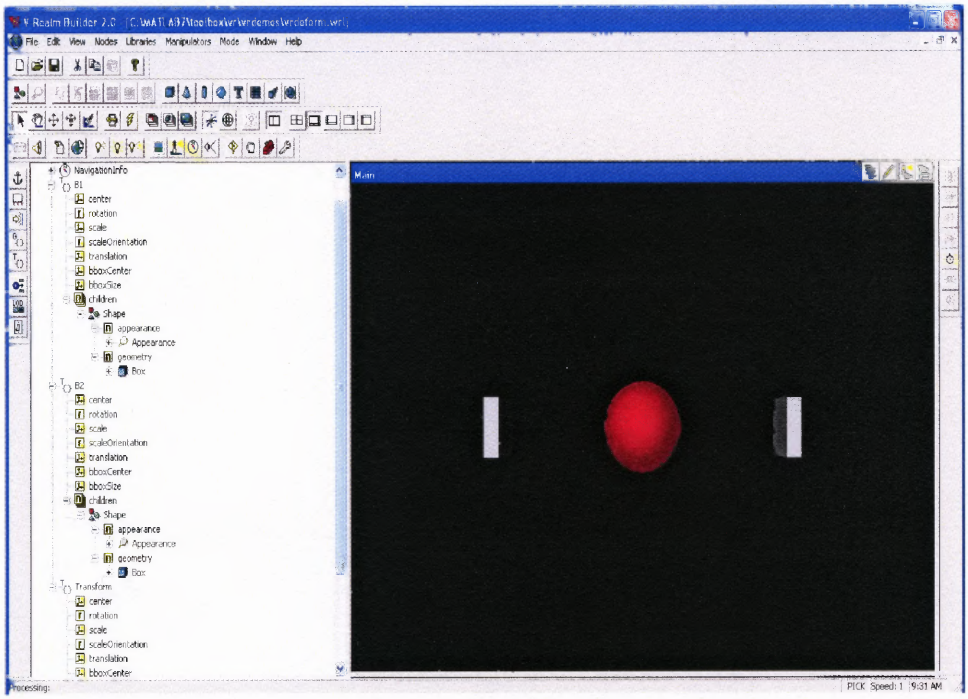

Figure 2.1 VREALM Builder.

2.2.2 Virtual Reality Toolbox

The Virtual Reality Toolbox serves as an interface between VRML world and MATLAB or SIMULINK. It provides MATLAB or Simulink access to fields of VRML nodes to control the position, rotation, and dimensions of the 3-D images defined in the virtual reality environment. The Virtual worlds can be viewed either using the Virtual Reality Toolbox viewer or the Blaxxun Contact plug-in for Web browsers.

2.3 Toolbox Modification

SIMROBOT toolbox's structure was studied and its code was decrypted. The algorithm of the robots has been modified has been modified so that it can accept input from Human Interface Devices (HID) like the Flock of Birds, IMU 6 DOF so that the objects can be controlled with the real time data coming in from these devices. Since the input is taken from a HID the input data is first filtered to compensate for the oscillations of the human hand. Simple threshold technique is used to eliminate all variations in the input data below a certain pre-determined amount. At initialization, the X and Z coordinates of the HID correspond to the center coordinates of the SIMROBOT. All new positions are calculated relative to these initial coordinates.

A scaling factor is used to convert the HID coordinates to SIMROBOT coordinates. The significance of this scaling factor is that it allows setting of the sensitivity of the game response to the input. A larger scaling factor would allow traversing the entire environment using a small change in the HID coordinates while a small scaling factor would require a larger hand movement to traverse the entire environment. The scaling factor can therefore be customized according to individual subjects taking into consideration their range of hand motion. The scaling factor can also be reduced with time, requiring the subject to put in more efforts gradually. The scaling factor can be used to increase the difficulty level and as a tool to predict the effectiveness of the therapy. At initialization the coordinates of HID correspond to center coordinates of SIMROBOT environment. As a result the robot controlled by the HID will always begin from the center of the world. This is required because all translations of the HID are calculated with respect to its initial position.

The formula for conversion from HID coordinates to SIMROBOT coordinates is:

X position coordinate of simrobot = (Change in X position of HID) * scaling_factor + Center point of simrobot (1)

The SIMROBOT coordinates calculated from the above formula are clipped to the extreme value of the SIMROBOT environment in case the calculated value exceeds the extreme values of the environment. This step is done to prevent it from going out of the bounds of the environment. Algorithms of all the robots in SIMROBOT are executed one after the other in a sequential order and the parameters like position, velocity, size and heading are updated. The robots can have one or more sensors present on them to detect the presence of another robot or an obstacle in the environment in the range of the sensor. SIMROBOT also provides an ability to detect collision of a robot with another robot on with an obstacle.

The VRML world is initialized by the SIMROBOT and the objects in the VRML world assume a position corresponding to robots in the SIMROBOT environment. The formula used to convert SIMROBOT coordinates to VRML coordinates is given as:

(X position in VRML) = (X position in SIMROBOT) * (2 * (Maximum X position in VRML viewpoint) / (X size of the SIMROBOT matrix)) – (Maximum X position in VRML viewpoint) (2)

Since VRML world is an infinite 3-D space, the extreme coordinates of the world in view are determined by the viewpoint. The VRML world has positive and negative coordinates whereas SIMROBOT has only positive coordinates. The maximum positive

value is subtracted from the calculated position so that there are both positive and negative values possible corresponding to SIMBROBOT position.

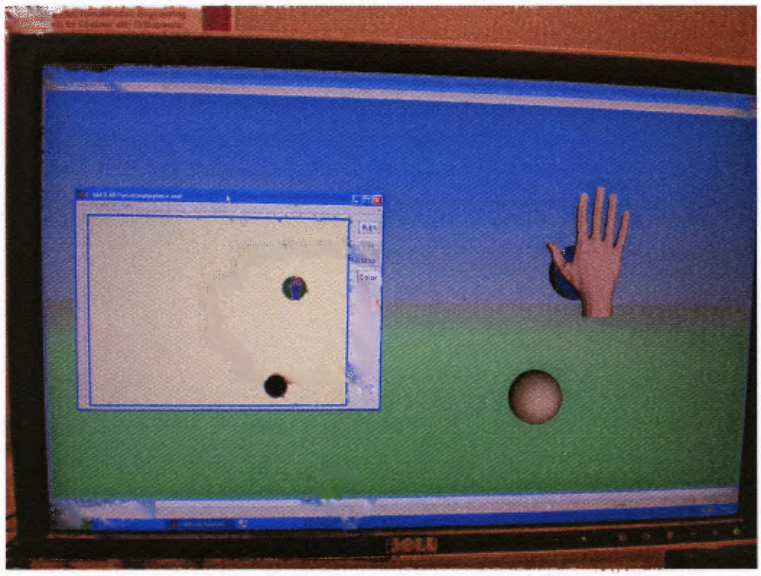

Figure 2.2 Correspondence between objects in VRML and SIMROBOT.

The figure shows the position of the hand in the SIMROBOT world and the corresponding position of the hand in the VRML world calculated using the formula devised. In a typical game the SIMROBOT initializes the HID device and VRML world at start up. Algorithms of each robot in the world are executed sequentially. Collision detection program checks for collision between robots or robot and the SIMROBOT environment. The VRML world is updated with the new positions of the SIMROBOT robots. This process is executed repeatedly in a loop till the end of game is reached. The HID device is stopped and the VRML world is closed.

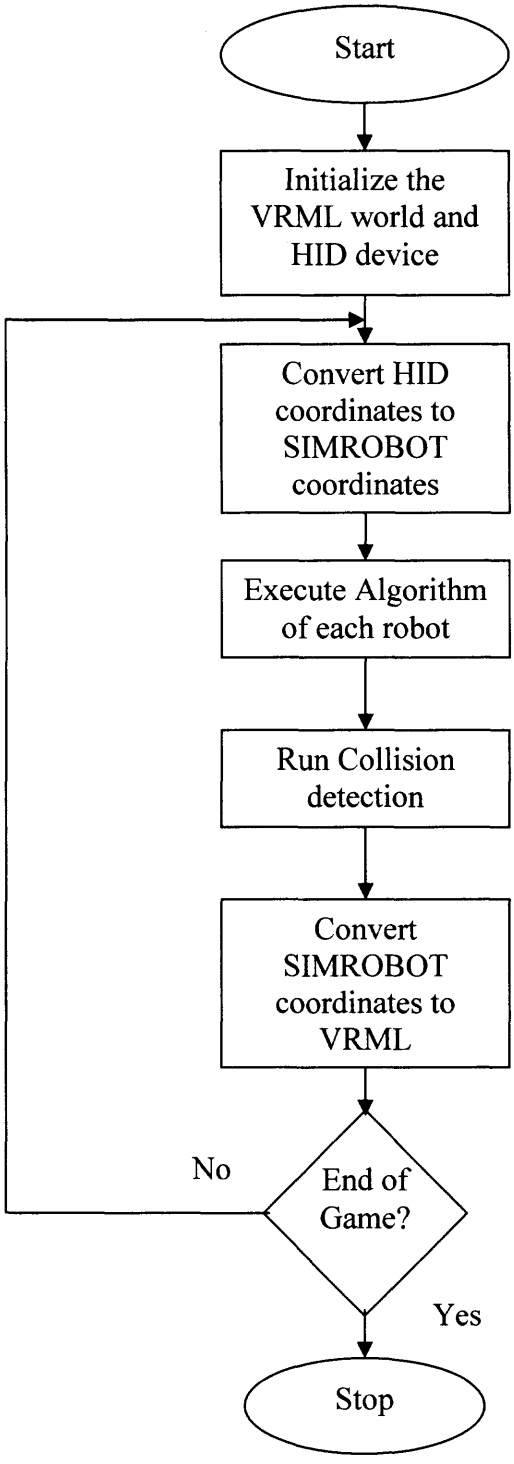

Figure 2.3 Typical flowchart of a game.

CHAPTER 3

INVESTIGATING THE FLOCK OF BIRDS®

3.1 Flock of Birds®

Flock of birds® from Ascension Technology Corporation, VT, USA, is a six degree of freedom tracking device which can simultaneously track the position and orientation of up to thirty receivers using a single transmitter. Each sensor is capable of making from 20 to 144 measurements per second of its position and orientation when it is located within ± 4 feet of its transmitter. The FOB determines the position and orientation by transmitting a pulsed DC magnetic field that is measured by receivers in the flock. From the measured magnetic field characteristics the receiver independently calculates the position and orientation and makes it available to the host computer. The Bird communicates with the computer through serial communication [16].

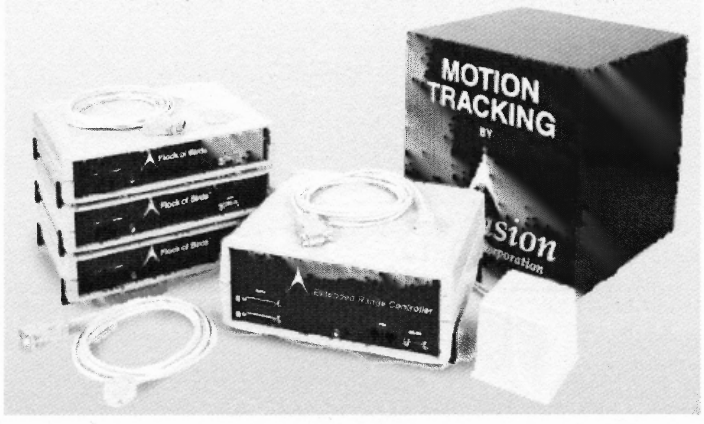

Figure 3.1 Ascension's Flock of Birds®.
(http://www.ascension-tech.com/products/flockofbirds.php)

3.2 Implementation of a game using FOB

A two degree of freedom game was implemented using the FOB and the new VRML graphical interface. The objective of the game in the picture below was to pick the ball and place it over the circular hole in the bottom right hand corner. The Virtual hand can be moved using the X and Y position data of the FOB. The algorithm of the ball was designed such that it follows the hand movement when the hand is moved over. Hence in the game the hand is moved over the ball and then used to guide the ball into the hole.

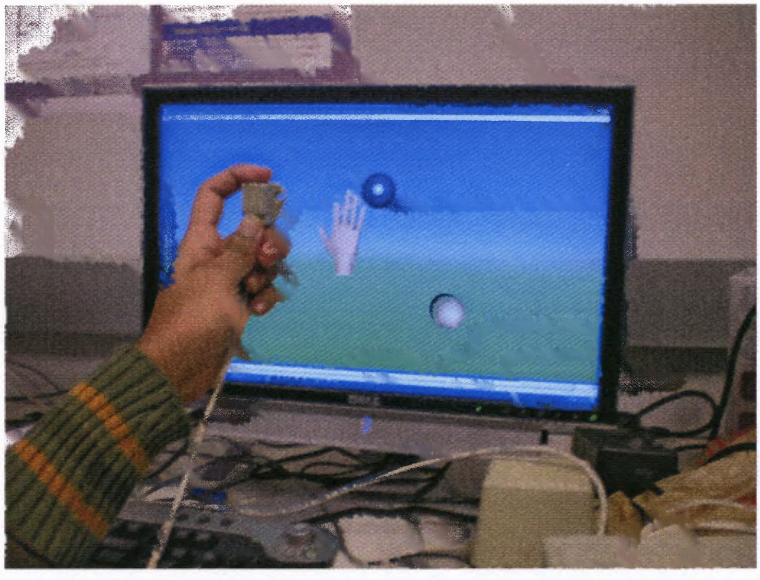

Figure 3.2 Simple game displaying hand motion tracked in VRML using FOB.

3.3 Current interfaces available in market

FOB provides good X, Y, Z, pitch, roll and yaw values at 100 frames/sec. However, the high cost of FOB makes it unfavorable for its implementation as the ideal HID for the neuro-rehabilitation games for home application. Also the wired nature of the sensor limits the mobility of the subjects. The FOB was chosen as a gold standard which will be

used to test the accuracy of other HID devices in estimating the translations and rotations in space.

3.3.1 Haptic Master

Haptic master from MOOG FCS, based in Netherlands, is a three degree of freedom force-controlled haptic interface. It provides user with haptic sensation feedback to closely simulate the weight and force found in a variety of human tasks.

Figure 3.3 Haptic Master from MOOG FCS.
(http://www.fcs-cs.com/robotics/products/hapticmaster)

3.3.2 The CyberGlove® II System

The Cyberglove® from Immersion technologies, California, USA, measures 22 joint angles with high accuracy using resistive bend sensors. It has three sensors per finger, four abduction sensors, a palm-arch sensor and a sensor to measure flexion and abduction. It enables real-time estimation of hand and finger motion.

Figure 3.4 The CyberGlove® II System: Wireless Data Glove from Immersion.
(http://www.immersion.com/3d/products/cyber_glove.php)

3.3.3 Space Navigator

Space Navigator from 3DConnexion, California, USA, is a 6 degree of freedom 3D mouse. It has advanced six degree of freedom optical sensor. In addition it provides two programmable buttons for additional functionality.

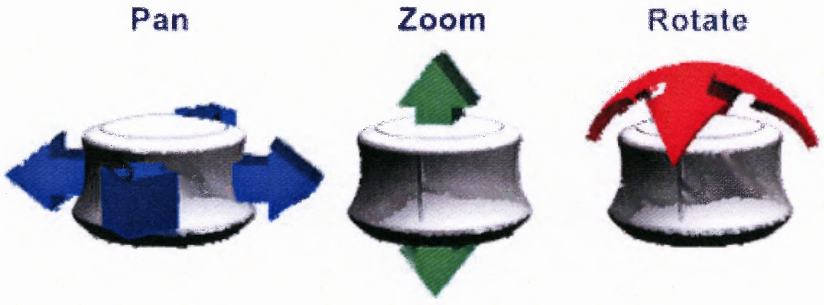

Figure 3.5 Space Navigator from 3DConnexion.
(http://www.3dconnexion.com/3dmouse/spacenavigator.php)

3.3.4 Moven

Moven from Xsens motion technologies, Netherlands, contains 16 inertial motion trackers, integrated in a comfortable lycra suit. The inertial motion trackers give absolute orientation estimates, which can also be used to calculate the 3-D linear accelerations in world coordinates. These accelerations in turn can be used for translation estimates of the body segments.

Figure 3.6 Moven suit from Xsens.
(http://www.moven.com/en/home_moven/product/product_overview.php)

3.3.5 Head Mounted Display

Head mounted displays involve goggles or helmets worn on the subject's head to display the virtual environment. 2-D or 3-D world can be displayed. It also has provision for stereo sound.

Figure 3.7 The 5DT HMD 800 series of head mounted display.
(http://www.5dt.com/products/ihmd03.html)

3.3.6 Sensable Phantom Desktop

The Phantom Desktop is a product of INITION based in London, UK. It consists of a stylus connected to links. It provides 6 degrees of freedom position sensing with respect to the base. It connects to the computer through a parallel port interface.

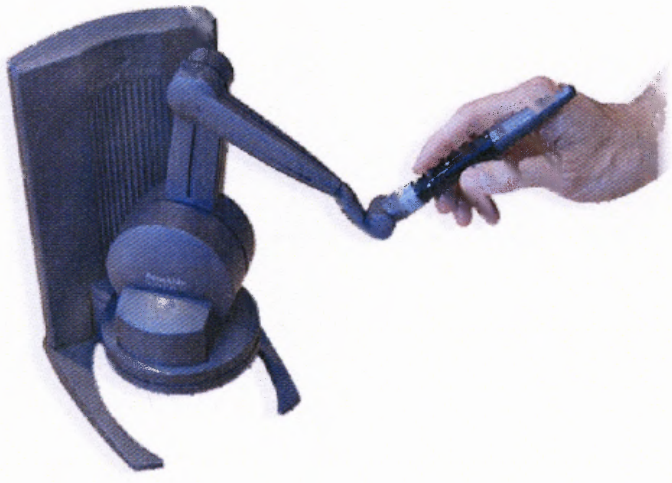

Figure 3.8 PHANTOM® Desktop™ from SENSABLE.
(http://www.inition.co.uk/inition/product.php?URL_=product_ffhaptic_sensable_phantomdesktop&SubCatID_=36)

The advancements in the MEMS technology have led to development of ultra small and very accurate sensors. Also the growth and developments in the wireless technology has not only slashed the price of these devices but also their size has shrunk making wireless telemetry on small devices viable. It is now possible to have position and angle sensing devices coupled with wireless connection integrated onto devices as small as a match box.

CHAPTER 4

INVESTIGATING THE WIIMOTE

4.1 Wii

Wii is the latest gaming console from Nintendo. It has a Wiimote which connects to the gaming console through a Bluetooth connection. The Wiimote incorporates a three axis accelerometer, IR sensor, speaker and a motor which vibrates to provide a rumble feedback [18]. The inbuilt 3-axis linear accelerometer is ADXL330 from Analog devices. The device is rated for +/-3g with 10% sensitivity. Inside the chip is a small micro mechanical structure which is supported by springs built out of silicon. Differential capacitance measurements allow the net displacement of the tiny mass to be converted to a voltage, which is then digitized. Thus, the Wii can sense the change in accelerations in the three dimensions. Integration of these accelerations provides velocity information which can be integrated to get three degrees of freedom: pitch, roll and yaw.

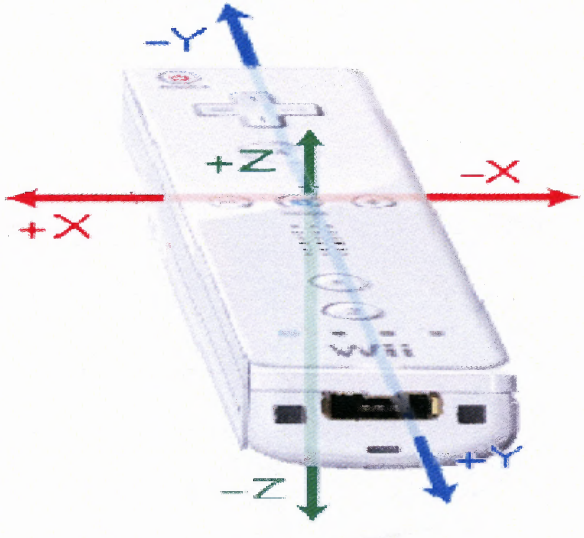

Figure 4.1 The Wiimote with the directions of the accelerometer.
(http://www.wiili.org/index.php/Wiimote)

4.2 Interfacing the Wii with MATLAB

The Wiimote pairs with the Wii console when its button labeled 1 and 2 are pressed simultaneously through a Bluetooth connection. The Wiimote can be connected to the personal computer by using a Bluetooth dongle and Bluesoleil software from IVT Corporation.

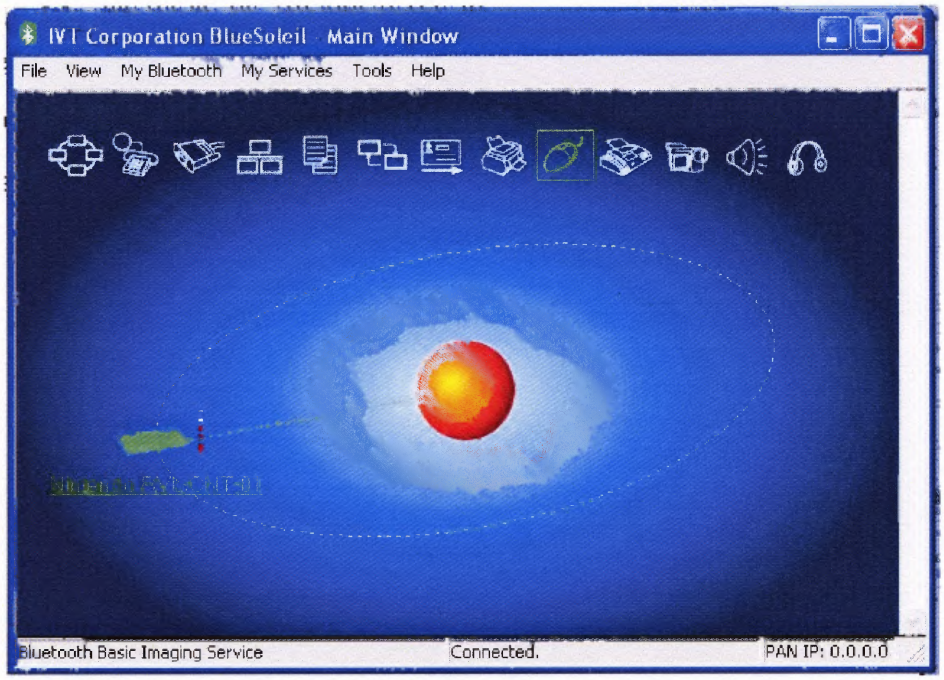

Figure 4.2 Wii connected to the computer using IVT BlueSoleil software.

MATLAB does not provide support for connecting with devices having a Bluetooth connection. Hence a dll file was written in MATLAB which calls a C program which can access the data coming from the Wiimote. The dll file returns the current reading on the accelerometer when invoked.

4.3 Application of Wii

The three accelerometers of Wii measure the accelerations in X, Y and Z axis respectively. Since the double integration of acceleration in theory gives displacement we can calculate the translation of Wii in space with respect to its initial position.

The displacement equations are given as,

$$X_{n+1} = \iint (\text{acceleration in X direction}) + X_n \quad (3)$$

$$Y_{n+1} = \iint (\text{acceleration in Y direction}) + Y_n \quad (4)$$

$$Z_{n+1} = \iint (\text{acceleration in Z direction}) + Z_n \quad (5)$$

4.4 Problem with Wiimote

The accelerometer readings of the Wiimote are not independent of gravity. The readings are linear and rotational accelerations of the system confounded by the earth's gravitational force. In order to find out the acceleration due to the movement of the Wiimote the static acceleration due to gravity must be subtracted from the accelerometers. The gravitational force is prevalent only on the z-accelerometer reading as long as the Wiimote is horizontal. However when the Wiimote is turned a component of gravity acts on each of the three accelerometers. In order to nullify the effect of gravity on each of the accelerometer readings an accurate estimate of its inclination with the vertical is required. Thus linear accelerations due to movement of the Wii can be measured accurately if the Wiimote was kept perfectly horizontal during displacement. Integration of this linear acceleration gives linear velocity and integrating linear velocity gives linear displacement. It is not possible practically to keep the Wiimote perfectly horizontal. The Wiimote does not provide any provision for estimating its tilt.

CHAPTER 5

IMU 6 DOF V3

5.1 Introduction

IMU 6 DOF is a 6 degree of freedom wireless device from Sparkfun Electronics [17]. It has a RS 232 Bluetooth dongle that's preconfigured to connect the device to the computer when both are powered up. It consists of a three axis accelerometer, MMA 7260Q from Freescale which can be programmed for 1.5g, 2g, 4g or 6g sensitivity. It also has two Invensense IDG 300, 500 degree/ second gyroscopes and Honeywell HMC 1043, a dual-axis magnetic sensor.

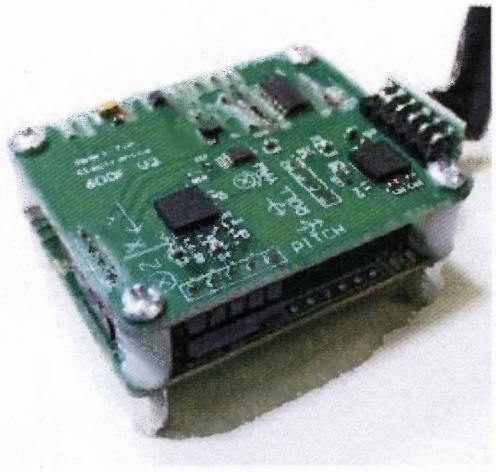

Figure 5.1 Sparkfun's IMU 6DOF V3.
(http://www.sparkfun.com/commerce/product_info.php?products_id=754)

The analog output of the sensors is converted to digital form by using 10 bit ADC. The readings can be accessed in either ASCII or binary format. In binary mode if all the channels are selected to be active the data is a packet of 20 bytes. Each channel is represented by 2 bytes of data; in addition we have count which gives the sample number

and two bytes each for start ("A") and stop ("Z") bytes. The output data is in the following order:

Table 5.1 Distribution of 20 bit packet from IMU.

1	2	3	4	5
Start Bit 'A'	LSB – Count	MSB – Count	LSB – MagX	MSB – MagX
6	7	8	9	10
LSB – MagY	MSB – MagY	LSB – AccX	MSB – AccX	LSB - AccY
11	12	13	14	15
MSB – AccY	LSB – AccZ	MSB – AccZ	LSB – Pitch	MSB - Pitch
16	17	18	19	20
LSB – Roll	MSB – Roll	LSB – Yaw	MSB – Yaw	Stop Bit 'Z'

5.2 Freescale MMA 7260Q 3 Axis Accelerometer

The MMA7260Q is a low cost capacitive micro-machined accelerometer having an adjustable g range from 1.5g to 6g. It has an inbuilt 1 pole low pass filter and internal temperature compensation. It produces an output voltage of 800 mV/g when set at 1.5g sensitivity. It has a low power operating voltage of 2.2 – 3.6 V [19].

5.2.1 Principle of Operation

The device consists of two surface micro-machined capacitive sensing cells. The g-cell is a mechanical structure formed from semi-conductor material. It can be modeled as a set of beams attached to a movable central mass that moves between the fixed beams. The movable beams deflect from their rest position when subjected to acceleration. As the beams attached to the central mass move towards one side its distance from the fixed beam on one side increases and that from the other fixed side decreases. The change in distance is proportional to the acceleration of the system.

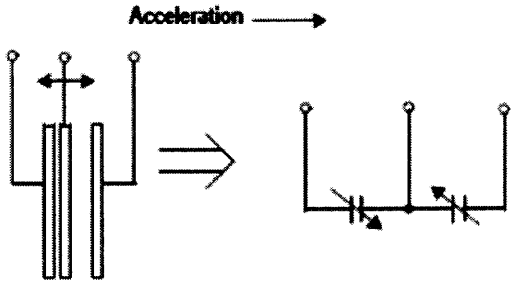

Figure 5.2 Simplified model of the MEMS accelerometer.
(MMA7260Q datasheet)

The g-cell beams form two back-to-back capacitors. As the center beam moves with acceleration, the distance between the beams change and each capacitor's value will change given by the formula

$$C = A * \varepsilon / D \qquad (6)$$

where A is the area of the beam, ε is the dielectric constant, and D is the distance between the beams.

5.3 Invensense Dual-Axis Gyroscope IDG-300

The IDG-300 gyro uses two sensor elements with novel vibrating dual-mass bulk silicon configurations that sense the rate of rotation about the X- and Y-axis (in-plane sensing). This results in a unique, integrated dual-axis gyro with guaranteed-by-design vibration rejection and high cross-axis isolation.

5.3.1 Principle of Operation

A rate gyroscope is a device used to measure angular motion. It is a MEMS based device which provides an output voltage proportional to the rate of turn applied to its sensitive axis. It is based on the principle of detection of coriolis forces. The Coriolis Effect is the apparent deflection of moving objects from a straight path when they are viewed from a rotating frame of reference. To use the Coriolis Effect for detecting angular rotation, a solid structure is forced to vibrate normally at its resonant frequency. This is achieved by applying an alternating voltage to the primary electrodes. The vibration provides the structure with a linear velocity component. When the structure is rotated, the coriolis forces cause the vibration motion of the structure to be coupled to another vibration mode or plane of the structure. The magnitude of this secondary vibration is proportional to the angular rate of turn.

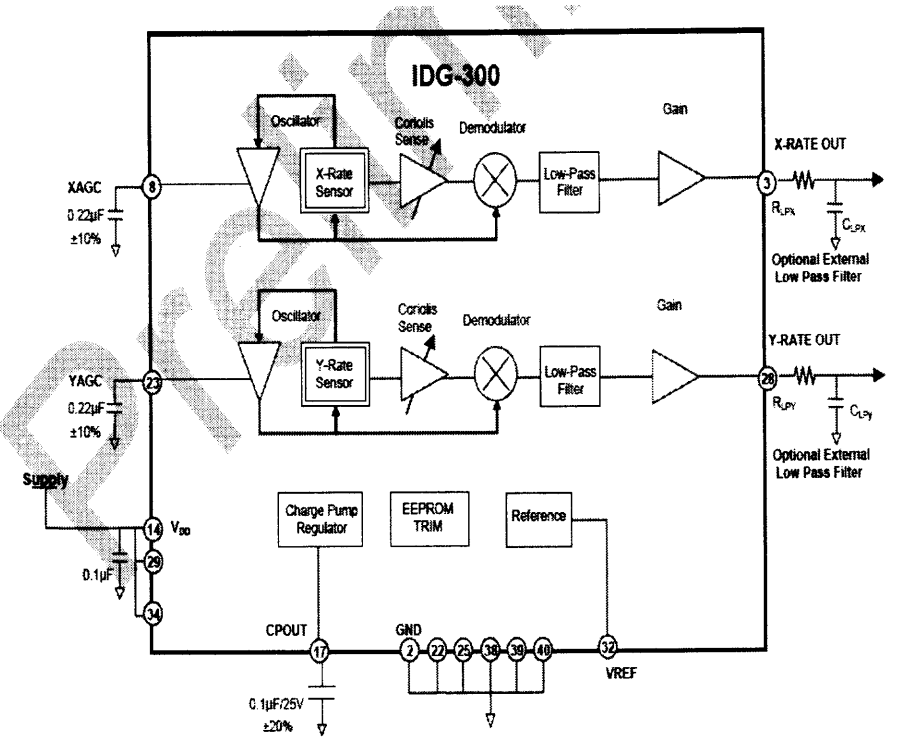

Figure 5.3 Simplified block diagram of the working of IDG-300 Gyroscope.
(IDG 300 datasheet [20])

The oscillator is used to keep the primary loop frequency at resonance and at constant amplitude. When the sensor is rotated about the X- or Yaxis, the Coriolis Effect causes a vibration that can be detected by a capacitive pickup. The resulting signal is amplified, demodulated, and filtered to produce an analog voltage that is proportional to the angular rate.

5.4 Honeywell HMC 1043 dual-axis magnetic sensor

Honeywell's magnetoresistive sensor contains two nearly-perfect orthogonally placed sensors. The sensors are extremely sensitive, low field, solid-state magnetic sensors designed to measure earth's magnetic field from 120 micro-gauss to 6 gauss [21].

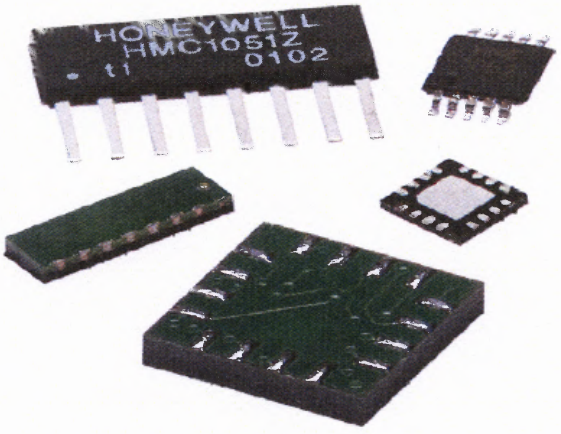

Figure 5.4 Honeywell HMC 104X series in different packaging available.
(Honeywell HMC 1043 datasheet)

5.4.1 Principle of Operation

HMC1043 consists of magneto-resistive sensor placed in a Wheatstone bridge configuration. This magneto-resistive sensor produces a change in resistance which is equivalent to the change in the magnetic field in that direction. Since, HMC 1043 is a dual axis magnetometer it has two magneto-resistive sensors which are placed in Wheatstone bridge configurations. The sensors convert the incident magnetic field in the sensitive axis direction into a differential voltage output.

Thus IMU has six independent degrees of freedom, three angular velocities detected by the gyroscopes and three accelerations calculated using the three axis accelerometer. The accelerations measured by the accelerometer are confounded by the acceleration due to gravity and linear and rotational accelerations. Integrating the angular velocity measured by the gyroscope gives angular displacement. This angular displacement gives the orientation of the IMU in space. This orientation of the IMU can be used to eliminate the effect of acceleration due to gravity from the reading of the accelerometer. Now if the IMU is subjected to only linear translation, the acceleration

measured by the accelerometer is only due to linear acceleration as the rotational acceleration is avoided and the gravitational acceleration is compensated using the orientation information of the device. Thus theoretically, it is possible to calculate the linear translation of the IMU by double integrating this linear acceleration.

CHAPTER 6

PROBLEMS WITH THE GYROSCOPE OF IMU

Gyroscope gives angular velocity which is integrated to calculate the angular displacement. However gyroscope has some inherent problems which lead to error in the angular velocity over time.

6.1 Toggle

6.1.1 The Problem

The value of the gyroscope toggles between two values, even for the rest state the gyroscope value toggles leading to an error of one point. This error is significant as it indicates the rotation of the device at approximately 1°/sec. The error if uncorrected can lead to significant error in the angle prediction over time.

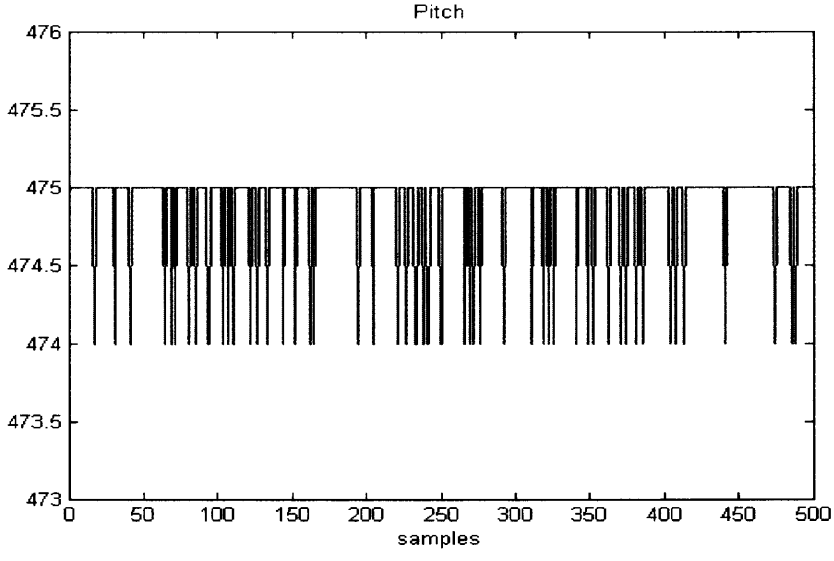

Figure 6.1 Values obtained from the pitch gyroscope over 500 samples with a sampling frequency of 100Hz showing the toggling effect.

6.1.2 Rectifying the problem

Average filtering is done to minimize the effect of toggle. It enables implementation of the filter in real time. A running average of the last three samples is performed and the result is rounded off. This results in elimination of spikes that occur in the gyroscope readings which is necessary to reduce the noise in the system.

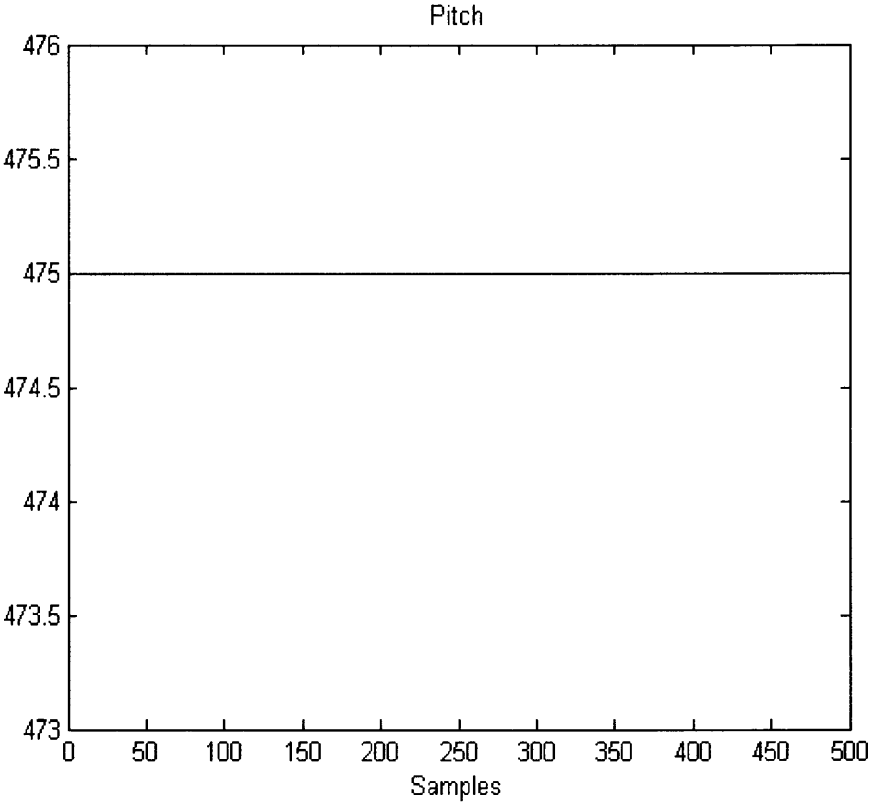

Figure 6.2 Filtered Pitch gyroscope output.

6.2 Bias Drift

Drift is defined as a small change in the bias value of the gyroscope. The gyroscope bias is the output when the angular velocity is zero. This bias value is significant as it is subtracted from all other readings of gyroscope to calculate the angular velocity. All gyroscopes experience drift due to factors such as temperature and stress. The problem of drift is very severe because the angular velocity is integrated to calculate the angular displacement. An error of 1°/sec will lead to an error of 60° per minute for the angular displacement. Hence, the gyroscope reading would indicate an angular displacement of 60° when the device was at rest at the end of one minute. The figure below shows the values obtained from the roll gyroscope kept at rest position over a period of 100 sec.

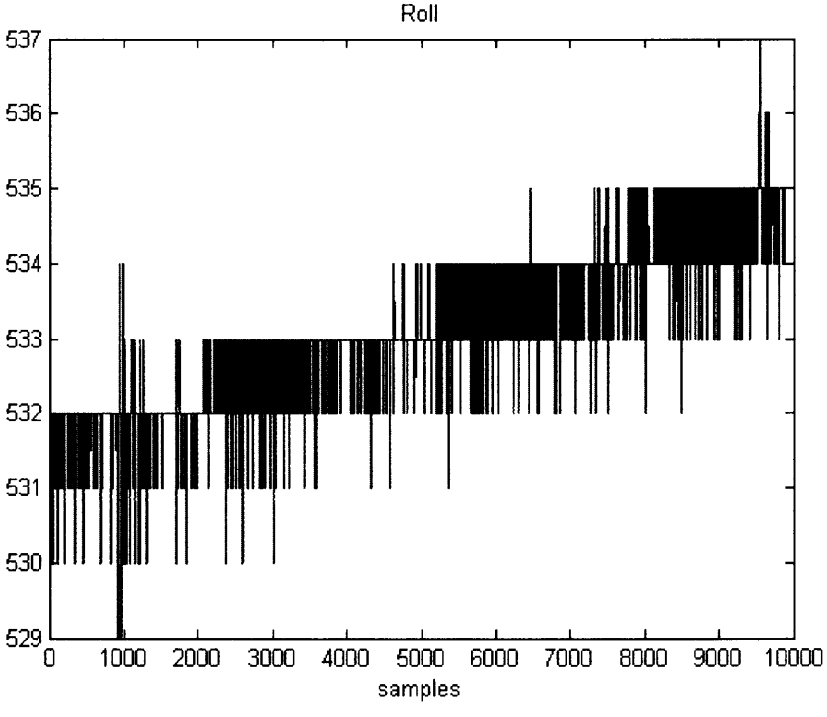

Figure 6.3 Values of the Roll gyroscope drifting over a period of 100 seconds with sampling frequency of 100 Hz.

6.3 Rectifying the problem

A number of techniques such as Kalman filtering, high pass filtering and automatic nulling algorithms are used for dealing with this problem. Kalman filtering is most widely used. It is a tool that estimates the states of a linear system and tries to minimize the variance of the estimation error. The filter changes the variables of the linear model using a second input which provides a good estimation of the output. It modifies the variables such that the output of the linear model is closer to the second input. In the case of the gyroscope, the accelerometers readings are taken as the second input. For Kalman filter an assumption is made that the average value of the noise in the system is zero. The values of the kalman filter are dependent on the sensor characteristics and need to be calculated practically. It is a highly complicated technique and is difficult to estimate the correct parameters of the filter.

A new technique of resetting the bias value of the gyroscope using detection of rest period was designed. A rest period is defined as moments during which the IMU is not moving. Since there is no movement the angular velocity has to be zero. This reading can therefore be used to incorporate the drift factor in the gyroscope reading by updating the gyroscope's bias value. The IMU is intended to be used for playing games; there will be moments while playing the game when the IMU is not moving. These moments are used to recalibrate the IMU.

The IMU is considered to be at rest when the following conditions are satisfied:

1. Change in X,Y,Z accelerometer $< \pm 2$
2. Change in X,Y,Z gyroscope $< \pm 2$
3. Difference in X,Y,Z gyroscope values with calibration value $< \pm 2$

6.4 Implementation of program to calculate angular displacement in MATLAB

THE IMU connects with the computer through a Bluetooth serial interface. A baud rate of 115200 with no parity bits and hardware flow control is selected. The data transfer rate is adjustable between 50 HZ and 300 Hz. Also the sensitivity of the accelerometers can be adjusted between 1.5g and 6g.

Table 6.1 Various parameters of IMU that can be set by the User.

%	&	‘	(	$	(space)
1.5g sensitivity	2g sensitivity	4g sensitivity	6g sensitivity	Start unit	Stop unit

)	*	+	,	-	.
50 Hz	100 Hz	150 Hz	200 Hz	250 Hz	300 Hz

The IMU is configured for 1.5g sensitivity at 100 Hz frequency. Since the LSB and MSB bit of each channel are sent separately unpacking of data is performed. Average filtering is performed on the input data to eliminate jitters. Run time averaging is performed by averaging the last three samples. Rest period is determined as a period when the values of the accelerometers and gyroscope change by not more than one unit. The values of the gyroscopes obtained when this condition is satisfied are used towards finding the new bias value. The bias value of the gyroscope is updated only when the rest period condition is satisfied seven times with the period between each of these readings being not more than 50 time samples. An average of these seven samples is used as the new base value. This prevents accidental update of the base value. The idea behind discarding gyroscope values which are more than 50 samples apart is to update the bias value of gyroscope using values that characterize the current state of the gyroscope.

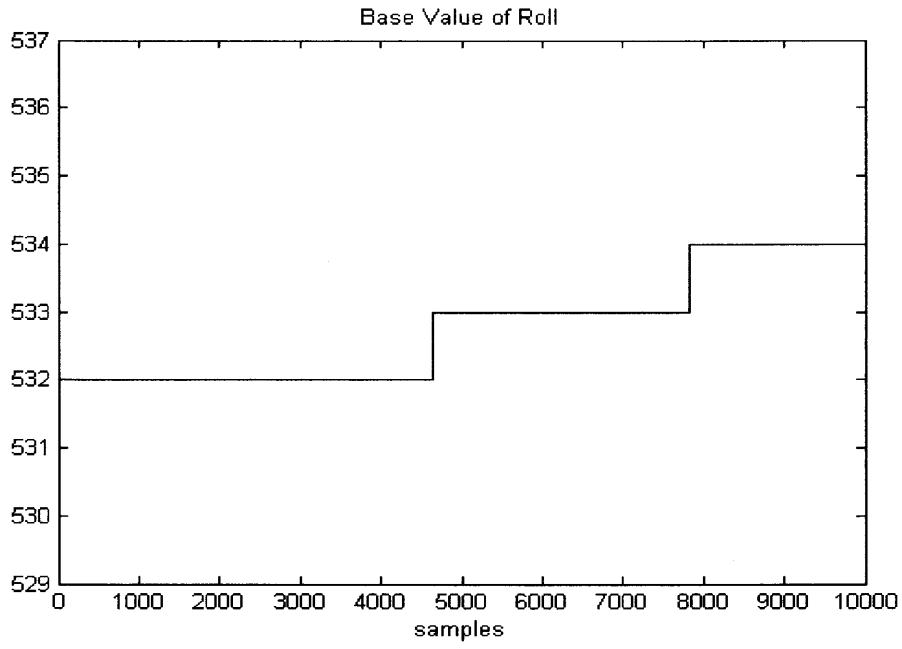

Figure 6.4 Recalibration of the base value of the roll gyroscope when it was allowed to rest over a period of 100 seconds.

6.4.1 Need for a scaling factor

The angular velocity obtained from the gyroscope is integrated to calculate the angular displacement. In order to validate the authenticity of the angular displacement data, it was compared with the angles obtained using the FOB. The IMU unit and the transmitter of the FOB were placed parallel on a non-metallic beam. Both the units were placed in a fashion that their axis of rotation was in the same line. The devices were subjected to same angular rotation by rotating the beam manually about its axis and bringing it back to its rest state. A MATLAB program was used to collect data from both the units simultaneously and the integrated angular velocity data from the gyroscope was plotted against the true angular estimation of the FOB on the same graph.

Figure 6.5 Experimental setup showing the IMU and FOB on a beam.

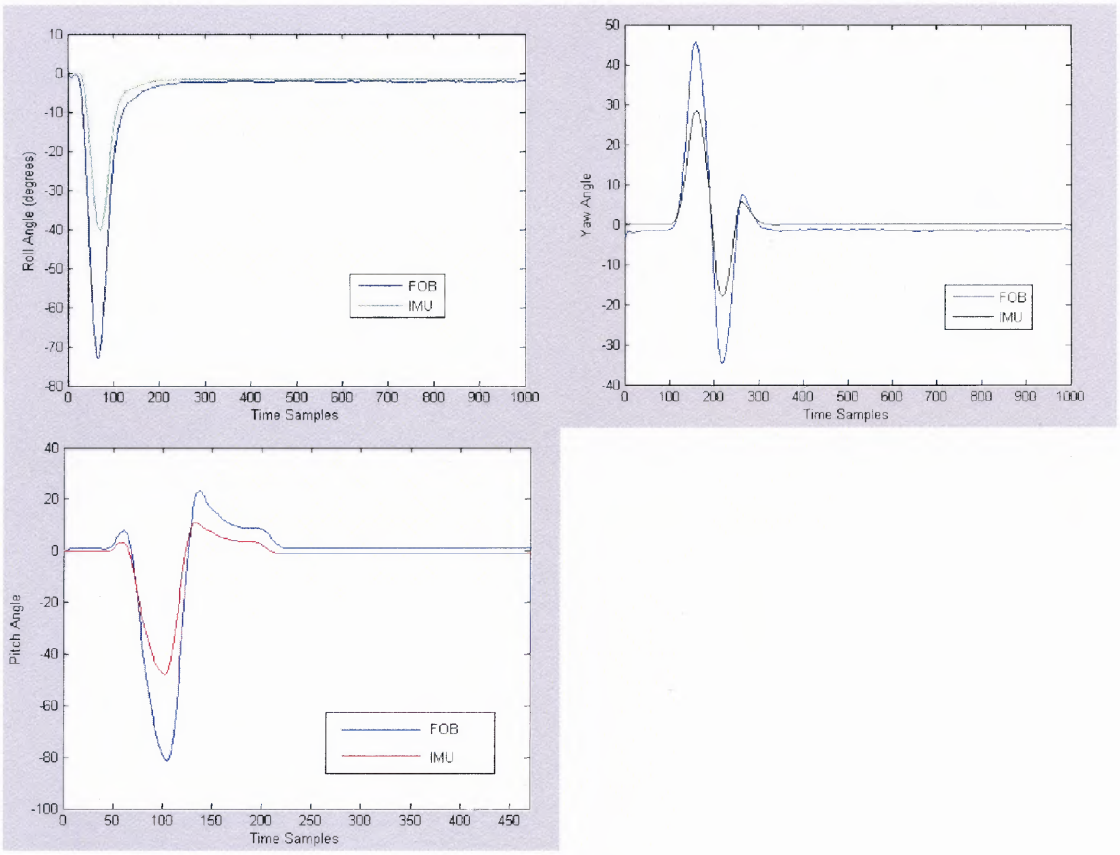

Figure 6.6 Pitch, Roll and Yaw angle of the IMU and FOB.

The three graphs were obtained independently by rotating the beam along the sensitive axis. From the graphs it can be seen that FOB and IMU have similar response to angular rotations. However IMU produces a smaller angular displacement for the same angular

displacement as that of FOB. Hence, the output of the IMU has to be scaled to produce correct angular displacement.

6.4.2 Mathematical proof for need of scaling factor

The output of IDG 300 is an analog voltage which is equivalent to angular velocity. It gives an output of 1.5 V for zero angular velocity. This output changes by 2mV for every degree/second change in angular velocity. Hence an angular velocity of +10 °/sec will increase the output voltage by 20 mV to 1.7 V. This analog output is given to a 10 bit Analog to Digital converter which gives values from 0 to 1023 for a change in the analog output from 0 to 3.3 V.

$$3300 \text{ mV} \equiv 1024 \tag{7}$$

$$\text{Hence,} \quad 1 \text{ mV} \equiv 1024/3300 \tag{8}$$

$$1 \text{ mV} \equiv 0.31 \tag{9}$$

$$2 \text{ mV} \equiv 0.62 \tag{10}$$

$$\text{But from [20],} \quad 2 \text{ mV} \equiv 1 \text{ °/sec} \tag{11}$$

$$\text{Hence,} \quad 0.62 \equiv 1 \text{ °/sec} \tag{12}$$

$$1 \equiv 1.613 \text{ °/sec} \tag{13}$$

Thus, the output of ADC is not directly the angular velocity. It needs to be scaled by a factor of 1.613 to get the true angular velocity.

6.5 Turn table test

The IMU was tested on a turn-table to test its efficiency in accurately estimating angular velocity. A turn-table is device which rotates with a constant angular velocity. Hence it can be used to test the angular velocity measurement of the gyroscope. KENWOOD turntable was used for the experiment. It provides two rotation speeds of 33 rpm and 45 rpm. The IMU was placed on the turn table such that IDG 300 was aligned with the center of the rotating disc and subjected to a rotation speed of 33 rpm.

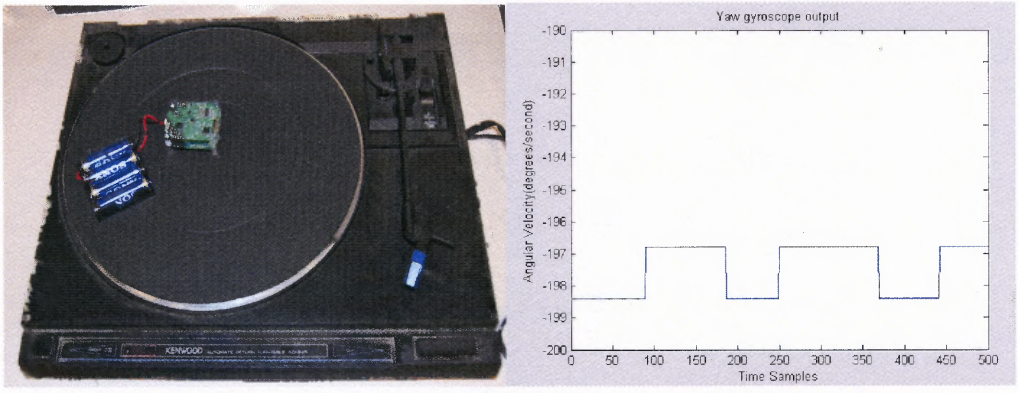

Figure 6.7 a) Experimental Setup of testing the IMU on the Turn-table.
b) Angular velocity output of the gyroscope.

Figure b shows the output of the yaw gyroscope when subjected to a constant speed of 33 rpm on the turn-table. The output of the gyroscope is nearly constant within a tolerance band of one.

Also, 33 rpm implies the turn table completes 0.55 of one revolution in one second. Therefore, angular displacement in 1 second is given by

$$\text{Angular displacement} = 360 * 0.55 \quad (14)$$

$$= 198 \quad (15)$$

Thus, the IMU is subjected to an angular velocity of 198 °/sec.

From figure b, it is evident that the angular velocity measured by the IMU is between 196.8 and 198.4. Thus the IMU measures the angular velocity with considerable accuracy. The output of the IMU is digitized using a 10 bit ADC. This provides 1023 discrete steps to represent the output of the gyroscope which changes from 0 to 3000 mV. This introduces a quantization error in measurement. Though the error in angular velocity measurement is very small this leads to a cumulative error in the measurement of angular displacement. This error becomes significant over time and cannot be ignored. To illustrate the gravity of the problem, The IMU was subjected to a single rotation about its axis and returned to its original position. Ideally, the system should have measured the angular displacement and returned to zero degree. However it can be seen that the reading settles down at 0.3145 degrees. As a result the next angle measurement will have an error of 0.3145 degree. The graph 6.8 b shows that there is a cumulative error of 4 degrees when the IMU was rotated four times and returned back to rest position.

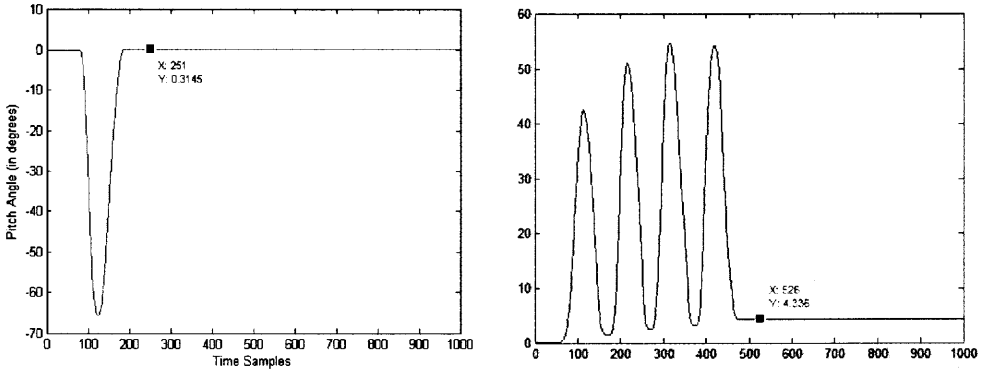

Figure 6.8 a) Accumulation of error after rotation.
b) Cumulative error in angle estimation.

To deal with this problem the gyroscope angle should be recalibrated periodically using the estimation of tilt of the device using accelerometer reading when the device is not rotating. The reading of the accelerometer however is quite noisy and hence accurate estimation of the orientation of the IMU from them is not possible. In order to partially

tackle this problem, an auto reset mechanism was incorporated into the algorithm. The algorithm detects when the IMU unit is horizontal. If the IMU is stationary and horizontal angular displacement in all three axes should be zero. Thus, the horizontal position is used to reset the angular displacement measurement and recalibrate the gyroscope. The condition of horizontal position is true if the accelerometer in z direction measures gravitational force and the accelerations in x and y directions are zero. The limitation of this technique is that the condition may not occur. In the case of video games, since there is visual feedback to calibrate the IMU needs to brought to rest in horizontal position for a fraction of a second if there is large discrepancy between intended and actual orientation.

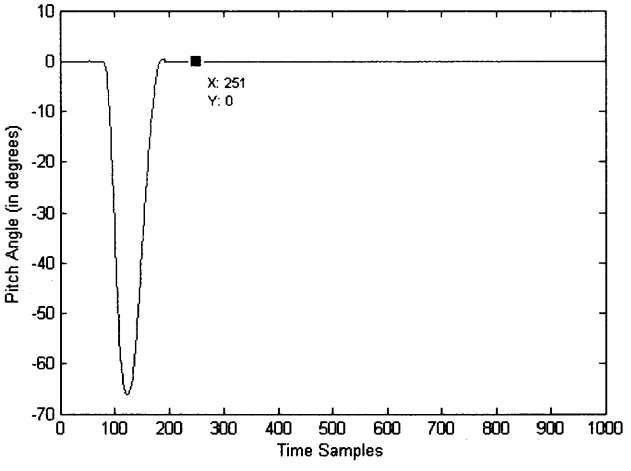

Figure 6.9 Output of Gyroscope with auto reset mechanism.

The figure shows a perfect zero degree output as compared to an output of 0.3145 degree shown in the previous figure.

CHAPTER 7

OBSERVATIONS AND CONCLUSIONS

The new toolbox with VRML graphics provides an engaging environment by creating worlds which replicate the real world closely. It provides an ability to maneuver the objects of the world in three dimensions. The use of SIMROBOT for collision detection however limits the ability of the game to detect collision in two dimensions. This toolbox creates a test bed to test the effectiveness of these games with respect to similar games created in HANDS UP. The Flock of Birds gives a good estimation of the three rotations and translations. It is very expensive and cannot be used for gaming systems for home applications. Also the freedom of movement of the subjects is limited by the length of the cable. External magnetic field also affects the readings of the FOB.

The Wiimote is attractive due to small compact structure, light weight and wireless nature. The effect of the gravitational force on the accelerometer readings limits its use for calculating linear displacements. The IMU 6 DOF with a three axis accelerometer and two gyroscopes has the potential of giving six independent degrees of freedom. The gyroscope values have a tendency to drift with time but this problem of drift can be compensated by using the rest period detection test.Due to the quantization error in the ADC the IMU accumulates error in angle estimation over time. This leads to an error in gravity compensation and correspondingly linear acceleration and linear position. A better resolution ADC can help in reducing the quantization error and enable accurate estimation of six degrees of freedom.

APPENDIX A

MODIFICATIONS MADE IN THE SIMROBOT TOOLBOX

The SIMROBOT toolbox contains three important m files, viewcb, update and run. The viewcb file initializes the GUI and is used to load the simrobot environment. It also takes in the input for the scale factor which is sent to the fobgame5 file to determine the sensitivity of the game to the user input. The FOB along with the VRML world and the world preferences are initialized in the start case. A special case called 'CASE 500' is called when the game is over. The need for the case arises from the fact that delete function in SIMROBOT does not actually delete the robot but just makes it invisible; this creates a problem as deleting an object in VRML actually deletes it. The serial port is closed in case the game in abruptly stopped using the stop button in GUI.

VIEWCB FILE

```
case 500 (line 29)
fwrite(C,66); % FOB stream stop
fclose(C);   %FOB
delete(C);
clear C;
vrclose;
set(findobj('Tag','stop'),'Enable','off')
set(findobj('Tag','start'),'Enable','on')

case 'simulation'  (line 222)
%initializing the fob
% initialize any HID device here....KUNAL

C = serial('COM1');
set(C,'BaudRate',115200);
set(C,'InputBufferSize',100000);
fopen(C);
pause(1)
```

```
Q=[80 7 0 100];
fwrite(C,Q);
%FILTER OFF
T=[80 4 7 0]; %80 is change value, 4 is filter, 7 if off all, 0 is msb
 fwrite(C,T);

% VERIFY MEASUREMENT/SAMPLE RATE
% 79 examines value, 7 is new measurement rate
L=[79 7];
fwrite(C,L);
pause(.1);
d=C.BytesAvailable;
NewMeasurementRate = fread(C,d);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% START DATA COLLECTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fwrite(C, 86); %  86 is position only



fwrite(C,64); % Start FOB strea
% open the VRML world and set the display
% parameters...KUNAL
world=vrworld('onlysphere1.wrl');
vrsetpref('DefaultFigureToolBar','off');
vrsetpref('DefaultFigureStatusBar','on');
vrsetpref('DefaultFigureNavPanel','none');
vrsetpref('DefaultFigurePosition',[5 50 2500 5050]);
open(world);
view(world);
% Take user input for determining the sensitivity of
% the input...kunal
x_scale_fact=inputdlg('Enter X scale');
y_scale_fact=inputdlg('Enter Y scale');
% converting the cell to double...Kunal
x_scale_fact=char(x_scale_fact);
x_scale_fact=str2num(x_scale_fact);
y_scale_fact=char(y_scale_fact);
y_scale_fact=str2num(y_scale_fact);
% initial position of FOB all calculations of the....
%fob position are done relative to this position
[init_x init_y init_z] = fob_getdata(C);

data1=[init_x init_y init_z];  %  kunal
```

```
data1(4)=x_scale_fact;
data1(5)=y_scale_fact;
.
```

UPDATE FILE

The run file calls each and every robot sequentially. The update file is used to update all the robots in the SIMROBOT environment. Modifications are made to use setvel function to set the position in the environment because setpos function gives unexpected results. Since we get real position and not relative position, we directly update the position. Also, in order to prevent the robot from going outside the environment limits thresholding is performed

```
function [new,newmatrix,newrobots] = update(simrobot,matrix,robots)
% UPDATE    (system) updates the simulation.

if (simrobot.power)& (~simrobot.crashed)

% TAKEN OUT FOR SPEED
% & (~simrobot.crashed)
% while (simrobot.power)

  %simrobot.velocity(1) = simrobot.velocity(1) ;
  %simrobot.velocity(2) = simrobot.velocity(2) ;

%    if simrobot.number==2 || simrobot.number==3

 xs=simrobot.velocity(1);
 ys=simrobot.velocity(2);

 rotspd=simrobot.heading;

%vvv using this case allows for spinning
%    else
%
%
%
%    [xs ,ys ,rotspd] = mmodel(simrobot.velocity(1), simrobot.velocity(2),
simrobot.heading);
%
%    end


 xd = get(simrobot.patch,'XData');
yd = get(simrobot.patch,'YData');
```

```
 %TAKEN OUT FOR SPEED
%       % ***** Remove robot from matrix *****
drmex(round([xd';yd']),matrix,robots,simrobot.number);

myrotate(simrobot.patch,[0 0 1],rotspd,[simrobot.position(1) simrobot.position(2) 0])

 [x_size z_size]=size(matrix);



  % Old position
xpos = simrobot.position(1);

ypos = simrobot.position(2);
% ***** Move robot and store data  *****
%simrobot.position(1) = simrobot.position(1) + xs;  % Update x
%simrobot.position(2) = simrobot.position(2) + ys;  % Update y
simrobot.position(1) = simrobot.velocity(1);
simrobot.position(2) = simrobot.velocity(2);

 if simrobot.position(1)> x_size-10
    simrobot.position(1)=x_size-10;
 elseif simrobot.position(1) <10
    simrobot.position(1) = 10;
 elseif simrobot.position(2)>z_size-10
    simrobot.position(2)=z_size-10;
 elseif simrobot.position(2)<10
    simrobot.position(2)=10;
 end
%  TAKEN OUT FOR SPEED
simrobot.heading = simrobot.heading + rotspd;                % Update heading
if simrobot.heading >= 360
  simrobot.heading = simrobot.heading - 360;
end
if simrobot.heading < 0
  simrobot.heading = 360 + simrobot.heading;
end

xd = get(simrobot.patch,'XData');
yd = get(simrobot.patch,'YData');


% Move patch to new location
xd = xd - xpos;

yd = yd - ypos;
```

```
 %xd=xpos;
 %yd=ypos;
xd = simrobot.position(1)+xd;
yd = simrobot.position(2)+yd;

set(simrobot.patch,'XData',xd,'YData',yd)

% ****** Place robot to matrix ******
 prmex(round([xd';yd']),matrix,robots,simrobot.number);

end

% This part is always done
% We need to store robot's position
simrobot.history = [simrobot.history ; simrobot.position simrobot.heading];
% clear xd yd
newrobots = robots;
new = simrobot;
newmatrix = matrix;
```

FUNCTION VRML_COORD

This function converts SIMROBOT coordinates to VRML coordinates.

```
function [x_vr z_vr]= vrml_coord(x_coord,z_coord,x_size,z_size)
% this function takes simrobot coordinates as input and gives vrml
% coordinates as output
%function [x y]= vrml_coord(x_change y_change x_size z_size)

% x_ratio=x_size/12;  % sim to vrml conv.
% z_ratio=z_size/6;
% x_vr= +(x_change/x_ratio); % vrml cord
% z_vr=(z_size/2)+(z_change/z_ratio);
x_vr=(x_coord*200/(x_size))-100;
z_vr=(z_coord*90/(z_size))-45;

if x_vr >100
   x_vr=100;
elseif x_vr <-100
   x_vr =-100;
elseif z_vr >45
   z_vr =45;
elseif z_vr<-45
   z_vr=-45;
end
```

FOBGAME5 ALGORITHM

This algorithm is used to control the hand in VRML using the FOB

```
function new= fobgame5(simrobot,matrix,step,C,data1,world)
% This algorithm was used to move the hand using the FOB
% created by KUNAL
x_scale=data1(4);% input from the dialog box
z_scale=data1(5);
[x_size z_size]=size(matrix);
%x_pixel=300;
%z_pixel=220;
% the ratio is used to calculate the correspondance between FOB coordinates
% and SIMROBOT environment

x_ratio = (x_size)/x_scale;
z_ratio =(z_size)/z_scale;
% putting the HAND in the center position at initialization
if step<=1
   x_pos=150;
   z_pos=110;
else
   %[x_old z_old]=getpos(simrobot);
   [X2 Y2 Z2]=fob_getdata(C);

   x_pos=(-X2+data1(1))*x_ratio+150;
   z_pos=(Z2-data1(3))*z_ratio+110;
end

old_pos=getpos(simrobot);

if abs(x_pos-old_pos(1))<=3
   x_pos=old_pos(1);
end

if abs(z_pos-old_pos(2))<=3
   z_pos=old_pos(2);
end
position=getpos(simrobot);
loc=findobj('Tag','BlueCrashes');
set(loc,'Userdata',position);

% x_change = x_pos-(x_size/2);
% z_change = z_pos-(z_size/2);
[x_vr z_vr]=vrml_coord(x_pos,z_pos,x_size,z_size);
```

```
y_vr=0;
%vrmlacess1(x_vr,z_vr,world);
hand=vrnode(world,'hand');
hand.translation =[x_vr z_vr y_vr]; % Since its 2D and we using x and y of
%vrml for x and z of fob resp.
vrdrawnow;
pause(0.01);

simrobot=setvel(simrobot,[x_pos z_pos]);
simrobot=setaccel(simrobot,[0 0]);
simrobot=clearcf(simrobot);
new=simrobot;
```

MOVCIRC1 ALGORITHM

```
function new =movcirc1(simrobot,matrix,step,C,data1,world)

[x_size z_size]=size(matrix);

global applause
position = getpos(simrobot);
% your algorithm starts here
[dist,num] = readusonic(simrobot,'sensor_1',matrix);

pos=getpos(simrobot);
circle = vrnode(world,'sphere');
   if num==3
     % pos=old;

   %simrobot=setvel(simrobot,[pos(1) pos(2)]);
      delete(simrobot);
      viewcb(500);
      % sound(applause)
      f=str2num(get(findobj('Tag', 'BlueCrashes'),'String'));
      set(findobj('Tag', 'BlueCrashes'),'String', num2str(f+1));
   elseif num==2
   loc=findobj('Tag','BlueCrashes');
   newpos=get(loc,'Userdata');
   pos=newpos;
% new=setloc(simrobot,[79 187])
   simrobot=setvel(simrobot,[pos(1) pos(2)]);

   else
   %pos=old;
```

```
   simrobot=setvel(simrobot,[pos(1) pos(2)]);
   end

if isvalid(circle)
   [x_vr z_vr]=vrml_coord(pos(1),pos(2),x_size,z_size);
   circle = vrnode(world,'sphere');
   y_vr=-20;
   circle.translation = [x_vr z_vr y_vr];
   vrdrawnow;
end


pause(0.01);


% end of your algorithm
simrobot=setaccel(simrobot,[0 0]);
simrobot=clearcf(simrobot);
new =simrobot;
```

ALGBASKET ALGORITHM

```
function new = alg_name(simrobot,matrix,step,C,data1,world)

% your algorithm starts here
[x_size z_size]=size(matrix);
old =getpos(simrobot);
simrobot=setvel(simrobot,[old(1) old(2)]);
dest=vrnode(world,'sphere_dest');
% x_diff=(x_size/2)-old(1);
% z_diff=(z_size/2)-old(2);
[x_vr z_vr]=vrml_coord(old(1),old(2),x_size,z_size);
y_vr=-20;
dest.translation=[x_vr z_vr y_vr];

% end of your algorithm

simrobot=clearcf(simrobot);
new = simrobot;
```

APPENDIX B

PROGRAM TO ACCESS WII

Wii_data_acq File

```
clc
clear all
for i=1:100
   [a(i) b(i) c(i)] = main();
   a=a*9.81
   b=b*9.81;
   c=c*9.81;
   plot(i,a(i),'--rs','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','g',...
            'MarkerSize',10)
   ylim([-9.8 9.8]);
   xlim([0 100]);
   hold on
   pause(0.01)
end
```

MAIN FILE

```
//Wiimote 0.2 by Kevin Forbes (http://simulatedcomicproduct.com)
//This code is public domain, and comes with no warranty. The user takes full
responsibility for anything that happens as a result from using this code.

#include "wiimote.h"
#include <stdio.h>
#include "mex.h"
//Features:
// Read Accelerometer, button values from the wiimote
// Read Accelerometer, stick, and button values from the nunchuck
// Preliminary IR support

//Known issues:
// The IR support is spotty at best. It tends to kick out if you plug your 'chuck in and out
too many times
// Reading 'chuck calibration data doesn't seem to work, so the code just uses defaults
// Multiple Wiimote support not yet tested
```

```
// May only work with Bluesoleil stack?

//Instructions:
// See below for how to connect to a device and start the data stream.
// It is up to the user to call heartbeat fast enough - if you're too slow, you will loose data.
Ideally, this would be done in a separate thread
// There are several public functions for getting the values from the wiimote. Look in
cWiiMote::PrintStatus for examples.

//Version History:
//0.1 Preliminary Release
//0.2 Added nunchuck, IR support


void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])  // added
by kunal
                //int nargs, const char * cargs)
{
        float ax,ay,az;
        cWiiMote wiimote;

        if (wiimote.ConnectToDevice() &&
                wiimote.StartDataStream())
        {
                for (;;)
                {
                        wiimote.HeartBeat();
                        wiimote.PrintStatus();
                        /*ofstream accdata;//("output.txt",ios::app);
                        accdata.open("output.txt");
                        accdata<<wX;
                        accdata<<wY;
                        accdata<<wZ;
                        accdata.close();*/
                        wiimote.GetCalibratedAcceleration(ax,ay,az);
                        plhs[0]=ax;                     // Added by Kunal

                        //Added by Abraham
                        wiimote.WriteBuffer();
                }
        }
        else printf("Connection Failed");

        //return 0;
}
```

APPENDIX C

PROGRAM FOR IMU 6 DOF

```
gyro_calib;
C = serial('COM1');
set(C,'BaudRate',115200);
set(C,'FlowControl','hardware')
set(C,'InputBufferSize',100000);
fopen(C)
data=C.BytesAvailable;
if data>0
   junkf=fread(C,data);
end
%fwrite(C,',');
fwrite(C,'$');
time=0;

pitch_angle=0;
roll_angle=0;
yaw_angle=0;


j=1;
m=1;
samples=80000;
abc=1;
for i=1:samples
   tic
   while data<20
   data=C.BytesAvailable;
   end

   if data>=20
      w=fread(C,20);

      data1(1)=256*w(2)+w(3);
      data1(2)=256*w(4)+w(5);
      data1(3)=256*w(6)+w(7);
      data1(4)=256*w(8)+w(9);
      data1(5)=256*w(10)+w(11);
      data1(6)=256*w(12)+w(13);
      data1(7)=256*w(14)+w(15);
      data1(8)=256*w(16)+w(17);
      data1(9)=256*w(18)+w(19);
```

```
    data=0;
  end

  wireless_data(i,:)=data1;
  time_sample=toc;
  time=time+time_sample;
  wireless_time(i)=time;
  nad(i,:)=wireless_data(i,:);


%   if i==1
%      pitch_base=wireless_data(i,7);
%      roll_base=wireless_data(i,8);
%      yaw_base=wireless_data(i,9);
%   end
  %detection of rest period

  if i>2
    wireless_data(i,:)=round((wireless_data(i,:)+wireless_data(i-1,:)+wireless_data(i-
2,:))/3);
    mad(i,:)=wireless_data(i,:);

    if abs(wireless_data(i,4)-wireless_data(i-1,4))<2 && ...
       abs(wireless_data(i,5)-wireless_data(i-1,5))<2 && ...
       abs(wireless_data(i,6)-wireless_data(i-1,6))<2 && ...
       abs(wireless_data(i,7)-wireless_data(i-1,7))<2 && ...
       abs(wireless_data(i,8)-wireless_data(i-1,8))<2 && ...
       abs(wireless_data(i,9)-wireless_data(i-1,9))<2 && ...
       abs(wireless_data(i,7)-pitch_base)<3 && ...
       abs(wireless_data(i,8)-roll_base)<3 && ...
       abs(wireless_data(i,9)-yaw_base)<3

       wireless_data(i,7)=pitch_base;
       wireless_data(i,8)=roll_base;
       wireless_data(i,9)=yaw_base;

       p_b(j)=wireless_data(i,7);
       r_b(j)=wireless_data(i,8);
       y_b(j)=wireless_data(i,9);



       j=j+1;
       count(j)=i;
       if j>2
         if count(j)-count(j-1)>50
```

```
            j=1;
          end
        end
        k(abc)=i;
        abc=abc+1;
        if j==8
          pitch_base= round((p_b(1)+p_b(2)+p_b(3)+p_b(4)+p_b(5)+p_b(6)+p_b(7))/7);
          roll_base = round((r_b(1)+r_b(2)+r_b(3)+r_b(4)+r_b(5)+r_b(6)+r_b(7))/7);
          yaw_base=round((y_b(1)+y_b(2)+y_b(3)+y_b(4)+y_b(5)+y_b(6)+y_b(7))/7);
          j=1;
          m=m+1;

        end

        if wireless_data(i,4)-accx_base<3 && ...
            wireless_data(i,5)-accy_base<3 && ...
            wireless_data(i,6)-accz_base<3
          pitch_angle=0;
          roll_angle=0;
          yaw_angle=0;
          tp=i;
        end

      end

    end

    pitch_angle=(wireless_data(i,7)-pitch_base)*0.0163+pitch_angle;

    roll_angle=(wireless_data(i,8)-roll_base)*0.0173+roll_angle;
    yaw_angle=(wireless_data(i,9)-yaw_base)*0.0173+yaw_angle;

    pv(i)=wireless_data(i,7)-pitch_base;
    rv(i)=wireless_data(i,8)-roll_base;
    yv(i)=wireless_data(i,9)-yaw_base;


    p1(i)=pitch_base;
    r1(i)=roll_base;
    y1(i)=yaw_base;
    p_a(i)=pitch_angle;
    r_a(i)=roll_angle;
    y_a(i)=yaw_angle;
end
mad(1:2,:)=wireless_data(1:2,:);
t=0:0.01:0.01*(samples-1);
```

```
pitch_cum=cumtrapz(t,pv);
pitch_cum1=pitch_cum*1.63;
%plot(pitch_cum1)
roll_cum=cumtrapz(t,rv);
roll_cum1=roll_cum*1.7;
%figure,plot(roll_cum1);
yaw_cum=cumtrapz(t,yv);
yaw_cum1=yaw_cum*1.7;
%figure,plot(yaw_cum1);
figure,plot(p1);
fwrite(C,' ');
fclose(C);
plot(p_a);
```

CALIBRATION OF GYROSCOPE

```
C = serial('COM1');
set(C,'BaudRate',115200);
set(C,'FlowControl','hardware')
set(C,'InputBufferSize',100000);
fopen(C)
data=C.BytesAvailable;
if data>0
   junkf=fread(C,data);
end
fwrite(C,'$');
time=0;

samples=10;
for i=1:samples
   tic
   while data<20
   data=C.BytesAvailable;
   end

   if data>=20
      w=fread(C,20);

      data1(1)=256*w(2)+w(3);
      data1(2)=256*w(4)+w(5);
      data1(3)=256*w(6)+w(7);
      data1(4)=256*w(8)+w(9);
```

```
      data1(5)=256*w(10)+w(11);
      data1(6)=256*w(12)+w(13);
      data1(7)=256*w(14)+w(15);
      data1(8)=256*w(16)+w(17);
      data1(9)=256*w(18)+w(19);
      data=0;
    end

    wireless_data(i,:)=data1;
    time_sample=toc;
    time=time+time_sample;
    wireless_time(i)=time;
    end
fwrite(C,' ');
fclose(C);

pitch_sum=0;
roll_sum=0;
yaw_sum=0;
accz_sum=0;
accy_sum=0;
accx_sum=0;
for i=1:samples
    pitch_sum=pitch_sum+wireless_data(i,7);
    roll_sum=roll_sum+wireless_data(i,8);
    yaw_sum=yaw_sum+wireless_data(i,9);
    accz_sum=accz_sum+wireless_data(i,6);
    accy_sum=accy_sum+wireless_data(i,5);
    accx_sum=accx_sum+wireless_data(i,4);
end

pitch_base=round(pitch_sum/samples);
roll_base=round(roll_sum/samples);
yaw_base=round(yaw_sum/samples);
accx_base=round(accx_sum/samples);
accy_base=round(accy_sum/samples);
accz_base=round(accz_sum/samples);
```

REFERENCES

1. W. Jenkins, F. Seil and M. Merzenich, "Reorganization of neocortical representations after brain injury: A neurophysiological model of the bases of recovery from stroke," in *Progress in Brain*, New York: Elsevier, 1987.

2. B. Kopp et al., "Plasticity in the motor system related to therapy-induced improvement of movement after stroke," in *Neuroreport*, vol. 10, no. 4, pp. 807–810, Mar. 17, 1999.

3. J.W. Lee and K. Rim, "Maximum finger force prediction using a planar simulation of the middle finger." in *Proc. Instrum. Mech. Eng.*, vol. 204, pp. 160–178 1990.

4. J. Liepert,W. H. Miltner, H. Bauder, M. Sommer, C. Dettmers, E. Taub,and C.Weiller, "Motor cortex plasticity during constraint-induced movement therapy in stroke patients.", in *Neurosci. Lett.*, vol. 250, no. 1, pp. 5–8, 1998.

5. R. J. Nudo, "Neural substrates for the effects of rehabilitative training on motor recovery after ischemic infarction,"in *Science*, vol. 272, pp.1791–1794, 1996.

6. R. Foulds et al. "Sensory-motor enhancement in a virtual therapeutic environment.", in *Virtual Reality* 2007.

7. S. Trojan and J. Pokorny. "Theoretical and clinical significance of neuroplasticity.", in *Bratisl Lek Listy*. 98, 667-673 2007.

8. W. Jenkins, M. Merzenich," Reorganization of neocortical representation after brain injury: a neurophysiological model of the bases of recovery from stroke.", in *Progress in Brain*. Elsevier,New York 1987.

9. R. Nudo, "Neural substrates for the effects of rehabilitative training on motor recovery after ischemic infarction.", in *Science* 272: 1791 – 1794 1996.

10. R. Chen, L. Cohen, M. Hallet, "Nervous system reorganization following injury.", in *Neuroscience* 111(4) : 761-773 2000.

11. D. Jack, "Virtual reality-enhanced stroke rehabilitation.", in *IEEE Trans Neural Syst Rehabil Eng* 9(3):308–318 2001.

12. N. O'Dwyer, L. Ada, "Reflex hyperexcitability and muscle contracture in relation to spastic hypertonia.", in *Curr Opin Neurol*9(6):451–455 1996.

13. D. Reid, Department of Occupational therapy, Toronto in The influence of virtual reality on playfulness in children with cerebral palsy: A pilot study, Occupational Therapy International, 11(3), 131-144, 2004.

14. H. Krebs, N. Hogan, M. Aisen, "Volpe BT Robot-aided neurorehabilitation." in *IEEE Trans Rehabil Eng* 6(1):75–87 1998.

15. J. Deutsch. "Rehabilitation of Musculoskeletal Injuries Using the Rutgers Ankle Haptic Interface: Three Case Reports. Program in Physical Therapy, UMDNJ-SHRP, Newark NJ 2001.

16. Flock of Birds. (October 15, 2006) http://www.ascension-ech.com/products/flockofbirds.php.

17. IMU 6 DOF V3 (March 14, 2007) http://www.sparkfun.com/commerce/product_info.php?products_id=754.

18. Wiimote (July 12, 2007) http://www.wiili.org/index.php/Main_Page.

19. Freescale MMA 7260Q 3 axis accelerometer (August 10, 2007) http://www.freescale.com/files/sensors/doc/fact_sheet/MMA7260QFS.pdf.

20. Invensense IDG 300 gyroscope (October 26, 2007) http://www.invensense.com/shared/pdf/IDG300_Datasheet.pdf.

21. Honeywell HMC 1043 dual axis magnetic sensor (November 10, 2007) http://www.ssec.honeywell.com/magnetic/datasheets/hmc1043.pdf.