Motion sensing interface for the visualization of multidimensional data using inertial measurement

Hussaini Safiullah

2011

Hussaini, S. (2011). Motion sensing interface for the visualization of multidimensional data using inertial measurement.. Master' s thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/43996>

<https://doi.org/10.32657/10356/43996>

NANYANG
TECHNOLOGICAL
UNIVERSITY

**MOTION SENSING INTERFACE FOR THE
VISUALIZATION OF MULTIDIMENSIONAL DATA
USING INERTIAL MEASUREMENT**

SAFIULLAH HUSSAINI
SCHOOL OF COMPUTER ENGINEERING
2011

**MOTION SENSING INTERFACE FOR THE
VISUALIZATION OF MULTIDIMENSIONAL DATA
USING INERTIAL MEASUREMENT**

SAFIULLAH HUSSAINI

School of Computer Engineering

A thesis submitted to the Nanyang Technological
University in fulfilment of the requirements for the degree
of Masters of Engineering

2011

TABLE OF CONTENTS

TABLE OF CONTENTS	III
ABSTRACT	VII
NOMENCLATURE FOR EQUATIONS	IX
LIST OF ACRONYMS	X
TABLE OF FIGURES	XI
LIST OF EQUATIONS	XIV
CHAPTER 1 INTRODUCTION	1
1.1. MOTIVATION	4
1.2. OBJECTIVES	6
1.3. ORGANIZATION OF THIS THESIS	8
CHAPTER 2 LITERATURE REVIEW	10
2.1. POSITIONING AND POSITION TRACKING SYSTEMS IN HUMAN COMPUTER INTERACTION	10
2.2. INERTIAL SYSTEMS	17
CHAPTER 3 INERTIAL SYSTEMS	22
3.1. INTRODUCTION TO INERTIAL SYSTEMS	22
3.2. THEORETICAL PRINCIPLES	25
3.3. PRINCIPLES OF GYRO-FREE IMU	33
3.4. PRINCIPLES OF GYRO-BASED IMU	39
CHAPTER 4 INTERFACE DESIGN	42

4.1.	INTRODUCTION	42
4.2.	SIGNAL DESCRIPTION AND CHARACTERISTICS	43
4.3.	ADAPTIVE COMPLEMENTARY FILTERS FOR SPECIAL ORTHOGONAL GROUP	46
4.4.	LINEAR ACCELERATION/VELOCITY EXTRACTION AND POSITION CALCULATION	49
4.4.1.	<i>VELOCITY BIAS REMOVAL</i>	50
4.4.2.	<i>POSITION CALCULATION</i>	53
4.5.	PROOF OF CONCEPT PROTOTYPE (VERSION 1.0) FOR GF-IMU	54
4.6.	GYRO BASED IMU PROTOTYPE (VERSION 2.0)	56
4.7.	IMPLEMENTATION	57
4.7.1.	<i>INERTIAL INPUT MEASUREMENT AND VISUALIZATION UNIT (IMVU)</i>	57
4.7.2.	<i>COMPUTER-END ALGORITHM AND DISPLAY SOFTWARE</i>	58
4.7.3.	<i>DATA ACQUISITION</i>	59
4.7.4.	<i>SIGNAL EXTRACTION AND CONDITIONING</i>	60
4.7.5.	<i>DISCUSSION ON THE MISSING MOMENT ARM</i>	62
4.8.	MEASUREMENT TESTING	65
	CHAPTER 5 INTERACTION DESIGN	73
5.1.	VISUALIZATION REQUIREMENTS	74
5.2.	DISPLAY AND SENSING CONSTRAINTS	76
5.2.1.	<i>DISPLAY ERGONOMICS</i>	76
5.2.2.	<i>SENSOR CONSTRAINTS</i>	77
5.3.	VISUAL INTERACTION MODES	78
5.3.1.	<i>BROWSE MODE</i>	78
5.3.2.	<i>FOLLOW MODE</i>	81
5.3.3.	<i>JOG MODE</i>	82
	CHAPTER 6 USER STUDY AND DISCUSSION	85
6.1.	MOUSE-ONLY AND KEYBOARD-ONLY INTERFACES	86

6.2.	EXPERIMENT 1 DESIGN AND RESULTS	89
6.2.1.	<i>DESIGN OF EXPERIMENT 1</i>	89
6.2.2.	<i>RESULTS OF EXPERIMENT 1</i>	92
6.3.	EXPERIMENT 2 DESIGN AND RESULTS	100
6.3.1.	<i>DESIGN OF EXPERIMENT 2</i>	100
6.3.2.	<i>RESULTS OF EXPERIMENT 2</i>	102
6.4.	EXPERIMENT 3 DESIGN AND RESULTS	105
6.4.1.	<i>DESIGN OF EXPERIMENT 3</i>	105
6.4.2.	<i>RESULTS OF EXPERIMENT 3</i>	107
	CHAPTER 7 DESIGN CHALLENGES	112
7.1.	CHALLENGES WHILE DESIGNING GF-IMU BASED INERTACTIVE DEVICE	112
7.2.	CHALLENGES WHILE DESIGNING GYRO-IMU BASED INERTACTIVE DEVICE	113
7.3.	DISCUSSION ON THE PERFORMANCE OF GYRO-IMU AND GF-IMU	115
	CHAPTER 8 CONCLUSION AND FUTURE WORK	117
8.1.	CONTRIBUTIONS	117
8.2.	POTENTIAL APPLICATIONS	119
8.3.	FUTURE WORK	121
	REFERENCES	123
	APPENDIX A OLDER PROTOTYPES	131
A.1	VERSION 0.1	131
A.1.1	<i>SENSOR/ACCELEROMETERS</i>	132
A.1.2	<i>MICROCONTROLLER</i>	132
A.1.3	<i>INTERFACE</i>	133
A.1.4	<i>SOFTWARE (REMOTE DEVICE)</i>	133
A.1.5	<i>SOFTWARE (DATA ACQUISITION AND RECORD, VER. 1.0)</i>	133
A.1.6	<i>SOFTWARE (DATA ACQUISITION AND VISUALIZATION VER. 1.1)</i>	134
A.1.7	<i>TEST HARDWARE</i>	135

A.2 VERSION 0.2	135
<i>A.2.1 REMOTE DEVICE</i>	<i>136</i>
<i>A.2.2 COMPUTER-END ALGORITHM AND DISPLAY SOFTWARE</i>	<i>137</i>
APPENDIX B STATE SPACE DESCRIPTION	139
APPENDIX C PUBLICATIONS ARISING FROM M.ENG RESEARCH	143

ABSTRACT

3-dimensional (3D) motion-based interfaces are the next step in the evolution of human input devices for the computer, with applications like interactive 3D visualization and motion/trajectory control for robots providing the motivation for their development. Such interfaces are also being developed by the industry for their potential as game controllers. Even portable devices like cell phones are now fitted with the sensors for motion-based human interaction.

This document describes the design, implementation and evaluation of a portable inertial measurement based motion tracking and visualization device (Inertial Input Measurement and Visualization Unit or IMVU) that acts as a tangible interface translating physical six degrees-of-freedom (DOF) to corresponding motion in a 3D virtual environment. This work investigated two main sensor configurations, namely a gyro-free approach that used four 3-axis accelerometers and a conventional inertial measurement approach that used a gyro-accelerometer configuration. Issues such as placement of inertial sensors, the conditioning and interpretation of their outputs were discussed. Based on the characteristics of the inertial sensing systems, appropriate interaction designs for interactive exploratory visualization were proposed. User study and evaluation were conducted for both designs and the results reveal that the gyro-accelerometer configuration provided superior performance in terms of controllability and ease of use. The study also shows that training on the novel tangible interface produced noticeable improvement in the user's performance

in tasks such as controlling the 3D orientation and position of virtual objects in 3D space.

NOMENCLATURE FOR EQUATIONS

Vector quantities:

- 3D real vector: $\vec{a}$
- n -dimensional column vectors: $\mathbf{a}$

Matrices:

- Matrix $\mathbf{A}$
- 3D vector represented as a Matrix a_x
- Quaternion $\mathbf{q}$
- Vector represented as quaternion $\mathbf{p}(\vec{a})$
- Quaternion represented as a matrix $\mathbf{m}(\mathbf{q})$

Scalar quantities and components:

- Scalar quantity/variable a
- Vector or Matrix components a_x , $\mathbf{A}_{11}$
- Constants A

Operations:

- Derivative $\dot{a}$, $\dot{\vec{a}}$, $\dot{\mathbf{a}}$, $\dot{\mathbf{A}}$
- Quaternion product $\mathbf{q} \cdot \mathbf{p}$
- Quaternion conjugate $\mathbf{q}^*$
- Cross/outer product $\vec{a} \times \vec{c}$
- Matrix product $\mathbf{A} \cdot \mathbf{X}$, $\mathbf{A} \cdot \mathbf{x}$
- Natural exponent $\exp(a)$
- Matrix exponent $\exp(\mathbf{A})$

LIST OF ACRONYMS

3D	Three Dimensions
2D	Two Dimensions
GUI	Graphical User Interface
CLI	Command Line Interface
MRI	Magnetic Resonance Imaging
CAT	Computer Aided Tomography
GPS	Global Positioning System
IMVU	Inertial input Measurement Visualization Unit
IMU	Inertial Measurement Unit
RF	Radio Frequency
ZVC	Zero Velocity Compensation
HCI	Human Computer Interaction
INS	Inertial Navigation Scheme
$SO\{3\}$	Special Orthogonal Group 3
$SE\{3\}$	Special Euclidean group
GF-IMU	Gyro Free Inertial Measurement Unit
IGS	Inertial Guidance System
MEMS	Micro Electro Mechanical Systems
UAV	Unmanned Aerial Vehicle
ADC	Analog to Digital Converter
LCD	Liquid Crystal Display

TABLE OF FIGURES

Figure 1.1 3D volumetric texture visualization of MRI images.	5
Figure 1.2 A simplified model of the IMVU.	7
Figure 2.1 3D Pen.....	12
Figure 2.2 3D visualization unit using props.	13
Figure 2.3 The hand based gesture recognition system.	13
Figure 2.4 The Wii-mote.....	15
Figure 2.5 The iPhone.	15
Figure 2.6 The iPad.	16
Figure 2.7 The Sony Sixaxis PS3 Controller..	16
Figure 3.1 Symmetric and asymmetric configurations.	25
Figure 3.2 Inertial and body frame of reference.	26
Figure 3.3 Inertial and body frame of reference.	27
Figure 3.4 The placement scheme to measure rotation about a single axis	33
Figure 3.5 The placement scheme to measure all rotations for a Gyro Free Inertial Measurement Unit.	38
Figure 3.6 Accelerometer placement in Gyro based IMU.	40
Figure 3.7 The sensor placement scheme in a Gyroscope based IMU..	40
Figure 4.1 The motion applied to the IMVU.	43
Figure 4.2 Frequency response for an accelerometer signal.	45
Figure 4.3 Angular velocity signal (blue) and total acceleration signal.....	46
Figure 4.4 Adapting frequency response for the SO{3} filter.....	48
Figure 4.5 The acceleration signal and velocity signals.	51
Figure 4.6 The signal flow diagram for position and orientation algorithm	54
Figure 4.7 Placement of the four accelerometers.....	55
Figure 4.8 System layout.	55

Figure 4.9 The placement of the gyroscope and the accelerometer in the Gyro-based IMVU...	57
Figure 4.10 System architecture overview.....	59
Figure 4.11 The rotational forces and accelerometer axis	61
Figure 4.12 Rotating gravity, centripetal forces and tangential forces on the IMVU.	63
Figure 4.13 The graph showing measured distances when the GF-IMU was moved against the test distance.....	67
Figure 4.14 The graph showing measured distances when the Gyro-based IMU was moved against the test distance.....	68
Figure 4.15 The typical settling time for the GF IMU.....	69
Figure 4.16 The typical settling time for the Gyro based IMU.....	69
Figure 4.17 The graph showing measured angles when the GF-IMU was moved against the test angle.....	71
Figure 4.18 The graph showing measured angles when the Gyro-based IMU was moved against the test angle.	72
Figure 5.1 IMVU application for viewing MRI volumetric texture.	74
Figure 5.2 Coarse mapping.....	75
Figure 5.3 Precise mapping.	76
Figure 5.4 Rotational limits used in the interaction design.....	77
Figure 5.5 Z-axis rotation example.....	79
Figure 6.1 Mouse interactions for 3D visualization for translational motion.	87
Figure 6.2 Mouse interactions for 3D visualization for rotational motion	87
Figure 6.3 Screen shot of the task 1.....	92
Figure 6.4 Screen shots of task 2 and task 3.	92
Figure 6.5 The graph shows the average time taken (in milliseconds) and its standard deviations for completing task 1 versus the attempt number.....	94
Figure 6.6 Average time taken (in milliseconds) and its standard deviations for performing task 2 versus the attempt number.....	96
Figure 6.7 Average time taken (in milliseconds) and its standard deviations for performing task 3 versus the attempt number.....	97
Figure 6.8 The wireframe view showing experiment 2 tunnel track.	102

Figure 6.9 The mouse backtracking problem.	104
Figure 6.10 A view of the tunnel like path the user has to follow in Experiment 2.	106
Figure 6.11 Some of the obstacles the user has to negotiate while performing in the Experiment 2.	106
Figure 6.12 A screen shot of the Experiment 2 showing the lit tunnel from the users' perspective.	106
Figure 6.13 The graph showing signatures of overlapping normal Gaussian distribution.....	109
Figure 7.1 Total acceleration and gravity vector.	114
Figure 8.1 A screen shot of the demo.	121
Figure 8.2 A screen shot of the demo from a different angle.	121

LIST OF EQUATIONS

Eq 3.1.....	28
Eq 3.2.....	28
Eq 3.3.....	29
Eq 3.4.....	29
Eq 3.5.....	29
Eq 3.6.....	30
Eq 3.7.....	30
Eq 3.8.....	30
Eq 3.9.....	31
Eq 3.10.....	31
Eq 3.11.....	31
Eq 3.12.....	31
Eq 3.13.....	34
Eq 3.14.....	36
Eq 3.15.....	37
Eq 3.16.....	37
Eq 3.17.....	37
Eq 3.18.....	37
Eq 4.1.....	47
Eq 4.2.....	47
Eq 4.3.....	47
Eq 4.4.....	47
Eq 4.5.....	49
Eq 4.6.....	52
Eq 4.7.....	52
Eq 4.8.....	52

Eq 4.9.....	53
Eq 4.10.....	61
Eq 4.11.....	61
Eq 4.12.....	64
Eq 5.1.....	80
Eq 5.2.....	81
Eq 5.3.....	83
Eq 6.1.....	107

CHAPTER 1

INTRODUCTION

The past thirty years have been eventful for the field of computer engineering. Not only have we seen the advent of faster, better computers, but also have discovered new applications for their use. While those applications have increased exponentially, the methods to communicate with computers have not. For a long time, computers have had fairly limited devices to allow users to interact with them, called legacy devices. The mouse and the keyboard have been around for a long time and have also been effective. But now, as the world is finding computers to be more than just a tool, it also wants new ways to communicate with them. Human Computer Interaction (HCI) has therefore become one of the blooming fields of research.

Advancement in computer software and hardware, especially the development of position based pointing devices such as mouse, track ball, track pad and light pens, etc and graphics rendering displays gave birth to Graphical User Interfaces (GUI). The GUI is a major upgrade to its predecessor, the Command Line Interface (CLI) as far as ease of use, required learning curve for the user and multitasking is concerned to communicate with the applications. However, as the applications of the computer grew complex, multiple parameters needed to be manipulated by the user simultaneously, as is the case with many

visualization applications. Both the GUI and its associated pointing devices became cumbersome to use. Tasks such as viewing three-dimensional objects rendered on a display, or navigating through a three-dimensional virtual environment can become very tedious with mouse and keyboard, especially when the users' aim is not just to view the data but also to work or manipulate it. Such example of an application would be sculpturing a 3D model in software such as Autodesk's 3D Studio MAX™ or the industry standard known as AutoCAD.

When dealing with higher dimensional data such as three dimensional objects, volumetric textures for Magnetic Resonance Imaging (MRI) and Computer Aided Tomography (CAT) scans, GUI requires users to interact with graphical control elements such as slide bars and buttons to perform key visualization or manipulation tasks. This is because many popular pointing devices only provide two degrees of control (e.g. conventional mouse and tablet). The manipulation and interaction of higher dimension datasets tends to be cumbersome and comparatively unintuitive due to lack of higher degrees-of-freedom. It requires the user to focus on the usage of the interface rather than on the problem. This only diverts the attention of the user, which would have been better spent performing the overall task as compared to manipulating GUI controls.

Position plays an important role in interfaces that allow users to point to objects or control an object's position or orientation. New devices that provide user control with more degrees of freedom are readily available in the market. These devices employ various methods to calculate or control position and orientation.

Irrespective of the sensors employed, there are two methods to find position and orientation of any object; the first method requires displacement information for a number of global reference points from the object [1, 2, 3]. By computing the angles between these displacement vectors one can deduce the position of an object. This method is called triangulation and it can be very accurate depending on the accuracy of the measured displacements. To find the position of an object in 3D, 3 or 4 reference points are needed to measure displacement depending on the restrictions in the environment. Thus the setup of such reference points is quintessential to triangulation. In a mobile operating environment where a user may move around from one place to another, which may or may not be covered by our reference points, it may not be feasible to setup such points altogether. Furthermore, existing reference points such as GSM base stations, GPS, and Wi-Fi may not provide the resolution necessary for a Human Computer Interface, which maybe a few centimeters.

Thus a different technique is required to capture position based human gestures. In this technique, we measure or calculate the change in position from another measureable quantity, and update this change to the object's previously known position. This method of positioning or position tracking is called dead reckoning [1, 2, 3]. A variation of this method is used in generally all mouse-like devices; however, it is more widely used in aerospace navigation systems employing inertial sensors such as gyroscopes and accelerometers. The advantage of systems based on dead reckoning, especially inertial sensor based systems over triangulation based positioning systems is that they are self-contained, requiring no external influences. This makes such a system very

robust and secure but at a cost of accuracy. The low accuracy is because of the errors in measured change that get accumulated over time as dead reckoning is a recursive method where outputs of the current cycle will become inputs in the next.

Previously, input and output devices for the computer used to be disparate, both functionally and spatially. This was because hardware used to be large and bulky. However, now when portable computing devices are getting smaller and more compact, designers neither have the luxury nor the space to keep the two functions separate. At present, when the world is starting to find the computer more useful than just a mathematical tool and is adopting all its variations, from cell phones to workstations, this disparity between the input and output mechanisms becomes redundant. As humans, we expect immediate response from the same object the stimulus is applied to in our environment. Thus to make computing devices mimic general objects, a seamless, real-time input and output functions need to be performed by a device. We intend to follow this model when designing our device, thus providing the user with an input and output device responsive enough to offer a seamless, engrossing user experience.

1.1. MOTIVATION

Consider a neurosurgeon that has on his computer a detailed 3D volumetric data set of one of his patient with a tumor in his brain. Before taking careful

measurements of the tumor for possible precision surgical removal, he would first need to visualize the size, shape, position and orientation of the tumor. He would also need to form a visual map of how the tumor is physically embroiled with neighboring cerebral tissue and the possible side effects for the patient when removing the tumor. This visual map of the internal structure of the tissue region of interest is currently formed by analyzing a series of MRI images consisting of a closely-spaced sequence of 2D image slices usually taken parallel to the sagittal, coronal or transverse planes of the patient. Given that the neurosurgeon is a seasoned practitioner and has a good visual notion of the internal 3D structure of healthy tissues, an exploratory visualization tool such as the “virtual MRI scanner” shown in Figure 1.1 could help him acquire such a visual map quickly. This is because such a visualization unit allows him to perform interactive 3D exploratory visualization of the tumor region by seamlessly specifying the travel path and orientation of a 2D visual slice plane, which “acquires” and displays real-time 2D MRI images at each corresponding 3D location. Choosing various trajectories that cuts through the tumor tissue will allow him to quickly visualize its size, orientation and shape.

Figure 1.1. The ability to have controlled visualization of real-time 2D visual slices through arbitrary trajectories in the 3D space of the image dataset can allow a seasoned medical practitioner to quickly acquire a visual map of the internal structure of a problematic tissue.

Another interesting application is an input and display device for portable gaming. Imagine a virtual world that can be explored and interacted with using a portable display and input unit much like a Gameboy or PSP. The user's avatar can be controlled more intuitively using tilt and jerks to move about and perform actions like jump, sidestep, pan or rotate the view instead of keys or joysticks. Such an experience would not only be more immersive, but intuitive as well, far from the current devices employed in the industry like mouse, keyboards and gamepads. With the display mounted right on top of the input device, the user would feel like they are peering directly into a world through a movable "window". The current industry is also moving into such a direction, with portable devices like the Apple's iPhone™ and the iPad™, which already have some of the capabilities that we want in our device. However, currently these devices are restricted to three degrees of freedom at most. We are proposing a device that can give the user virtually all six degrees of freedom. This expansion of capabilities is possible because of the use of Inertial Measurement or Dead Reckoning and multiple strategically placed inertial sensors.

1.2. OBJECTIVES

The main objective of this project is to develop a motion input based tangible interface for visualizing multi-dimensional data, such as 3D models, volumetric textures (MRI and CT scan data), etc. However there are other applications, such as visualization of multi-level textures or floor plans (of an Integrated Circuit or buildings), maps or photo libraries, even be used as a Human

Machine Interfaces for control applications etc. Our system level proposal is a computing device with a display and an embedded positional input capability as shown in Figure 1.2.

Figure 1.2. A simplified model of our proposed device. The sensor electronics will capture the movement stimulus from the user while the mounted display allows the interactive graphics to be visualized.

The proposed device is named the Inertial input Measurement Visualization Unit (IMVU) and is shown in Figure 1.2. It is designed to be intuitive to use, and must therefore recognize appropriate motion-based human gestures when applied. The IMVU must also be ergonomic to use and provide a simple user interface, which requires only a mild learning curve for the user to use the device according to their visualization needs. Some of these design objectives are investigated in the user study conducted by us presented in Chapter 6.

The objectives of our research project are stated as follows:

- Design and implementation of a six degrees-of-freedom inertial sensing hardware systems.
- The software and algorithms, as well as the signal conditioning operations required to extract three-dimensional position and orientation information from the inertial sensing system.
- Interaction design for the motion gesture based exploration of simple three-dimensional visual data.
- A comprehensive user study of the proposed design (IMVU).
- Performance comparison between different Inertial Measurement Units (Gyro Free Inertial Measurement Unit and Gyro based Inertial Measurement Unit).

1.3. ORGANIZATION OF THIS THESIS

This report comprises of eight chapters and two appendices. *Chapter 1* provides an introduction to our field, while *Chapter 2* shows the literature survey we conducted during the initial and subsequent stages of the project. It also provides a pedantry of the various existing devices and methods available, commercially or as prototypes, and lists their capabilities, comparing them with that of our proposed device.

Chapter 3 provides a brief explanation of inertial systems and the challenges associated with their usage. It also provides the proposed design for our IMVU, and the mathematical proof of concept for the algorithm. This stage of the

project provided the necessary preliminary information for the solution we have developed.

Chapter 4 deals with the development and implementation of the various Inertial Measurement Units (IMUs). The implementation aspects for the interaction design are discussed in *Chapter 5*. *Chapter 6* reports on the two IMU designs and the user performance using the interface developed. *Chapter 7* discusses the various challenges of designing an inertial sensor based real-time motion tracking human input device. *Chapter 8* discusses the potential applications for our interface and speculates on the future research regarding our interface and associated applications.

This thesis also has two associated appendices that show the progressive prototypes that were developed to test the various aspects of the designs, the mathematical implementation of state space equations, which are necessary for the Inertial Measurement Algorithm.

CHAPTER 2

LITERATURE REVIEW

This chapter reviews the literature on various pointing or position tracking devices and techniques similar or related to the visualization device proposed in this project. New human input devices especially pointing devices show a trend towards new position tracking and pointing capabilities that go beyond the traditional mouse and keyboard. Also a number of commercially available devices are beginning to combine input and display interfaces for portable and mobile devices. The following sections briefly discuss these devices.

2.1. POSITIONING AND POSITION TRACKING SYSTEMS IN HUMAN COMPUTER INTERACTION

Position plays a very important role while communicating information. For example, written characters in any language are completely dependent on the position of points and their patterns thus formed. Similarly, pointing and touching a particular point on an object is dependent on our perception of position. How different objects look from different orientations and portions often gives us an idea of its overall 3D shape and form.

Some actions we perform are based on positions of various objects in space,

while others give us information based on their relative position. One example is sign language, which is dependent on the position and orientation of one's fingers and hands. Thus most of our gestures are based on position or orientation of hands and the path they take to get to that position as much as the symbolism they carry. When translating these human friendly interactions for a computer, a method for measuring position is necessary. Depending on the application and environment, this can be done through visual sensors such as cameras, feedback devices like lasers, ultrasound or RF transceivers, or inertial sensors. In this section, we discuss some of the interesting devices and methods employed for interaction based on position measurement and tracking. We shall also discuss very briefly, how the techniques are relevant to our investigation into a viable positioning method for our application.

The pen-like device defined in [19] (a simplified model shown in Figure 2.1) is the first example. It solves a position-tracking problem by using a 3-axis accelerometer and a 3-axis gyroscope for inertial measurement and trajectory estimation specifically for gesture recognition. With this pen-like device, one can write an alphabet in the air or on a surface and the algorithm will deduce the alphabet "written". The device also uses an error compensation technique called Zero Velocity Compensation (ZVC). Unfortunately this method requires one to store the samples between two points on the velocity signal which are zero or near zero, which makes it an offline process and therefore inapplicable to a real-time application. A similar device is proposed in [21] by just sensing the direction of gravity, however, this device is not as effective as the one proposed in [19] as the user studies show. Furthermore, [19] can recognize

more complex shapes as well. A pointing method employing two accelerometers is proposed in [20].

Figure 2.1. The wand-like device acts as a three-dimensional pen. A path resembling an alphabet can be traced using the device and the algorithm can easily classify the path to the associated alphabet. The device is discussed in detail in [19].

A method for visualization of cutting plane for volumetric textures is proposed in [22] (see Figure 2.2) focuses exactly on the problem/application we aim to solve. It uses visually identifiable interfaces to traverse or bisect a 3D model or volume. The interface is directly focused on virtual or augmented reality applications. However such a method requires elaborate setup that is not portable. Furthermore, it has disparate components, like a separate display device and several input devices and/or props to capture the user interactions. Another method similar in features to this approach is described in [23].

Figure 2.2. The visualization and interaction method shown above uses object and cutting plane props (the cube as the object prop and the wireframe square as the cutting plane prop as seen in the picture) captured by stereoscopic cameras to estimate position and orientation of the cutting plane and the object. The picture is taken from [22].

Position tracking and hand gesture recognition methods are proposed in [24, 25] (Shown in Figure 2.3) using a patch-colored glove as markers to recognize the fingers and palm. The position of these elements is estimated by occlusion mapping using cameras. Another system using accelerometers which does not track individual fingers, but the hand of the user is proposed in [26]. Both these methods are used for gesture recognition.

Figure 2.3. The hand based gesture recognition system defined in [24,25] uses stereoscopic cameras and multicolored glove to estimate the position of each control element (fingers, palm and wrist). The picture is taken from [24, 25].

An application of interest and great importance to the field of navigation is personal navigation especially in GPS-denied environments. Methods of this were proposed in [27] and [28]. It uses an Inertial Measurement Unit (IMU) worn by a person to track their movements over large distances. The method also employs a ZVC method to reduce errors from the position signal.

Position tracking employing other methods or sensors (for example acoustic sensors and radio) for various other applications including ubiquitous computing and device co-location was also investigated. A relative positioning method using acoustic sensors for ubiquitous computing was proposed in [29] and a device co-location method was proposed in [30] using a small RFM transceiver combined with three ultrasonic transducers mounted on a number of devices.

A 3D-mouse was discussed in [31] which uses two perpendicularly mounted planar (2-axis) accelerometers to detect the orientation (angular position) of the device much like the Wii-mote (Figure 2.4) or iPhone (Figure 2.5) do with their 3-axis accelerometers. The method measures the gravity vector and calculates the various angles to deduce the object's orientation. Various commercial devices such as the 3D Mouse, iPhone, iPad (See Figure 2.6), Wii-mote and the Sony Sixaxis PS3 controller, (See Figure 2.7) etc have also been inspired by this design.

New motion controlled input devices marketed by Sony and Microsoft also employ some of the methods mentioned earlier. The Sony's Playstation™ Move

controller employs occlusion mapping using a camera as well as inertial sensors to provide the user with complete 6 DOF and gesture recognition. Microsoft's Kinect recognizes gestures using cameras to detect the user's position and gestures. Also the new iPhone 4 employs 3-axis gyroscopes as well as the existing 3-axis accelerometer featured in the previous versions to provide the user with 6 DOF on a portable platform.

Figure 2.4. The Wii-mote. Used in conjunction with a base device, it can provide up to 6 degrees of freedom, however it requires unobstructed line of sight with the based device to work and a television as a display. The picture is taken from NYU home pages [46].

Figure 2.5. The iPhone. Intended as a next-generation Smartphone, it is a stand-alone computing device with later models equipped with proximity sensors, ambient light sensors, accelerometers, magnetometers and assisted GPS etc to provide the user a rich, seamless experience. However it is not capable of 6 DOF for human interaction. Picture taken from [47].

Figure 2.6. The iPad, It is a cross between a cell phone and a tablet PC. However it is much less capable than both iPhone and Wii-mote as far as 6 degrees-of-freedom are concerned. This is the closest device to the visualization device (IMVU) proposed in this project as far as form factor and gesture control is concerned. Picture taken from [48].

Figure 2.7. The Sony Sixaxis PS3 Controller is capable of 6 degrees-of-freedom without line of sight communication. Picture taken from [49].

Our study of viable positioning systems for HCI applications led to choosing inertial systems for various reasons. One reason is that inertial systems can be made into a compact, stand-alone system. Another reason was that inertial systems are able to capture applied forces, which at times would be enough information required to extract gesture information in addition to their position calculation capability, essentially providing a shorter process path. We started

with the study of inertial systems and the various intricacies associated with their design and implementation.

2.2. INERTIAL SYSTEMS

Inertial systems have been a well-researched field with various applications in military and aerospace. Most of this research has become quite standard over the years and has many textbooks on the subject [1, 2, 3]. However, new research is still underway especially in the field of strap-down systems, example Gyro-Free Strap-down systems. A strap-down system is one in which the sensors are mounted statically, i.e. directly onto the chassis of the object. Such systems are more complex as far as algorithm is concerned, but mechanically simpler than the older IMU which used Gyro stabilized platforms for mounting accelerometers.

Although gyro free inertial system itself is not a new concept, first proposed by DiNapoli in 1965 in his dissertation [6] and Schuler et al in 1967 in [4], it has only now become a viable real-time alternative due to powerful processors. This is because Gyro free systems require linear accelerometers configured in such a way so as to measure angular motion, which adds to the complexity of the Position Calculation Algorithm or Inertial Navigation Scheme (INS). This is discussed in detail in Chapter 3.

Inertial systems require complex signal processing [12, 13, 14]. Usually,

systems have a single channel while some systems may have two that are processed simultaneously by the associated signal-processing block [32], which is common in communications systems nowadays. However, inertial systems have a number of channels sending data into the signal-processing block. Each channel is associated to an inertial sensor. The complexity arises due to the fact that these channels are coupled together tightly because of their physical association to the sensors and thus the algorithmic description of these channels may change with varying motion executed onto the object.

The intrinsic problem with inertial systems is that they are by definition open looped systems of the second order. This means such systems are performing mathematical integration on their sensor data (signals from gyroscope and accelerometers) to calculate positional information. Thus when an error occurs it cannot be detected (by a feedback which is intrinsic in closed looped systems) and these errors get accumulated in the output data. As the system keeps running, more and more errors get accumulated until the system output becomes completely inaccurate and useless. These errors are ever present in minute quantities as noise (mostly Gaussian in nature) in the sensor readings. Thus a method is required to remove the said noise to make the system usable for greater duration. One solution for this is to use Wiener filtering; these filters extended to non-stationary processes much like ours are called Kalman filters [5]. These are adaptive filters used specifically to reduce the Gaussian noise from a system or signal. A lot of research has also been done regarding the implementation of Kalman filters in various applications [7, 8, 9, 10, 11].

In order to reduce the accumulated errors, many filter solutions have been proposed over the years [5, 33, 34, 35, 36]. Most of these methods require the aid of external sensors like GPS and RF beacons. The data from the inertial sensors and the GPS poses a sensor/data fusion problem especially for the orientation calculation. Our application does not have the luxury of a second disparate sensor to compare the INS outputs. However, the use of inertial sensors itself provides a second source for comparison. Statically, accelerometers and magnetometers provide a fairly accurate measure of gravity and magnetic north. However, dynamically these sensors provide an output distorted by the applied angular and linear forces. For the resolution of orientation, gyroscopes provide the accurate measurement of angular velocities, which are dynamic in nature, i.e. they are short-lived transients for most applications and stabilize to a constant value. The gyroscopes do not however provide a stable output for static situations due to noise being integrated, which perturbs the orientation outputs. This poses a two-fold problem of sensor fusion as well as filtering.

The aforementioned problem can be solved with the use of complementary filters in the frequency domain [34], the high frequency component from the gyroscopes being favored during motion, while the low frequency component being favored during the static operating regions. However frequency based filters are static in the frequency domain, and these ranges change dramatically from application to application. The definition of the static and dynamic ranges also becomes difficult where the two regions overlap considerably, as is the case with our intended application. Thus an adaptive filtering method is

essential. One option would be Kalman filters, however these filters may not be feasible for low-level application where computational power of the processor is low. They also require a good estimate of the noise beforehand thus requiring pre-computed noise analysis that would take significant time and expertise on the user's behalf. Another new method is proposed in [33] is a complementary filter approach applied, not in the frequency domain, but on the rotational matrix (Lie) group also known as the Special Orthogonal Group 3 ($SO\{3\}$). This method gives very good results for much less computation for our specific application and thus forms the basis of our new algorithm. It will be discussed in detail in Chapters 3 and 4.

The two configurations for accelerometers that inspired the proposed Gyro Free Inertial Measurement Unit (GF-IMU) design are defined in [15, 16, 17, 18]. The [15, 16] in essence improve the performance and sensor count in [17, 18], however since rotation has different effect on each configuration which results in different signal pathways, each paper gave a different insight into the functions of the GF-IMU. Detailed discussions about the capabilities and limitations of these configurations are provided in [15, 16, 18]. However, a brief and simplified introduction is necessary to provide an appreciation of the complexities and the extent of our implementation, which is provided in Chapter 3. The extent of implementation is the more salient attributes of the project. The designs proposed in the two papers are intended for vehicular navigation and guidance. Our application only takes a small subset from these designs to achieve our human interaction goals. Furthermore, the designs are only theoretical and simulated designs, therefore real-world prototyping of

devices built around the designs leads to many bottlenecks and intricacies which may veer our project from its intended goals, thus a customized sensor layout was developed.

Since our project is focused on the implementation of a positioning mechanism, no such comparison is provided between various positioning techniques. However, the application can certainly benefit from the fusion of various techniques as each technique complements another in its efficacy. Thus a sensor fusion problem arises which becomes an integral part of the system design. Several interesting works can be referenced for these techniques. However the work shown in [37] using complementary filter on the Special Euclidean group ($SE\{3\}$) is noteworthy. Another sensor fusion by complementary Kalman filter for an inertial based header tracking application provides an insight into a traditional Kalman filter based sensor fusion.

CHAPTER 3

INERTIAL SYSTEMS

3.1. INTRODUCTION TO INERTIAL SYSTEMS

Inertial systems, for example, INS and Inertial Guidance System (IGS), are systems that employ inertial sensors such as accelerometers, gyroscopes and magnetometers along with a computer algorithm to determine the position of an object through a process called dead reckoning. The dead reckoning is a process of calculating one's current position in relation to their previous position.

INS, due to being bulky and expensive, has been strictly associated with aerospace and military applications for positioning. Also since the inertial systems are immune to external jamming and infiltration, these systems tend to be more secure than other triangulation techniques that use signals from other external sources, such as radios etc. This encourages their use for military applications that require security and robustness. However, since the advent of Micro Electro Mechanical Systems (MEMS), the inertial measurement technology has become accessible to scientific and commercial applications.

One of the key advantages of using MEMS based inertial system for an application like ours is the fact that it is light, small and thus convenient for portability. Moreover, it does not require external reference signals. Thus it

eliminates the need for an elaborate setup that a comparable system would require, which may limit portability. The drawback of a system developed using inertial sensors is accuracy. Thus there is a compromise of accuracy for greater versatility (the ability of the device to be used in many environments), scalability (the ability of the device to be used in systems of various physical sizes and complexity), portability (the ability of the device to be light, compact and mobile) and tolerate interference. Since our focus are HCI related applications, where versatility and portability are more important criteria, making inertial sensor approach a more suitable choice. The lower accuracy can in many cases be elevated by an appropriate interaction design (See Chapter 5).

One of the key features of any inertial system is the placement and orientation of its sensors, as well as the choice of the sensors. This would determine some of the key components in the algorithmic data flow. This project pursues two separate threads in this regard. The first thread presented in this report is the Gyro-free inertial system, i.e., a system developed using only linear accelerometers placed in such a manner so as to give both linear accelerations, due to translations as well as the angular accelerations due to rotations. Such systems have more complicated algorithms for the resolution of various data components from multiple sensor readings. Previously, gyroscopes were not only expensive and required extensive maintenance, but were very bulky, which reduced the applications where they could be used. But now due to MEMS technology, this is not the case. With MEMS manufacturing technologies, gyroscopes and accelerometers can now be manufactured in extremely small sizes and packaged into integrated circuits making them more usable in small

consumer oriented applications. The second thread, which uses a combination of gyroscopes and accelerometers, was also undertaken; a comparison of the two systems will be shown in subsequent chapters.

Sensor placement is however an important component of a design in order to utilize the sensors for attitude (orientation¹) determination. Asymmetric placement (See Figure 3.1b) may require some transformations for the data that would require a more capable processor. An overall symmetric placement, like the one shown in Figure 3.1a, is computationally less complex to work with. Accelerometers may be placed around our display and input device in a strap down manner in a symmetric configuration, which would easily allow them to measure angular components and consequently resolve those using much less computation power and thus amenable to a microcontroller implementation.

¹ In satellite terminology, the term *attitude* refers to the orientation of the orbiting body with respect to earth or any other global reference, since inertial systems were initially developed for satellites, they carry a lot of terminology from aerospace applications in many textbooks. In this text, they may be used interchangeably. Robert J. Twiss, Eldridge M. Moores (1992). "§2.1 The orientation of structures". *Structural geology* (2nd ed.). Macmillan. p. 11. ISBN0716722526. "...the attitude of a plane or a line — that is, its orientation in space — is fundamental to the description of structures."

Figure 3.1. The two sensor configurations are examples of symmetric and asymmetric configurations. a) The configuration has sensors placed symmetrically on x and y-axes. b) The sensors are symmetric about y-axis (green) but asymmetric about x-axis (red).

3.2. THEORETICAL PRINCIPLES

In essence, the inertial systems measure forces exerted at a particular point. Much like any other instrumentation and measurement device, the inertial systems are composed of a set of sensors and an associated set of computational devices (analog or digital) which extract the relevant information from the sensors, format the sensor data, and eventually dispatch the relevant data to the intended party.

Inertial systems are mostly employed to measure position and/or the attitude of a frame of reference with respect to a base reference frame. Yet, the inertial sensors only detect forces applied to a particular point or reference frame origin. Therefore there is an intermediate step in between the data acquisition and the information transfer. This intermediate step essentially converts the

forces measured to position by employing the Newtonian mechanics (Newton's Second Laws of Motion and various derived equations consequently), and is called the inertial measurement scheme. This scheme translates the sensor geometry/configuration and the forces measured by those sensors to position and orientation information. The type of sensors used and the placement scheme always depends on the environment and the application.

Figure 3.2. The figure shows the orientation change and the respective angle changes to the global reference frame.

Figure 3.3. The figure shows an ideal placement scheme for gyroscopes and accelerometers to capture rotations and translations. In the ideal case, the sensors are infinitesimally small, orthogonal and can be mounted at the center of mass. Unless stated otherwise, we shall be using the axis color scheme shown, i.e. red for x -axis, green for y -axis and blue for z -axis.

For an ideal IMU shown in Figure 3.3, with infinitesimally small sensors mounted at the center of mass of the object, the sensors needed would be three accelerometers, each along the object's frame of reference axes and three gyroscopes mounted such that they measure angular velocity around each on the axes. The first thing the algorithm needs to know is the current orientation of the object. Since there is no method to measure the orientation with gyroscopes directly, we calculate the current orientation from the previous orientation by updating it with the change in orientation (angular velocity). As shown in Figure 3.2, the object frame of reference in red is the previous state of the object. After rotation, the object frame changes to the one depicted in grey.

For orientation, integration of angular velocity does not give the correct angular position of the object because of the coupling of orthogonal axes. For this reason, unit quaternion is employed to represent rotation of the object. Thus a

method to integrate quaternion is required in order to translate measured angular velocity to orientation. This is shown in the following equation.

$${}_n\dot{\mathbf{q}}_k = {}_n\mathbf{q}_{k-1} \cdot \mathbf{p}({}_b\vec{\omega}_k)$$

Eq 3.1

Where ${}_n\mathbf{q}_{k-1}$ is the quaternion representing the orientation of the object in the previous state, while ${}_n\dot{\mathbf{q}}_k$ is the derivative of the quaternion for the current state in the inertial frame of reference. ${}_b\vec{\omega}_k$ is the measurement of the angular velocity from the sensors in the current state. The operator $\mathbf{p}$ applied to body frame angular velocity vector converts it into a quaternion by adding a zero scalar value.

$$\mathbf{p}({}_b\vec{\omega}) = 0 + {}_b\omega_x \cdot \hat{i} + {}_b\omega_y \cdot \hat{j} + {}_b\omega_z \cdot \hat{k}$$

Eq 3.2

Where $\hat{i}$, $\hat{j}$ and $\hat{k}$ are the unit orthogonal vectors align with the x, y and z-axes in our frame of reference. By solving the differential equation in Eq 3.1, we can get the quaternion for the current state. This is presented in the following equation.

$${}_n\mathbf{q}_k = {}_n\mathbf{q}_{k-1} \cdot \exp(\mathbf{p}(\vec{\sigma}))$$

Eq 3.3

Where

$$\vec{\sigma} = \int_{t_{k-1}}^{t_k} \vec{\omega}. dt$$

Eq 3.4

Where, t_k and t_{k-1} are the points on the time axis which mark the interval where $\vec{\omega}$ is defined. The exponential function for a quaternion is defined as:

$$\exp(\mathbf{p}(\vec{\sigma})) = \begin{bmatrix} a_c \\ a_s \sigma_x \\ a_s \sigma_y \\ a_s \sigma_z \end{bmatrix}$$

Eq 3.5

Where σ_x , σ_y and σ_z are the orthogonal components of the $\vec{\sigma}$ vector and a_c and a_s are defined as follows.

$$a_c = \cos(|\vec{\sigma}|)$$

Eq 3.6

$$a_s = \frac{\sin(|\vec{\sigma}|)}{|\vec{\sigma}|}$$

Eq 3.7

The force measurements are influenced by gravity. The effects of gravity need to be removed before integrating linear acceleration to estimate the position. However since the force is measured in the body frame where as the gravity is in the inertial frame, the force vector needs to be rotated. The following equation depicts this.

$$\mathbf{p}(\vec{a}_c) = {}_n\mathbf{q}_k \cdot \mathbf{p}({}_b\vec{a}_m) \cdot {}_n\mathbf{q}_k^* - \mathbf{p}({}_n\vec{g}_m)$$

Eq 3.8

Where $\vec{a}_c$ is the acceleration vector for the current state compensated for gravity estimated for the current state. $\mathbf{q}_k$ is the rotation quaternion which converts the measured acceleration vector in the body frame to a total acceleration estimated in the inertial frame for the current state. $\vec{a}_m$ is the measured acceleration and ${}_n\vec{g}_m$ is the measured gravity vector.

Once the acceleration in the inertial state has been deduced, the position and the

velocity can be iteratively estimated with the use of these recursive kinematics equations.

$$\vec{v}_k = \vec{v}_{k-1} + T \cdot \vec{a}_c$$

Eq 3.9

$$\vec{p}_k = \vec{p}_{k-1} + T \cdot \vec{v}_{k-1} + T^2 \cdot \vec{a}_c$$

Eq 3.10

Where, $\vec{v}_k$ and $\vec{p}_{k-1}$ are the velocity and position calculated for the current state (or iteration). $\vec{v}_{k-1}$ and $\vec{p}_{k-1}$ are the velocity and position vectors from the previous state or iteration. $\vec{a}_k$ is the linear acceleration measured for the current iteration and T is the time taken between the current iteration and the previous iteration. We assume that T is significantly small and that any of the equation quantities do not vary significantly during this period.

The above set of equations can also be written using state space notation as mathematical shorthand as (See Appendix B).

$$\mathbf{x}_k = \mathbf{A} \cdot \mathbf{x}_{k-1} + \mathbf{B} \cdot \mathbf{u}_k$$

Eq 3.11

$$\mathbf{y}_k = \mathbf{C} \cdot \mathbf{x}_k$$

Eq 3.12

Where $\mathbf{x}_k$ is the state vector describing the current state of the system, while the $\mathbf{x}_{k-1}$ is the state vector describing the previous state of the system. The states of a system are certain variable parameters associated with that system that may change during the evolution of the system itself. These parameters define the system itself in the domain in which it is being observed. In our case of an inertial system, the state of the system can be completely defined with its linear velocity and position. These parameters will become the components of the state column vector. The column vector $\mathbf{y}_k$ is the output vector for the system, while $\mathbf{u}_k$ is the input vector. $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are the state, input and output transition matrices respectively. The above set of equations are a generalized mathematical short hand, for our particular system, when all the matrices and column vector components are inserted, this set of equations will resemble the equations of kinematics shown in Eq 3.9 and Eq 3.10.

A major justification for the use of state space method is ease of implementation because it abstracts the associated equations (in our case, the equations of kinematics) into a standard set of equation (Eqs 3.11 and 3.12). The state space approach also provides a method to implement the system ready for a slew of state filters, such as Kalman filters that improve the accuracy of the system many folds. The down side of this method is computationally intensive and taxing on the memory resources because the matrix data structures may contain unnecessary elements. However, with some optimizations, and with the use of floating point hardware on newer processors, the burden can be lightened which makes this approach feasible.

3.3. PRINCIPLES OF GYRO-FREE IMU

One of our proposed designs is a gyro-free scheme, which means the orientation update is not associated to the gyros, as is usually the case for a Gyro based inertial system. As this design scheme has only multiple accelerometers, the orientation information is coupled with the position information in the same accelerometer signals. Thus decoupling these signals is our first challenge and there are many ways to accomplish this.

Figure 3.4. The placement scheme to measure rotation about a single axis, currently it is associated to the z-axis. Rotations around z-axis mean that the object's orientation does not change about the z-axis, instead the x and y-axes will rotate. The large green and red arrows show the effect of tangential and centripetal forces on the accelerometers respectively.

Normally when systems employ accelerometers, single accelerometers are placed to register motion across a single axis. Some systems may put the

accelerometers at certain angles to increase sensitivity to certain axes more than the other. We place accelerometers about a moment arm as shown in Figure 3.4 so that they can capture unbalanced forces about that moment arm. These unbalancing forces applied tangential forces as shown by the large green arrows in the Figure 3.4. Once the object starts to rotate, the centripetal force also starts to manifest in the accelerometers mounted in line with the moment arm, in this case the x -axis. In Figure 3.4 the large red arrows show the centripetal forces. This forces always point towards the center of the rotation irrespective of the direction of the rotation.

Subtracting the axes signals from the respective axes of the two accelerometers placed on opposite ends of the moment arm decouples the rotational signal from the raw signal. This moment arm will be responsible for the motion across ² a single axis only, in the case of Figure 3.4, this axis is the x -axis.

The subtraction of the axes data gives us the force magnitudes for the y and z -axes ². The following relation gives the centripetal force.

$$F(t) = \frac{mv^2(t)}{r}$$

Eq 3.13

² The moment arm can be distinguished by the axis that will capture the centripetal force. However this will not be able to resolve the direction of the rotation. This information is deduced by the tangential forces. Thus the moment arm x shown in figure 3.4 will register rotations about z -axis and y -axis. In figure 3.5 the rotation about z -axis is shown.

Where $F(t)$ is the force function, $v(t)$ is the tangential or arc velocity, m is the mass (or inertia) and r is the radius of rotation.

We can integrate the tangential velocity once to find the change in angle (i.e. orientation). Integrating once would result in less error accumulation and faster processing, essentially giving almost the same result of a gyro. However the axes or direction of rotation cannot be resolved. The rotational direction information can be deduced from the tangential forces.

In order to account for all applicable rotations on an object (see Figure 3.5), there need to be three moment arms, similar to the one shown in Figure 3.4. Ideally each axis would be perpendicular to each other forming a Cartesian frame of reference. Tangential forces applied to each moment arm can be integrated twice to get the relative rotational angles by which the object is changed [16]. Another method proposed in [15, 17] uses the integrated tangential velocity to deduce the sign/direction of the angular velocity; this sign is combined with the centripetal force measured by the inline accelerometers to get the composite velocity. This composite signal is then integrated to get angular position. This method has one less integration, but due to incorrect sign deduction, it is prone to misinterpretation of direction. The centripetal force exerted on the object is measured by reading the sensors on the rotating moment arms. The sensor readings are subtracted to get tangential acceleration and centripetal acceleration (which is proportional to the squared of the

tangential velocity). This is depicted in the following equation.

$$|\omega_{c,m}| = \left| \sqrt{\frac{a_{n,m} - a_{n+1,m}}{r}} \right|$$

Eq 3.14

Where, $|\omega_{c,m}|$ is the magnitude of the angular velocity of object. Since this acceleration (calculated by subtracting the two inline components) is always in one direction i.e. pointing towards the center of mass of the object, we cannot deduce the direction of the rotation from this equation. As in the Figure 3.5, n denotes the sensor number ($n = \{1,2,3,4,5,6\}$), while m denotes the component of the sensor i.e. the axis ($m = \{x,y,z\}$). When used with the $\omega_{c,m}$ in the above equation, it denotes the moment arm for which the angular velocity is being measured. These moment arms shown in Figure 3.5 as 1, 2 and 3 can also be denoted by x , y and z respectively, by the axis in the object frame of reference to which they are parallel.

As shown by the Eq 3.14 and as explained earlier, we need a separate method to deduce the direction of the angular velocity (rotation), therefore we use a separate equation shown below, which employs integration.

$$\vec{\omega} = \omega_x \cdot \hat{i} + \omega_y \cdot \hat{j} + \omega_z \cdot \hat{k}$$

Eq 3.15

$$\omega_{c,x} = \int \frac{a_{1x} - a_{2x}}{2} \cdot dt$$

Eq 3.16

$$\omega_{c,y} = \int \frac{a_{3y} - a_{4y}}{2} \cdot dt$$

Eq 3.17

$$\omega_{c,z} = \int \frac{a_{5z} - a_{6z}}{2} \cdot dt$$

Eq 3.18

Where, by convention stated before $a_{n,m}$ denote the sensor and the axis used while $\omega_{c,m}$ denote the angular velocity for the particular moment arm denoted by the m symbol.

Figure 3.5. The placement scheme to measure all rotations for a Gyro Free Inertial Measurement Unit. Each sensor has a 3 axis accelerometer, each axis of the accelerometer is labeled by the convention anm (or $a_{n,m}$ in equation format), the n denote the sensor number as shown in the figure while m denote the direction. m is the member in the set $\{x, y, z\}$.

The second method to resolve the direction of rotations is to not use Eq 3.14 and only use the Eq 3.15, Eq 3.16, Eq 3.17 and Eq 3.18 to calculate the angular velocity. This method gives a smooth angular velocity signal but is more erroneous. This method is employed in [16]. After the angular velocity is resolved, Gyro Free Inertial Measurement Unit or GF-IMUs can use the Eq 3.3 much like the Gyro based devices.

Since there are a number of accelerometers present in a GF-IMU, the various accelerometers can be polled and composited together to reduce errors as much as possible before calculating translational motion. Once the algorithm deems all removable errors are reduced, the signal can be used by Eq 3.9 and Eq 3.10

to calculate the displacement over the interval by integrating the acceleration.

3.4. PRINCIPLES OF GYRO-BASED IMU

Although gyro based IMU is much simpler to construct mechanically and its algorithmic description is much simpler too, but since we built a prototype of each i.e. The Gyro based IMU and Gyro-free IMU, it would be prudent to have a discussion on the intricacies of the Gyro-based IMU in tandem with the GF-IMU.

As the name suggests, the Gyro-based IMU employs Gyroscopes to measure angular velocity. Since these sensors give a quite accurate reading of the angular velocity about their axis, there is no need to have moment arms, as was the case in a GF-IMU. Also, if the moving body is rigid, they can be placed almost anywhere about that object to measure the angular velocity with little or no transformation of the raw signal readouts. However, gyroscopes do have a varying bias that needs to be eliminated. This bias is more evident when the object is not being rotated, when the gyroscopes are supposed to give a zero output.

Although gyroscopes can be placed anywhere about the object, the challenge is to place accelerometers about the object. If the accelerometer is not placed at the center of gravity i.e. the point of rotation, then the accelerometers are subject to unbalancing forces, which may appear as linear forces at the

accelerometer outputs. Figure 3.6 depicts this concept.

Figure 3.6. The figure (a) shows the accelerometer measuring zero force when placed at the center of the gravity of the object, while (b) shows the accelerometer incorrectly measuring rotational force as linear force when placed at a displacement from the center of gravity.

Figure 3.7 The figure shows the sensor placement scheme in a Gyroscope based IMU. The accelerometer is placed at or as close as possible to the centre of gravity. If the accelerometer is at the centre of gravity and if the centre of gravity does not shift during rotational motion, no signal transformation is required. In the case of the gyroscopes, we only need to insure the rigidity of the object at the point of placement with the centre of gravity.

Thus the optimal position for the placement of the accelerometer is at the center of gravity for the device. When the accelerometer is mounted at a displaced point from the center of gravity, some transformation is required. In the IMVU,

the accelerometer is placed nearly at the center of gravity of our device. The small displacement from the center of gravity can be ignored in favor of reducing computations in the algorithm.

Since our device is essentially a rigid body, the gyroscopes can be placed virtually anywhere around it as long as the gyroscope's linear acceleration rejection parameter is sufficiently high. This parameter and its other characteristics can be found in the gyroscope's accompanying literature provided by the manufacturer. The overall placement strategy can be seen in Figure 3.7. The criterion for the placement of gyroscopes in order to measure the overall angular velocity of the object is that the point of placement be rigid with respect to the centre of gravity of the object.

CHAPTER 4

INTERFACE DESIGN

4.1. INTRODUCTION

After selection of the sensors, a potential placement strategy needs to be worked out along with a rudimentary orientation calculation algorithm. It was decided to test them in a manner that would not only allow us to learn how to use the sensors to make a feasible, working system but also to gain valuable insight as to what signals are generated by the sensors for a particular set of stimulus. For this we required several test devices employing the sensors and a test bench. A few such devices were built during the course of the project, which have been discussed briefly in Appendix A.

This chapter discusses some of the issues and challenges faced during the development of the numerous prototypes. However, the focus will be the Prototypes Version 1.0 and 2.0, where Version 1.0 is the Gyro-free proof-of-concept and Version 2.0 is the Gyro-based design with enhanced algorithm for position update.

Since the finalized algorithm can be applied to either Gyro-based or Gyro-free devices with only minor modifications, we shall abstract the choice of sensors

and placement strategies for the algorithmic description. We shall also focus on how an IMU for human interaction purposes is different from those employed for aerospace and robotic navigation. Figure 4.1 shows our IMVU and the various motions it is sensitive to.

Figure 4.1 The figure shows the various motions applied to the IMVU, The A shows the translations on the various axes, while B shows the various rotations. The translations are calculated by integrating the detected accelerations over time after gravity component has been removed from the acceleration signal. Our main focus is on the orientations which require integration of the sensor difference signal for the yaw angle (z-axis rotation) and decomposing the gravity along x and y -axis to calculate the pitch and roll angles.

4.2. SIGNAL DESCRIPTION AND CHARACTERISTICS

The difference between IMU used for aerospace and robotic navigation purposes, which is a subject of intense interest nowadays, is two-fold. But to

explain these phenomena, we need to discuss the types of movements and their effects on the signals generated. Earlier we discussed the abstraction of the sensors by categorizing the signals generated for each type of movement. These movements can be classified into two categories, translation or linear motion, and rotation. Ideally, the translations should only generate a linear acceleration signal while the rotation should only generate an angular velocity signal, regardless whether the angular velocity is being measured by Gyroscopes or accelerometers placed about a moment arm.

The accelerometer measurements are a combination of gravity and applied linear acceleration. Therefore the effects of gravity are indistinguishable from the effects of acceleration, i.e. the acceleration signal is constantly affected by gravity and its direction. These affects need to be removed in order to measure acceleration generated due to linear motion. The gravity is always pointing downwards and the sensors in a strap-down configuration are fixed to the chassis of the object. Thus the gravity vector/sensor rotates with the rotation of the object, therefore the components of the gravity vector change as the object rotates at a very low frequency. These changes in gravity are so slow that they can be regarded as the steady state of the total acceleration signal. However, if the object is rotated at a higher speed, the change in gravity can become quite large, and in such conditions it cannot be treated as a valid steady state.

While a rotation is being executed, the signal of interest is the angular velocity signal. However when the rotation stops, the angular velocity signal becomes less reliable due to noise. The angular velocity signal also has a slight bias,

which may vary from time to time even when no rotational motion is being applied. These are depicted in Figure 4.2 and 4.3.

Figure 4.2. The graph shows a typical angular velocity signal (blue) and its corresponding total acceleration (gravity and linear acceleration) signal (green). The blue signal shows short impulse like peaks corresponding to each rotation, while the total acceleration signal shows a step-like change corresponding to those rotations, which depict the change due to the rotating gravity vector.

The forces applied for movement of robots or small UAVs are quite significant and thus easily detected even by the less sensitive MEMS based accelerometers. Since the affects of the forces are intense and short-lived, they appear in the frequency domain such that they can be filtered out using a frequency based band-pass filter without the gravity changes affecting the output. However, frequency filters introduce significant delay for a real-time user feedback dependent human interface. The frequency response of the typical accelerometer signal is shown in Figure 4.3, which shows significant overlap between the effects due to gravity and effects due to linear motion.

Figure 4.3. The graph shows a typical frequency response for an accelerometer signal. The graph also shows low frequency regions that are affected by gravity and high frequency regions affected by applied forces. There is a slight overlap in the two affecting regions which is shown the red shaded region

Thus an adaptive filtering technique is required to maintain minimal delays between its input and filtered signals. The technique employed is an adaptive complementary filter acting on the Lie group (also known as the 3D rotation group or $SO\{3\}$) [33].

4.3. ADAPTIVE COMPLEMENTARY FILTERS FOR SPECIAL ORTHOGONAL GROUP

Due to the overlap between the frequency ranges for the low frequency gravity component of the total acceleration signal and the linear acceleration component, the frequency response of the filter needs to adapt in order to give a faithful signal. However, the adaptive filter needs a criterion for its adaptive

mechanism. This criterion is provided by the angular velocity signal. The filter is described in detail in [33]. We shall only give a brief description of this filter in this section.

This filter has three key components essential for its correct execution, a reference quaternion or Lie group matrix, the angular velocity vector, and the corrected output of the reference quaternion. The reference quaternion should give a steady state estimate of the orientation, thus the low pass filter signals of the total acceleration signal (combined preferably with a electronic compass signal to represent the complete orientation of the object) can be used. This quaternion combined with the previous output quaternion for the filter (feedback) is used to calculate an error quaternion. This error quaternion is used to update the output quaternion by inserting in the effects of the angular velocity signal. Mathematically this is shown by the following equations.

$$\mathbf{q}_e = \mathbf{q}_o^{-1} \cdot \mathbf{q}_m$$

Eq 4.1

$$\dot{\mathbf{q}}_o = \mathbf{q}_o \cdot \mathbf{p}(\vec{\omega} + \hat{b} + \hat{c})$$

Eq 4.2

$$\dot{\hat{b}} = -k_i \cdot \text{slr}(\mathbf{q}_e) \cdot \text{vec}(\mathbf{q}_e)$$

Eq 4.3

$$\dot{\hat{c}} = -k_p \cdot \text{slr}(\mathbf{q}_e) \cdot \text{vec}(\mathbf{q}_e)$$

Eq 4.4

Where $\mathbf{q}_e$ is the error quaternion, $\mathbf{q}_m$ is the measured quaternion from the

gravity vector and/or the magnetic north vector. $\mathbf{q}_0$ is the filter output quaternion. $\vec{\omega}$ is the measured angular velocity while $\hat{b}$ and $\hat{c}$ are the integrated and proportional corrective terms to remove bias and other unwanted signal components. k_i and k_p are the integrative and proportional gains for the filter. $slr(\)$ and $vec(\)$ are functions that extract the scalar and vector components of the quaternion argument respectively.

The key advantage of this filter can be seen in the frequency domain in Figure 4.4 where it exhibits adaptive behavior with the change in angular momentum. A low pass filtered total acceleration (the linear motion component removed) signal can be used in conjunction with this filter to fix the group delays produced by the low pass filtering operation.

Figure 4.4. The graph shows the adapting frequency response for the filter described above. The graph shows the gain-versus-frequency plot (bode plot) for the high angular momentum in red and low angular momentum in blue.

4.4. LINEAR ACCELERATION/VELOCITY EXTRACTION AND POSITION CALCULATION

The output unit quaternion from the SO{3} filter gives a quite precise estimate of the complete orientation of the object. From this quaternion, the direction of the downward pointing vector (gravity vector) can be calculated with a simple combination of the quaternion components.

$$\vec{g} = [q_b \quad q_c \quad q_d]$$

Eq 4.5

Where, $\vec{g}$ is the gravity vector, q_b through q_d are the quaternion components. More precisely, these components form the vector part of the quaternion.

Once the gravity vector is known it can be subtracted from the total acceleration vector measured earlier to get only the linear motion component of the total acceleration. However due to some minute delay introduced by the filter a small component of the gravity vector may remain. This undesirable gravity vector may be reduced by precise calibration as well as post-SO{3} estimation filtering.

Once the linear acceleration is known, it can be integrated to determine the linear velocity of the object using Eq 3.9. The integration is only performed when the change in acceleration increases beyond the noise envelope of the change in differential acceleration signal, which is pre-calculated during calibration. The calculated velocity has a drifting bias that needs to be removed.

This process shall be discussed in the following subsection.

4.4.1. Velocity Bias Removal

The velocity signal is the resultant of integration being performed on the acceleration signal whenever a significant change in the acceleration signal is detected. Our assumption is that the acceleration signal is a highly dynamic signal, which settles very quickly to a zero mean. Thus if the signal is only integrated when there is a significant change in it, most of the data is used in calculating the velocity signal. This is depicted in the Figure 4.5.

This assumption is based on the fact that the positive and negative peaks do not have significant area underneath them as compared to the area under the changing acceleration shown as the green shaded area in Figure 4.5. These areas are easily marked, as the differentiated acceleration signal (shown in Figure 4.5 middle) is well beyond the noise threshold in these regions, shown by the blue shaded regions in Figure 4.5. Thus when these peaks are not integrated they do not affect the overall shape and area-under-the-curve of the resultant velocity signal significantly, which are shown as the green signal in Figure 4.5. The percentage difference between the resultant velocity signal and the correct velocity signal was calculated experimentally to be about 5% to 15%, depending on the duration of the velocity. Note the comparison of the two signals in Figure 4.5 bottom. The two signal are shown as green (the optimally integrated resultant signal) and red (as the correct velocity signal).

Figure 4.5 The synchronized time domain signals show the acceleration signal (top) and its associated velocity signals (green and red at the bottom). The red velocity signal is the correctly integrated signal with the peaks integrated as well. The green signal shows the integration performed by our assumption that integrating only the significant changes gives a quite accurate alternative to the velocity signal. The peaks are quite small (in area) as compared to the changes thus they do not hold much energy to seriously hamper accuracy when not integrated. The signal in the middle is the acceleration difference signal. The blue shaded areas are when the difference signal exceeds the noise threshold. The signal beyond the threshold indicates that there is significant change in the acceleration signal; these are the green shaded areas on the signal. When only these areas of the acceleration signal are integrated, the resultant signal is shown in green. This signal (green) is an acceptable alternative to the signal in red.

Note a small velocity drift once the integration of the signal is complete in

either signal. Removing this drift is the primary purpose of the filter proposed in this section. The filter also smoothes the resultant velocity signal as well, while introducing minimal delay in the signal, especially when integration is being performed. This makes it a suitable alternative to Zero Velocity Compensation (ZVC) method for real-time applications.

The reason for calculating the velocity in this manner is to introduce less decisions and therefore a lesser number of algorithmic states in the algorithm flow. The biases in the velocity signal also need to be removed in order to integrate the velocity signal to get position. This is done by a variable window-averaging filter. This filter is defined by the following set of equations.

$$\bar{a}_k = \frac{n_1 \cdot \bar{a}_{k-1} + a_k}{n_1 + 1}$$

Eq 4.6

$$n_2 = A_1 \cdot \bar{a}_k + A_2 \cdot \bar{v}_k$$

Eq 4.7

$$\bar{v}_k = \frac{n_2 \cdot \bar{v}_{k-1} + v_k}{n_2 + 1}$$

Eq 4.8

Where, $\bar{a}_k$ and $\bar{v}_k$ are the running averages over the windows n_1 and n_2 . n_1 is a fixed window whose value ranges from 4 to 12 to smooth out the acceleration components. These running averages are calculated once every sample for each vector component separately. Thus there are separate averaging filters for

acceleration and velocity. n_2 is defined in such a way that if $\bar{a}_k$ and $\bar{v}_k$ are positive and negative respectively or vice versa, the n_2 approaches B which introduces a small integer offset ranging from 3 to 10. However when $\bar{a}_k$ and $\bar{v}_k$ have the same sign, n_2 becomes very large. Thus A_1 and A_2 are chosen so that their weighted sum tends towards zero when the signs are different and become very large when the signs are same. The filter is reset whenever the $\bar{v}_k$ approaches v_k .

4.4.2. Position Calculation

The position calculation is similar to the velocity calculation, where the difference of $\bar{v}_k$ and v_k is integrated whenever a significant change in that difference is detected. This is defined by the following equation, which is applicable to each axis.

$$p_k = \begin{cases} p_{k-1} & |v_k - \bar{v}_k| \leq V_{th} \\ (v_k - \bar{v}_k) \cdot T + p_{k-1} & |v_k - \bar{v}_k| > V_{th} \end{cases}$$

Eq 4.9

Where, p_k and p_{k-1} are the positions calculated for the current and previous sample respectively, v_k and $\bar{v}_k$ are velocity calculated and its respective bias filter output, and T is the sampling time. V_{th} is the pre-calculated threshold that controls the decision whether integration is necessary or not.

The Figure 4.6 shows the basic signal flow diagram for the algorithm.

Figure 4.6. The signal flow diagram for position and orientation estimation algorithm as discussed in Section 4.2 to 4.4.

4.5. PROOF OF CONCEPT PROTOTYPE (VERSION 1.0) FOR GF-IMU

The fully functional proof-of-concept design described here gives user full six-DOF control within the visualization viewport. It has four 3-axis accelerometers from Freescale Semiconductor Co, implementing an optimized version of the GF-IMU shown in Figure 3.5. In this design, one of the moment arms (z-axis moment arm) has been removed, as shown in Figure 4.7 and 4.8. The details on

how the missing moment arm can be compensated by using the other two moment arms will be discussed in Section 4.7.5. The design also incorporates several modes of operation to provide the user an ergonomic interaction design (Discussed in detail in Chapter 5).

Figure 4.7 Placement of the four accelerometers a1 to a4 with respect to the 3 spatial axes of the display plane. Sensors a1-a2 and a3-a4 form two moment arms 1 and 2 along the X and Y axis respectively. The radius, r is the uniform distance from the device center to each of the four sensors.

Figure 4.8 System layout. The four 3-axis accelerometers and the microcontroller kit are visible with the display moved out.

4.6. GYRO BASED IMU PROTOTYPE (VERSION 2.0)

Previously, the version 1.0 had a different algorithm implemented. However with the development of the Gyro based version (Version 2.0) and the parallel testing and comparison of the two led to similar modifications to its algorithm. Thus by the conclusion of the development stage, the two version had the same basic algorithmic framework. As the algorithm and its salient features have been discussed in Sections 4.2 through 4.4, the following sections will deal with the distinctions between the two versions, which are mainly in the hardware and firmware.

Since the version 2.0 is the Gyro based equivalent of the version 1.0, which allows for a stripped down design and less firmware and data acquisition procedures, the focus will be on the Gyro-free equivalent (version 1.0) as we delve into a discussion on the prototype implementation. For sensor placement issues refer to Section 3.4.

In accordance with the placement issues discussed in Section 3.4, the IMVU was designed such that the accelerometer was at the centre of gravity, and the gyroscope was at a rigid point with respect to the centre of gravity. This is shown in the following figure.

Figure 4.9 The placement of the gyroscope and the accelerometer in the Gyro-based IMVU. The design consideration was that the weight of the electronics (i.e., the display and the processing unit) be placed in the middle in layers as shown in Figure 4.8, so that the centre of gravity is not shifted.

4.7. IMPLEMENTATION

4.7.1. Inertial Input Measurement and Visualization Unit (IMVU)

The Tangible Interface Device has four 3-axis accelerometers in version 1.0, and one 3-axis accelerometer and one 3-axis gyroscope in version 2.0. Calibration can be performed separately rather than prior to each use. Only a small calibration step is performed before interacting with the device every time, although this can be bypassed to run on predefined values. The sensors signals are sampled at the rate of 30 Hz. 30 Hz provides a sampling rate that is fast enough to capture human stimulus for our algorithm. A lower sampling rate may alias some of the stimulus signals to a lower frequency. A communication layer implemented at the Sensor Data Acquisition Module (Sensor and Hardware Layer shown in Figure 4.9) and the Hardware Access Layer in Figure 4.9 allow for fast transmission stream from the hardware.

4.7.2. Computer-end Algorithm and Display Software

The flow control has been implemented to allow for better communication with the hardware. This has been implemented in a driver/hardware communication layer. A second modular layer has been added for filtering. A third layer has been implemented to calculate the velocity, distance and orientation using the GF-IMU methods specifically in the version 1.0, while the version 2.0 has the stripped down version of the same. The modular architecture of the software was written with migration to an embedded platform in mind is shown in the Figure 4.9.

Figure 4.10 System architecture overview. Each layer is modular with well-defined interfaces that can easily be removed, modified and re-inserted to work without affecting the rest of the layers. All this layers will be common to each visualization application.

4.7.3. Data Acquisition

The communication mechanism between the display interface and host PC is the standard serial port operating at 115200-baud rate. At each sample, a 16-byte packet is transmitted from the data acquisition module to the computer. To reduce framing errors, an alignment header and terminator byte is included in the packet together with a checksum to authenticate the packet. Within the

packet, byte stuffing for the header and terminator bytes as well as the marker bytes is performed at the transmitting end. The byte-unstuffing is done at the receiving end. A simple byte checksum is used to perform the data integrity test. Also the computer or host device can reset or stop the data transmission by sending a control message. The control message has no checksum, only a start byte and a known control word. Currently there is no mechanism to indicate whether the message has been received or not. The four 3-axis accelerometers are sampled at a rate of 30Hz and the data is transmitted synchronously with the ADC for version 1.0, while the same sampling rate is maintained for version 2.0, which has less number of channels to sample, namely 6. The algorithm uses the previously sent valid packet if a data packet is compromised to maintain a constant data rate for the algorithm. The data is sent in 12-bit ADC integer format. At the computer end, when the data is received, it is converted into floating point.

4.7.4. Signal Extraction and Conditioning

The algorithm runs right after the data acquisition shown in Figure 4.9. For version 1.0, once the packet is received, converted to usable floating-point values, and moved into the accelerometer data-structures, the accelerometer values on each moment arm are subtracted to give tangential and centripetal force values. This is given by the following equation.

$$\vec{a}_\theta = \vec{a}_1 - \vec{a}_2$$

Eq 4.10

$$\vec{a}_\phi = \vec{a}_3 - \vec{a}_4$$

Eq 4.11

Where $\vec{a}_\theta$ and $\vec{a}_\phi$ are the angular components (the forces generated due to rotations) though they are not exclusively associated with θ and ϕ as depicted. Here θ and ϕ are only used to differentiate rotational components from linear components. $\vec{a}_1$ and $\vec{a}_2$ are 3-axis accelerometers on moment arm 1. $\vec{a}_3$ and $\vec{a}_4$ are the 3-axis accelerometers on moment arm 2. The different axes in $\vec{a}_\theta$ and $\vec{a}_\phi$ have different physical interpretations. This is depicted in the Figure 4.10.

Figure 4.11 The forces induced and applied during rotation on the left, the actual sensor axes on the right. Both images show the top view. When an object is rotated, the user applies tangential forces tangent to the rotation of the object. Centripetal forces are induced during the rotation, which always point inward (towards the center). In the image on the left the blue arrows show the tangential forces, while the red arrows show the centripetal force. In the image to the right, the blue arrows show the accelerometer y-axes while the red ones show the accelerometer x-axes. Thus for moment arm 1 the centripetal force would appear on the x-axis while for the moment arm 2, the centripetal force will appear on the y axis.

The tangential forces will appear on the y and/or z-axis signal when the moment arm 1 is being rotated while the x-axis signal will contain the centripetal forces in the $\vec{a}_\theta$ vector signal. Similarly the tangential forces will appear on the x and/or z-axis when moment arm 2 is being rotated while the y-axis will show the centripetal forces in the $\vec{a}_\phi$ vector signal. No such conversions are necessary for version 2.0, which only needs to read gyroscope values to measure accurate angular velocity.

4.7.5. Discussion on the Missing Moment Arm

This section is specific to Gyro-Free Inertial Measurement Unit (GF-IMU) based Version 1.0 proof-of-concept. In the previous chapter, we discussed a method to detect rotational components of a motion applied to an object with linear accelerometers (Section 3.3). Placing accelerometers along and perpendicular to a moment arm captures the centripetal and tangential accelerations in their respective accelerometer axes when the moment arm is rotated. Three such moment arms are required to capture all applied rotations. Earlier we discussed that the gravity vector can also aid us to calculate the orientation much more accurately than the integration method. By combining the two methods, we can remove one moment arm (z-axis moment arm) and divided its function to the remaining two moment arms. This method is only applicable to our application as the low frequency components generated due to gravity are much larger than the higher frequency components generated due to

applied rotational forces (centripetal and tangential forces as shown in Figure 4.10). By detecting tangential forces about any one of the moment arms we can ‘multiplex’ between the rotations about the x , y and z -axes. This principle is detailed in the Figure 4.11.

Figure 4.12 The figure shows the forces generated when a particular rotational motion is applied. In A, the rotation axis is the x -axis, in this rotation the gravity vector also rotates, changing along the yz -plane (not shown). Similarly, B shows the rotation for the y -axis. Both of these rotations can be implemented by observing the change in the gravity vector. However, to incorporate the rotation about z -axis, as shown in C, we have to observe the generated tangential and centripetal forces. Notice the differences in the direction of these forces. This shall become the basis for our multiplexing scheme. The red, green and blue arrows denote the x , y and z axes along the device frame of reference.

As stated earlier, the gravity manifests itself at a very low frequency in our accelerometer signal. Due to this somewhat static nature of the gravity, we can use it to calculate the precise orientation of the IMVU. However to incorporate the z -axis rotations, in which gravity changes minimally, the dynamic components manifested by applied rotational motion needs to be integrated according to Eq 3.15 and Eq 3.18. However, all these forces (Gravity as well as rotational forces) need to be isolated to be used for their respective roles using a state machine. This rotation detection and isolation is referred to as multiplexing between the rotating axes. Its mathematical implementation is detailed in Eq 4.12, which shows the orientation calculation when any single moment arm registers a rotation being applied (Figure 4.11).

$$\theta_k = \begin{cases} \cos^{-1}\left(\frac{\vec{g}_z}{\sqrt{\vec{g}_z^2 + \vec{g}_x^2}}\right) & |\dot{\vec{a}}_z| < (3 \cdot E[\dot{\vec{a}}]) \wedge |\vec{a}_z| < (3 \cdot E[\vec{a}]) \\ \theta_{k-1} & |\dot{\vec{a}}_z| \geq (3 \cdot E[\dot{\vec{a}}]) \vee |\vec{a}_z| \geq (3 \cdot E[\vec{a}]) \end{cases}$$

Eq 4.12

θ_k is the pitch angle for the current state k , and θ_{k-1} is the pitch angle for the previous state $k-1$, $\vec{a}_z$ is the total applied force in the z -direction as measured by the accelerometers. Similarly, $\vec{a}_y$ is the total force applied in the y -direction as measured by the accelerometers. The ‘dot’ notation denotes the derivative of the respective signals while the ‘ E ’ notation denotes the standard deviation for the signal. The $\vec{g}_y$ and $\vec{g}_z$ are the ‘ y ’ and ‘ z ’ components for the measured gravity vector. The $\wedge$ and $\vee$ are the logical AND and OR operators respectively.

This equation illustrates the coupling between this absolute method and the

dead reckoning method shown in the previous chapter. The dead reckoning method is only used when the second condition holds true in Eq 4.12 i.e. when the rotation about z -axis is applied. When the second condition in Eq 4.12 holds true, the integration is being performed either for translations or the z -axis rotation and the state of the gravity dependent rotational axes i.e. x and y -axis is kept the same.

The rotational motion depicted in Figure 4.11C can only be calculated using the moment arm method discussed in Chapter 2. Since both the moment arms will be active during the motion as opposed to the activity in only a single moment arm for rotational motions in Figure 4.11A and Figure 4.11B, we can easily differentiate between the two. Also note that the directions of the tangential forces generated for different rotations are not the same.

The Figure 4.11A, Figure 4.11B and Figure 4.11C show all these intricacies put into the overall positioning algorithm. Once the orientation has been resolved by the aforementioned method, we can proceed to find the position. The accelerometer sensors are averaged to reduce errors between individual sensors before integrating.

4.8. MEASUREMENT TESTING

The IMU testing and calibration is a quintessential component in the development of inertial systems. However, since the development of the IMU is only a small segment of the overall design process for the IMVU human

interface, it shall be discussed only briefly since the validation of the IMU building block is necessary.

In this section, measurement data collected for three axes (out of the overall six axes, i.e. y-rotational axis, x-translation axis and z-rotational axis) of the GF-IMU as well as Gyro-based IMU will be presented and discussed. Only three axes are in focus as the data shown will be similar for the remaining axes.

The two IMUs were mounted on the same chassis and moved to a graduated test distance on the x-axis (rotated to a graduated angle on the y-axis for rotation) and the measured value was read from both the IMUs. The settling time for the IMUs was also noted. The graphs in Figures 4.13 and 4.14 depict these results.

The distance measurements for the Gyro-based IMU, as shown in Figure 4.14, are slightly less precise as the standard deviation is higher than that of the GF-IMU as shown in Figure 4.13. As the distances become larger the error grows in both IMUs, which can be seen by the increasing trend in the standard deviations. This is because of the dead reckoning method which accumulates data as well as errors over time. The slightly higher standard deviation for the Gyro-based IMU can be attributed to the fact that only a single accelerometer is used to measure the acceleration, where as the GF-IMU uses 4 accelerometers which are averaged for translation calculations. This averaging of sensor measurements minimizes the error significantly.

Figure 4.13 The graph shows the measured distance when the GF-IMU was moved against the test distance for ten instances of the experiment per test distance. The test distances are 2, 5, 10 and 20 cm. The measured values are also on the centimetre scale. The standard deviation brackets are shown to be very low especially in the case of the first two entries.

Figure 4.14 The graph shows the measured distance when the Gyro-based IMU was moved against the test distance for ten instances of the experiment per test distance. The test distances are 2, 5, 10 and 20 cm. The measured values are also on the centimetre scale.

The Figures 4.15 and 4.16 show the typical settling time for the GF-IMU and the Gyro-based IMU during rotational motion testing. The IMUs were rotated to test angle and the settling time for the IMUs was noted.

Figure 4.15 The typical settling time for the GF IMU is measured to be 287 milliseconds for y-axis rotations.

Figure 4.16 The typical settling time for the Gyro based IMU is measured to be 166 milliseconds for y-axis rotations.

As the Figures 4.15 and 4.16 shows, the settling time for the Gyro-based IMU is significantly less than that of the GF-IMU. This is because the gyroscopes

provide the angular velocity in the Gyro-based IMU with significantly less delay than that measured using the two-accelerometer moment arm in a GF-IMU. However the precision and accuracy is similar for both IMUs as show in the graphs in Figure 4.17 and 4.18.

The graphs in Figure 4.17 and 4.18 show that the standard deviations for the GF-IMU and the Gyro-based IMU are quite similar as both the IMUs calculate orientation with the help of the gravity vector once the IMU has stopped rotating i.e., settle to a stable position. Also the mean for the measured values are quite closer to the test value, unlike in the case of the distance measurement. This is because the orientation angles are calculated from the gravity vector measurement, which is static in nature where as forces measured for displacement measurement are dynamic (or transient) in nature i.e., the forces die out quickly and settle to a mean value where as the gravity vector keeps its state once the device has settled.

The characteristics of the other remaining axes are similar to the two discussed herein, and thus will not be discussed separately.

Figure 4.17 The graph shows the measured angle when the GF-IMU was moved against the test angle for ten instances of the experiment per test angle. The test angles are 10, 20 and 30 degrees. The measured values are also on the degrees scale.

Figure 4.18 The graph shows the measured angle when the Gyro-based IMU was moved against the test angle for ten instances of the experiment per test angle. The test angles are 10, 20 and 30 degrees. The measured values are also on the degrees scale.

CHAPTER 5

INTERACTION DESIGN

This chapter describes the various considerations in the design of the interactive exploratory visualization features of the IMVU. Once position and orientation information is available from the method discussed in Chapter 4, we can incorporate a seamless user interaction layer on top of it to enable the device to recognize gestures.

The underlying positioning algorithm only outputs the calculated position and orientation of the device in the real world. However there are various constraints for the user's motion. Because of these constraints we have implemented an interaction layer to provide the user with an ergonomically considered means to manipulate the IMVU. The user interaction design also ameliorates some of the deficiencies inherent in our method of position calculation (i.e. Dead Reckoning). Dead reckoning method employing inertial sensors is prone to minute errors in the sensor signals. Since these signals are integrated, or rather accumulated over time, so are these errors. As a result, the system becomes inaccurate after some duration unless there is a method to remove these errors via either an external source or any other algorithmic means.

Figure 5.1. (a) The inertial-controlled display unit with its respective 3D labeled axes. Touch-buttons on the left and right allow the user to turn motion sensing on or off at any time. (b) The image on the IMVU's display is extracted from a 2D visual slice whose 3D position and orientation can be controlled directly through the corresponding physical manipulation of the device.

5.1. VISUALIZATION REQUIREMENTS

Chapter 1 suggested the visualization of MRI data as a potential application, as shown in Figure 5.1. In visual exploration of 3D volumetric datasets (e.g. MRI data), different visualization dimension in the 3D space requires different degree of interactive control and precision. Consider the visual view changes due to translation in the x and y directions of the IMVU as shown in Figure 5.2A. Accurate location of a particular image feature to a specific display screen position is not important, since no additional structural information of the embedded 3D object can be solicited with precise x and y translational control. This argument is also partially true with the rotation of the 2D visual slice plane about the IMVU's z -axis as shown in Figure 5.2B. Only coarse angular positional control is required to get the current orientation of the image section in the 2D visual slice into what the user may consider the near upright position for effective visual recognition and interpretation. Precise angular alignment is

not required to extract further understanding of the image structure, in any case, a slight head tilt by the user can further compensate for any shortfall in the current orientation.

Figure 5.2 Coarse mapping required for A) positional control of translational motion in the X and Y directions and B) angular control of rotational motion in the Z-axis.

However, there are spatial dimensions in the 3D visualization space, in which precise control is important as the perceived image structure of the embedded 3D object can change significantly with minor angular tilt or translational displacement of the IMVU. Figure 5.3A shows an example of rotating the IMVU about the x -axis (the same is true of a tilt about the y -axis), where a significant change in the 2D image slice occurs with a small change in rotation angle. Figure 5.3B shows another such example resulting from moving the IMVU along the z -axis.

Figure 5.3 Precise mapping required for A) angular control rotation about the X-axis and B) positional control of translational motion along the Z-axis.

5.2. DISPLAY AND SENSING CONSTRAINTS

With the desired visualization requirements stated in section 5.1, these must now be mapped into the constraints presented by both the ergonomics of using a portable LCD display screen and the different sensing capabilities of the inertial sensors in the various spatial axes as described in Chapter 4.

5.2.1. Display Ergonomics

Firstly, the IMVU is designed with a flat LCD display with viewing angle of $\pm 60^\circ$ in left-right direction and $\pm 30^\circ$ in the top-bottom direction. However, from

empirical studies with several users, the comfortable natural inclination about the x -axis (see Side view in Figure 5.4) is about 30° . Ergonomically comfortable wrist flexing about the x -axis is about $\pm 30^\circ$ about this natural inclination, which corresponds well with the top-bottom viewing angle of the LCD display used. Rotation limits about the y -axis (see Front view in Figure 5.4) was set to $\pm 45^\circ$ about the horizontal as further tilting produces too much visual depth differential between the left and right edges of the display screen. Figure 5.4 shows the limits of the various rotational gestures adopted in interaction design strategies in Section 5.3.

Figure 5.4 Rotational limits used in the interaction design.

5.2.2. Sensor Constraints

There are basically two categories of sensor information produced by our inertial sensing system. The first is *static information*, (previously we have mentioned these as low frequency components in the sensor signal) which

changes when the sensor orientation changes with respect to earth's gravitational force. They are useful in relaying absolute attributes such as the current tilt of the IMVU with respect to the floor and are exploited in providing the precision control features of the IMVU. The other type of sensor information is the *dynamic information*, (previously described as high frequency components in the sensor signals, and thus the name dynamic components) which is solicited from the forces acting on the sensor during rotation (tangential forces) or non-constant translational motion of the sensor. Since such information are only present when the user apply some kind of motion to the IMVU, this dynamic information is exploited in our design of gesture-based interactions, which will support our implementation of the coarse control features.

5.3. VISUAL INTERACTION MODES

Due to the display ergonomics and the different types of sensor information available, we have proposed three modes for visual interaction with the proposed IMVU. They are the *Browse*, *Follow* and *Jog* modes.

5.3.1. Browse Mode

Consider a case in which the user wishes to rotate the current view by a large angular change of 90° about the z -axis. Given the ergonomically limited rotation that can be applied to the IMVU, this desired angular change can be initiated by executing a strong long jerk-like rotation of the IMVU in the

required angular direction (see Figure 5.5A). The displayed image will then execute a continuous rotation in the specified direction. The user stops the rotation by applying a gentle rotation to the IMVU in the reverse direction (see Figure 5.5B). Nuances such as the speed of continuous rotation can be controlled by the strength of the initial forces applied to initiate the browse mode. This same browse mode gestures can be applied to rotations in other 2 axes or translations in any of the 3 spatial dimensions. In short, browse mode allows the user to move through large portions of the 3D data set where detailed visual analysis is not required and the goal is to get to a visual region of interest quickly and but in a visually exploratory manner.

Figure 5.5 Z-axis rotation example. A) Starting continuous rotation in the Browse mode with strong rotation gesture. B) Stopping Browse mode with a gentle rotation gesture in the opposite direction.

To illustrate the difference in implementation of the interaction design for different axes, we describe the z -axis and the x -axis rotations. As noted, the z -axis rotation uses dead reckoning of the tangential forces to estimate the angular displacement. On the other hand, the x -axis rotation uses the change in the direction of gravity to calculate the orientation, which therefore allows the user

to control the absolute orientation. The Browse mode allows the user to visually parse through the 3D dataset quickly in order to find a region of interest and the rate of browsing (usually associated with the user's familiarity with the dataset) is controlled by the angular velocity about the specified axis. The condition to deactivate and activate Browse mode for Z-axis rotation and the subsequent angular velocity is governed by

$$\omega_z = \begin{cases} A.(\vec{\omega}_i \cdot \vec{g}_i) & (\vec{\omega}_i \cdot \vec{g}_i) > D_{az} \\ 0 & (\vec{\omega} \cdot \vec{g}) \neq (\vec{\omega}_i \cdot \vec{g}_i) \wedge (\vec{\omega} \cdot \vec{g}) \neq 0 \end{cases}$$

Eq 5.1

Where $\vec{\omega} \cdot \vec{g}$ is the current angular velocity for the z -component in the inertial frame. Once browse mode is activated and is converted into an appropriate angular velocity via a pre-defined scale factor A . The initial angular velocity along z -axis in the inertial frame $(\vec{\omega} \cdot \vec{g})$ activates the Browse mode when it exceeds a user definable threshold of D_{az} and this can be realized by simple rotation gesture about the z -axis. Once browse mode is active, the angular velocity of the displayed rotation about the z -axis will be at a constant value proportional to the strength of the initial rotational jerk gesture. The conditions to exit Browse mode are described in the lower portion of Eq 5.1. The quantities for the initial angular velocity $(\vec{\omega} \cdot \vec{g})$ and the current angular velocity $(\vec{\omega} \cdot \vec{g})$ must be non-zero (i.e. rotation gesture is present) and the current force is in the opposite direction from the initial force. To switch out of Browse mode, the user simply has to apply any amount of rotation in the opposite direction from the initial force. The described implementation is applicable to other motions using dead reckoning, namely translations in the x , y and z -axes.

The Browse mode for rotation about the x and y axes is implemented differently from what was described for the z -axis as we can take advantage of the absolute orientation sensing capabilities about these two axes. The angular velocity about the x -axis is governed by

$$\omega_x = \begin{cases} A \cdot (\theta - D_R) & |\theta| > D_R \\ 0 & |\theta| \approx N_x \end{cases}$$

Eq 5.2

Where, θ is the orientation about the x -axis measured relative to the neutral orientation position given by N_x (see Side view in Figure 5.4). The positive value A is a pre-defined scale factor. The Browse mode is triggered when the user changes the orientation of the display plane beyond a threshold value of D_R about the neutral orientation position N_x . In our experiments, we set D_R to about 15° . Once the device is in Browse mode, simply varying the orientation of the display plane about the x -axis can change the browsing rate. We exit Browse mode when the user brings the device back to its neutral position, N_x . The same method is used to design the Browse mode for the y -axis rotation. A is the scaling factor which indicates a preset speed of rotation assigned by the user.

5.3.2. Follow Mode

The Follow mode uses the changes in the static information produced by the inertial sensors to allow the user to have direct control of the current orientation of the 2D visualization plane. In the Follow mode, slow deliberate tilt of the IMVU about the x and y axes will provide corresponding changes to the

absolute orientation of the 2D visual plane; provided the current orientation θ_x (or θ_y for the y -axis) keeps within the D_R threshold given in Eq 5.2. This is part of the precise control feature of the IMVU, which allows the user to make detailed visual analysis of the image generated by the 2D visualization plane when it is carefully adjusted to the desired 3D orientation in the visualization space. Follow mode is available for rotation about the x and y axes only. As highlighted in Figure 5.3A, these are the operations that required precise control.

5.3.3. Jog Mode

The jog mode attempts to emulate the Follow mode visualization capabilities so that the user can bring and then hold the 2D visual plane in any arbitrary 3D position and orientation. However, because several spatial axes do not produce meaningful static information, the user's dynamic gestures are required to make incremental angular and positional changes to the 2D visual plane. In the Jog mode, small light jerks in the required direction that do not exceed the threshold of D_n where $n=\{x, y, z\}$ assessed for translational axes from the Eq. 5.1 are used to move the 2D visual plane with users' jerks in the corresponding direction. The motion of the visual plane will be maintained until the user's jerk gesture comes to a complete halt. This is not a very efficient way to get to a precise 3D position as short incremental jerks make the process slow and long incremental jerks prevents fine grain positioning and may cause an overshoot of the intended target position. The reason for this is the choice of lower thresholds. The integration of the dynamic components is only performed when the

difference signal exceeds these lower thresholds. Thus the user has to apply a force to exceed these thresholds in order for the device to measure position. This hinders fine grain position control to some extent. However the thresholds can be reduced to gain more sensitivity, but this may allow some noise to cross over and get integrated as well, thus producing drifts. Fortunately, the Jog mode is implemented for visualization operations that required only coarse control (see Figure 5.2). The one exception is the translational motion along the Z-axis, which also uses the Jog mode when precise Follow mode control would be preferable. The jog mode velocity is given by:

$$v_m[k] = \begin{cases} v_m[k-1] + T \cdot a_m[k] & |\dot{a}_m[k]| > D_m \\ 0 & |\dot{a}_m[k]| \leq D_m \end{cases}$$

Eq 5.3

Where, $v_m[k]$ is the current velocity and $a_m[k]$ is the current measured acceleration. $v_m[k-1]$ is the previous velocity calculated and T is the sampling time. D_m is the threshold for the particular axis denoted by m . The thresholds are extracted from recorded data during calibration. The m subscript, as earlier, denotes the x , y and z -axes for the device.

With the interaction design in place, the user will be able to use the device to control their viewing position and orientation in the 3D virtual space displayed on the IMVU screen. Some degree of adoption to this interaction design is required by most users before reasonable control of desired on the IMVU can be obtained. This is discussed in detail in Chapter 6.

The interaction design is part of the application and not the firmware, which is the case for the Positioning layer (or Algorithmic Layer defined in Figure 4.8). Therefore the developer or the user can modify the interaction layer thresholds in order to customize the interface to their needs.

CHAPTER 6

USER STUDY AND DISCUSSION

The goal of this project, as stated earlier in Chapter 1, was not just to design a motion based visualization unit, but to prove to some extent that such an interface can be a viable alternative to the traditional mouse, keyboard and GUI based interfaces so prevalent today. To achieve this goal, a series of user-oriented tests were conducted during the various stages of the project to measure key performance indices from the IMVU as well as traditional interfaces for a comparative analysis. The user studies were composed of users that would potentially be using such a device for their visualization applications such as avid gamers, 3D model designers, animation artists, mechanical engineers, electronics engineers, geologist etc. The experiments were designed to measure the key performance indices for the interface, which are intuitiveness, responsiveness, sensitivity, perceived ease of use, and learning period.

There were three experiments performed. The first experiment was strictly a performance evaluation experiment which allowed us to distinguish between the performances of the two IMU i.e. the Gyro based IMU and the Gyro Free IMU. The second experiment was designed to differentiate the IMVU from the legacy input devices used for 3D exploration. The third experiment shows a macroscopic level evaluation of the IMVU when compared with these legacy

devices. These experiments and their results are detailed in the following sections.

These series of experiments were done on an HP® Pentium™ 4 PC with 2 GB of memory and running at 3.0 GHz. A custom rendering engine was built using Microsoft® DirectX® 9.0 SDK with an nVidia® QuadroFX Graphics Processor Unit. In order to compare the IMVU with the mouse and keyboard interfaces, we first need to define an interaction design associated with these interfaces in order to manipulate a 3D view. In the next section, such interaction designs will be discussed.

It should be noted that since our goal is to test the IMVU solely as an interactive, human input interface, the experiments shown hereafter are design to depict the device performance from such a perspective. The efficiency and accuracy of the inertial measurement itself are not the scope of our research and thus are not discussed in detail. Ideally, our interaction design should abstract the intricacies of the inertial measurement principle and its limitations from the user.

6.1. MOUSE-ONLY AND KEYBOARD-ONLY INTERFACES

The interface design for the mouse-only interface is inspired by the Autodesk Maya™ 3D modeling software's mouse interface with a few modifications. By default the x -axis and the y -axis motion of the mouse controls the x and y -axis

translations of the view. The scroll wheel controls the z -axis velocity. This mode is called the translation mode. When the left mouse button is pressed, the mouse interface switches to the angle mode, in which the x and y -axis motions control the Yaw and Pitch angles in the virtual world. The scroll wheel controls the Roll angle. This is shown in Figure 6.1 and 6.2.

Figure 6.1 The figure shows how the mouse actions are mapped on the actions of the 3D virtual world. This shows the translational mode, in which mouse motion (Shown in A) is used to translate the user view/camera (shown in B). The only exception is the use of the scroll wheel, which is used to control the velocity along the z -axis (shown by the blue arrow in B).

Figure 6.2 The figure shows the rotational mode mouse interaction that activates when the left mouse button is pressed (signified by the bright red arrow). The scroll wheel controls the z -axis rotation also called roll. The up-and-down movement controls the pitch angle, while the left-and-right motion controls the yaw angles.

The keyboard-only interface only maps 3D navigation actions onto the various keys. The key mapping is shown in Table 6.1.

Table 6.1 The table shows the keys mapped onto the various navigation actions in the virtual world

Keyboard Keys	Virtual World Actions
'Up' Key	Move (Translate) Up
'Down' Key	Move (Translate) Down
'Left' Key	Move (Translate) Left
'Right' Key	Move (Translate) Right
'Page Up' Key	Accelerate
'Page Down' Key	Decelerate
'A' Key	Turn Left
'S' Key	Pitch Down
'D' Key	Turn Right
'W' Key	Pitch Up
'Q' Key	Rotate Anti-clockwise
'E' Key	Rotate Clockwise

The reason why the mouse and the keyboard were chosen as our reference interfaces is because, firstly our user base is familiar with these interfaces and do not require extensive training on their usage. Secondly, these interfaces are quite prevalent in applications where 3D navigation, exploration and manipulation activities are required.

6.2. EXPERIMENT 1 DESIGN AND RESULTS

This experiment was designed specifically to gauge the required learning curve for the users adapting to the device while varying the difficulty between various actions executable in a 3D dataset. It was also used to gather data to draw a comparison between the two designed IMUs according to their controllability and other performance ratings. For this purpose a user study comprising of two user groups, one of six individuals for Version 1.0 (GF-IMU) and the other of ten individuals for Version 2.0 (Gyro-IMU) was performed. The user profile comprised mainly of students and research staff from various fields of engineering and sciences including bioengineering, geophysics and mechanical engineering (4 males and 2 females for Version 1 testing and 7 males and 3 females for version 2 testing). No users were recalled in the Gyro IMU based version 2.0 testing from the Gyro-Free IMU based version 1.0 user group. Conventional interfaces like the keyboard and the mouse were not tested in this experiment because the users are already quite familiar with their usage and thus the data gathered will not give the correct duration for adapting to the device.

6.2.1. Design of Experiment 1

Each user, with only a rudimentary demonstration of the device was asked to perform three tasks. Each task was to be performed three times and the time taken to complete each task was logged. No prior handling or training was

provided before the start of the experiments so that performance improvement from practice can be observed. The metric for the completion of the task is 85% overlap between object and target outline (See Figure 6.3 and 6.4). However, before the second and the third attempts at the experiments, the users were allowed to train on the device for short durations of about 10-15 minutes. As the tasks are to be performed with both the variants of the IMVU i.e., the Gyro-free IMVU and the Gyro-based IMVU), the user groups of this study were separate and none of the users in a particular user group were allowed to handle the other variant. The three tasks are described as follows:

- 1) Task 1: centering and alignment of a displaced (on x -axis) and misaligned (rotated on the z -axis) letter ' **E** ', see Figure 6.3.
- 2) Task 2: centering and alignment of a displaced and misaligned letter ' **E** ' while following a step-by-step path. The letter ' **E** ' is displaced on the x and z axes and rotated on the y -axis, see Figure 6.4A.
- 3) Task 3: alignment of the letter ' **E** ' when it is rotated on all three axes, see Figure 6.4B.

In each case, the user has to align and match the size of the object (letter '**E**') to a reference outline (shown in a black outline in Figure 6.3 and 6.4) and hold the view for about 2 seconds. The task is not deemed completed if the object is on target during a transient pass by. Each task tests a particular subset of interaction design for different type of motion gestures about different spatial axes. The tasks were deliberately kept simple enough in order to gauge the

average learning curve required by the users when introduced to the device for the first time.

The IMVU variants were not compared with the legacy devices in this experiment as the users had ample prior experience on such devices. As the objective of this experiment is to gauge the effects of training on the user's ability to use the interfaces effectively, such an endeavor would be irrelevant. Also the tasks are ordered in increasing levels of number of axes of interaction, which we assume would result in a gradual increase in the difficulty of the tasks.

The participants were also asked to fill out a questionnaire that evaluates and quantifies their experience during the user study. The questionnaire was designed to estimate the five major key performance indexes that are:

- 1) Ease of use,
- 2) Respond intuitively to intended action and
- 3) Effects of training on performance.
- 4) The controllability of the device. (Whether the device felt too hard or too fluid to control i.e. too sensitive or too unresponsive.)

Figure 6.3 Screen shot of the task 1 specifically designed to manipulate multiple axes of freedom, namely x translation and z rotation. The users have to match the black outlined letter 'E' with the yellow 'E'.

A rating of 0-10 was requested from the user, with a value of 0 being most negative and the value of 10 being most positive.

Figure 6.4 Screen shots of (A) task 2, which is specifically designed to manipulate multiple axes of freedom, namely x translation, z translation and x rotation, and (B) task 3, which is specifically designed to manipulate all rotational axes, namely x rotation, y rotation and z rotation. The users have to match the black outlined letter 'E' with the yellow 'E'.

6.2.2. Results of Experiment 1

The results obtained for each of the designed tasks 1 through 3 are shown in

Figures 6.5-6.7. Task 1 is considered the easiest one to perform. Tasks 2 and 3 are considered of comparable difficulty. The Gyro-IMU and GF-IMU were both tested with the interaction design in place.

Figure 6.5 shows that all the users make significant improvements in their performance in manipulating the IMVU, after having a few minutes of practice on the device using Version 1.0 (GF-IMU) or Version 2.0 (Gyro-IMU). This can be seen in the sharp drop in the average time taken to complete the task 1 from attempt 1, which is with no training, compared to attempts 2 and 3, which are the two attempts after spending some time trying the device out. However the Gyro based IMU shows a much lower mean; showing that the users were able to complete the same task quicker on the Gyro based Version 2.0 than on the Version 1.0. This is because of the increased sensitivity and responsiveness for the Gyro-IMU over the GF-IMU. The reason why an increase in sensitivity and responsiveness reduces the time taken to complete the task, is because the user is not only able to interact and move on a more fine level, but also quickly correct mistakes like overshoots and unintended motions. This fact is further supported by the data gathered from the survey questionnaires (See Table 6.2). Since no users were recalled in the version 2.0 (Gyro-IMU based IMVU) testing from the previous version 1.0 test group, the data is free of any effects that may perturb it due to the users being accustomed to device interaction.

Figure 6.5 The graph shows the average time taken (in milliseconds) and its standard deviations for completing task 1 versus the attempt number. The data shown is the average of users in a particular user group in the user study for each of the 3 attempts. The first attempt is done without training and the second and third attempts are done after some training. The plots for the GF-IMU version and the Gyro-IMU version show that the Gyro based version is much more easily controllable and thus with a significantly less learning curve.

It was observed that the standard deviations among the six users were rather high for the GF-IMU based interface (Version 1.0), suggesting that some people are better at using inertial or motion-based interactive devices than others. However, with some exposure to the GF-IMU, the variability in proficiency among the different users decreased. The Gyro-IMU however, not only has a lower mean but also much lower standard deviation, which suggests that this version is easy for most users to perform the required task with. The users are able to complete the tasks in fairly similar time durations (low standard deviations) suggesting that the users are able to quickly familiarize themselves with the device and use it.

Although the short training period given to the users during the study did not significantly change the time taken by the users to complete each task using the Gyro-IMU based IMVU, but a small decreasing trend was noted. This suggests that the efficiency for the completion of the task increases with training, but not with a significant margin. The effects of training are much more evident in the Version 1.0 GF-IMU plot than in the one for Gyro-based IMU because the slight lag and higher thresholds increase the opportunity for the user to learn how to anticipate and accommodate for such intrinsic behavior. The Gyro-based IMU is already more sensitive and therefore sees less improvement in performance with practice.

Tasks 1, 2 and 3 differ from each other as they measure controllability (which is defined as the inverse of sensitivity in this case) and required learning curve for specific axes of interaction. The reason for keeping these tasks as separate was to keep the tasks simple. This would keep the difficulty of the task from influencing the learning curve for the interface. Although these tasks were kept easy, their relative difficulties varied from task to task as the number of axes of interaction tested in each task is different as discussed previously in Section 6.1.1. The Task 1 is the simplest of all, testing only two axes of interaction, i.e. one translational axis and one rotational axis. Figure 6.5 shows a reduction in the time taken by the users with fairly low standard deviations compared to those of Tasks 2 and 3 for either versions of the IMVU. Notice the difference between the data points for the Gyro based IMU and the Gyro Free IMU. This difference is due to the better sensitivity of Gyro-IMU over GF-IMU.

Figure 6.6 Average time taken (in milliseconds) and its standard deviations for performing task 2 versus the attempt number.

However, Tasks 2 and 3 are slightly more difficult because they test three axes of interaction i.e. two translational axes and one rotational axis for Task 2 and three rotational axes for Task 3. This perturbs the results slightly for both versions. As seen in Figure 6.6 for Task 2, the GF-IMU based version 1.0 has larger standard deviations than for Task 1. Similarly the Task 3's standard deviations are even larger. The version 2.0 (The Gyro-IMU based interface) plots for tasks 2 and 3 however show more consistent standard deviations. This difference in the data collected for the two interfaces can again be attributed to the marked difference in sensitivity. The lower sensitivity and responsiveness of the GF-IMU introduces a considerable learning curve, which allows some users to perform much better than others. The Gyro-IMU on the other hand allows users to learn quickly and perform consistently due to its higher responsiveness.

Figure 6.7 Average time taken (in milliseconds) and its standard deviations for performing task 3 versus the attempt number.

The slope of the plot for the Gyro based IMVU Version 2.0 for task 3 is of interest, as both the slope of the curve and the standard deviations are quite small. This is because of quicker responsiveness for the Version 2.0 especially in rotational motion (See Section 4.1) than the version 1.0, as aptly described by a user comment quoted as under:

“The previous device (version 1.0) had a significant lag in its rotations which made me over shoot the target quite a lot. I find this version (version 2.0) more responsive”

The experimental results (for Task 3 which focuses on Multi-axis rotational stimulus from the user) suggest that the improvement in rotational responsiveness in Version 2.0 over Version 1.0 is substantial enough to increase

the ease of use significantly while reducing the learning curve. Together with the intuitive nature and visual feedback coming from the display screen, it enhances the users' ability to correct mistakes and compensate quickly enough for the rotational motions. This is supported by the following user's quote.

“The rotations were quite confusing in the previous version because of a delay, but this one (version 2.0) has much better control.”

In summary, although, Gyro-IMU shows better performance figures from the GF-IMU figures, technically there are some advantages to using GF-IMU, which will be discussed in the next chapter. It is encouraging to observe that given some training in using a novel interactive visualization devices like the proposed IMVU, the majority of users are able to improve their performance when navigating in a 3D visual environment and controlling the virtual 3D camera to a desired orientation and position in the 3D visual space. This suggests that such devices have the potential to be used as effective interactive device for exploratory visualization albeit some measure of acclimatization is required on the users' behalf.

Table 6.2 shows the summary of replies to the questionnaires collected from our user group. User feedback validates our deductions from the performance figures. Note that The Gyro-IMU scores higher in Controllability and Ease of use categories compared to GF-IMU, whose scores are only marginally good in those categories. The most positive feedback was that the visualization device responded intuitively to intended action of the user. This characteristic is important as it speaks well of the fundamental concept driving the design of the

proposed inertial measurement visualization unit. The user feedback scores for intuitiveness are almost equal (only a negligible difference of 0.25), suggesting that both the GF-IMU and the Gyro-IMU, when applied to the task of 3D visual pose control in virtual space were observed to be equally intuitive. In essence, with the IMVU, the user's action match well with the resulting motion of the virtual object in the visual space and this strongly influences the Stimulus-Response (S-R) compatibility rating of the device [45].

Table 6.2 The table shows average values of various parameters gauged in our questionnaire. The device scores highest for controllability and intuitive design.

	Ease of use	Effect of training	Intuitiveness	Controllability
GF-IMU	6.25	6	7.25	6.75
Gyro-IMU	7.5	3.6	7	8

However, the effect of training in improving overall performance is lower for the Gyro-IMU version as depicted by Figures 6.5 through 6.7 suggesting that the initial control is already good enough such that prolong use gave only marginal improvement in perceived performance gain. The training only allows the user to familiarize and get comfortable with the various thresholds set in the interaction design. Therefore the user feedback shown in Table 6.2 validates our initial assessment of the experiment results that the Gyro-based IMU performs better for the particular interaction design than the GF-IMU based version.

6.3. EXPERIMENT 2 DESIGN AND RESULTS

This experiment was designed to measure the speed and controllability advantage gained by the IMVU due to high intuitiveness as compared to other popular interfaces. However, this experiment was only performed on the Version 2.0 (Gyro-based IMU) to measure the difference in the completion of the assigned task with the optimal responsiveness and sensitivity. This was deduced by comparing the results for the Experiment 1 for either IMU based input devices (See Section 6.2.1). As in Section 6.2.1, the results for Experiment 1 have clearly shown that the time taken by the users to complete the tasks for the Version 2.0 (Gyro based IMU) was significantly better than that of Version 1.0 which can be attributed to improved responsiveness.

6.3.1. Design of Experiment 2

This small user study comprising of three adept users in 3-dimensional maneuvering was also conducted to gauge the comparative intuitiveness for each of these interfaces. The user count was kept low on purpose because it employed only the most proficient users from our various user groups. These users had shown good reaction time to visual indicators due to their gaming background. This experiment uses a very narrow and small tunnel track shown in Figure 6.8. The track was designed to differentiate the IMVU from the mouse-only and keyboard-only interface on its level of intuitiveness. The track is composed of two turns designed to highlight the deficiencies, especially the unintuitive multi-axial interaction, of the mouse and keyboard interfaces. It also

distinguishes the IMVU on its advantage of intuitive multi-axial input.

The first turn is a sharp turn pitching down designed specifically to be overshoot by the user if executed too early or too late, or when their reaction time along the turn is slow. The second turn is a gradual turn which twists around the z-axis (World/inertial axis rather than the body/view axis as shown in Figure 6.8), so the user has to constantly course correct their turn at different rates in this narrower track.

This experiment was performed thrice per user per interface at three fixed speed levels of 5, 10 and 15 and the percentage of the path covered along the specific turn was logged. The results will be presented in Section 6.3.2. For this specific experiment, the IMVU is broken down, separating the display and the input devices. Also the interaction design is turned off completely. This is done so that the user can control the precise angle over all 360 degrees rather than the limited angle range implemented by the interaction design that was discussed in Section 5.3.

Figure 6.8 The above wireframe view shows experiment 2 tunnel track. The first and second turn are designed specifically to test responsiveness the user gains due to the intuitiveness of each interface.

6.3.2. Results of Experiment 2

The object of this experiment was to show the advantage of the IMVU over traditional interfaces. The major advantage is that the IMVU provides the user with simultaneous control of multiple axes using a physical gesture mapping that replicates the motion trajectory in virtual space unlike the keyboard-only and mouse-only interfaces.

Table 6.3 The table shows the results gathered for Experiment 2 for the first turn. Notice the mouse-only and keyboard-only entries for speed 15.

Users	Percentage converted before overshoot (First turn)								
	IMVU			Mouse-Only			Keyboard-Only		
	Speed 5	Speed 10	Speed 15	Speed 5	Speed 10	Speed 15	Speed 5	Speed 10	Speed 15
1	100%	100%	78%	100%	100%	62%	100%	100%	58%
2	100%	100%	100%	100%	100%	78%	100%	85%	62%
3	100%	100%	100%	100%	100%	62%	100%	100%	100%

Table 6.4 The table depicts the results for the Experiment 2 for the second turn. The entries for the Keyboard-only and mouse-only interfaces show the lack of controllability of these interfaces due to the unintuitive design for 3D applications.

Users	Percentage converted before overshoot (Second turn)								
	IMVU			Mouse-Only			Keyboard-Only		
	Speed 5	Speed 10	Speed 15	Speed 5	Speed 10	Speed 15	Speed 5	Speed 10	Speed 15
1	100%	100%	100%	100%	100%	100%	100%	100%	34%
2	100%	100%	100%	100%	100%	75%	100%	34%	75%
3	100%	100%	100%	100%	100%	34%	100%	58%	75%

The results for the first turn show that the IMVU allows the user to make a quick sharp turn. This is because of the intuitive nature of the interface, which allows the user to turn to the precise angle by exactly rotating the device to that actual angle rather than move the device over and over like in the case of a mouse. Also if the user intends to make a quicker turn, they only need to turn quicker to that angle rather than at a preset angular rate as is the case with the keyboard.

At higher travel speeds, where the user needs to perform more rapid steering maneuvers, the limitations of the traditional interfaces become apparent. This fact is evident from the entries at Speed 15 in Table 6.3 and 6.4 for all interfaces. For the case of the keyboard-only interface for the second turn, the reason for the somewhat similar points of overshoots was noted by one of our users. The fact that, for arbitrarily angled turns, one has to turn about the z-axis (View space in the second turn) to put the curve of the turn about the horizontal (or vertical) and then pitch up or down (or yaw right or left). Another approach is to make small turns about the x and y-axis (Pitch and Yaw) in a

proportionally multiplexed manner to simulate the turn to that angle. Both of these approaches make the keyboard very cumbersome as an input device for specifying rapid 3D maneuvers.

The mouse-only interface however is not prone to this problem. But it is still prone to a backtracking problem especially for the rotation shown in Figure 6.9, where the user has to move the mouse to a position where the user cannot move the physical mouse device further due to the borders of the track pad or other obstacles. In such a case the user has to backtrack the device to a cleared position and then execute the motion again to reach the intended angle. This backtracking action slows down the mouse-only interface significantly. The mapping of the amount of mouse's physical movement to the rate of turning in the virtual world is a fixed quantity determined by the program. Although in some application the user can control this sensitivity on the fly, but it still adds complexity to the interface, and diverts users attention to controlling the interface rather than performing the task.

Figure 6.9 The mouse backtracking problem is shown step-by-step. (1) The user tries to execute a large motion and (2) runs out of space on the track pad. Thus (3) the user lifts the mouse and brings the mouse back to a convenient position to continue the motion.

6.4. EXPERIMENT 3 DESIGN AND RESULTS

The user has to follow a tunnel-like path with various twists and turns (See Figure 6.10). The user also has to negotiate some obstacles along the path, such as narrow constricted egress, sharp turns and S-bends (See Figure 6.11A, B and C). The application mimics visualization of endoscopy performed on a patient as well as a tunnel racing video game like Descent Series. The goal of this experiment is to compare the overall performance of the user while using different interfaces to complete a time sensitive task. Figure 6.12 shows the user's view in the tunnel.

6.4.1. Design of Experiment 3

This experiment is a scored experiment, where the scoring is proportional to the ratio of time spent by the user within the tunnel and total time consumed by the user to get to the other end of the tunnel, the average traversal speed used by the user. Points are deducted if the user moves out of the designated path/tunnel.

Figure 6.10 A view of the tunnel like path the user has to follow in Experiment 2. The user is supposed to stay inside the tunnel. If the user moves beyond the walls of the tunnel, the scoring stops and the user is supposed to move back inside the walls as soon as possible.

A

B

C

Figure 6.11 Some of the obstacles the user has to negotiate while performing in the Experiment 2. A) Shows a constricted path the user has to enter. B) Shows an S-bend turn the user has to proceed along. C) Shows a sharp turn. These obstacles are designed such that the user is forced to reduce his or her speed and at times use one of the translation axes along with the rotational axes.

Figure 6.12 A screen shot of the Experiment 2 showing the lit tunnel from the users' perspective.

The score is calculated using the following equation.

$$Score = \frac{V_{AV}}{20} \times \left(\frac{T_i}{T_t} \right) \times (30000 - T_t)$$

Eq 6.1

Where, V_{AV} is the average speed at which the user completed the experiment, the speed is measured by the number of units the user traverses in each iteration and ranges from 1 to 20. T_i is the time spent inside the walls of the tunnel path. T_t is the total time taken to finish the task.

Once the task was complete, the user had to perform the same task with a mouse-only interface and a keyboard-only interface. The mouse-only interface is similar to many 3D modeling applications. The only difference is that most 3D modeling applications have some actions mapped on the keyboard making them a combined mouse and keyboard interface. However we will be treating the mouse as a stand-alone interface and thus those actions will be mapped onto the mouse along with the already present pointing/navigation actions. Similarly, the keyboard-only interface will map all the navigation actions strictly onto the keyboard. This will allow a fair and simpler comparison between the three interfaces.

6.4.2. Results of Experiment 3

The objective of this experiment was to measure the device performance when the user is required to perform with a time sensitive task, where the user will be

executing fast commands over much shorter intervals. In such a case, the responsiveness of the device would become a critical feature. The data collected will allow a qualitative analysis on how well the users can adapt to the device and perform a demanding task in a fairly short time. Also with the data from the same experiment performed on a mouse-only and a keyboard-only interface, we will be able to see how our interface compares to traditional devices with which the users have ample familiarity and exposure. The following tables and figures summarize the data collected during our experiments.

Table 6.5 The table shows the division of the scores collected from the user group for the experiment

User distribution for interfaces			
Score Level	Number of users in each category		
	IMVU	Mouse-only	Keyboard-only
>12000	1	3	4
11999 to 900	7	6	7
899 to 600	5	5	5
599 to 300	2	2	0
299 to 0	1	0	0

The results show that the IMVU (with the Gyro-IMU) can be used in a time-sensitive maneuver-oriented application with comparable overall performance as other interfaces such as the mouse and the keyboard. The keyboard and the mouse provide very fine control, which in its current state, the IMVU does not. However the IMVU provides a more direct mapping of the user's applied motion into the interaction (view motion) in 3D virtual world, making it more suited for applications where users have to execute a relatively complex trajectory quickly in real-time. The reason for IMVU's comparatively

inaccurate output for position as compared to other interfaces like the mouse is its dependency on inertial sensors. While these sensors make the IMVU portable and self-contained, it reduces the device's accuracy. MEMS-based Inertial sensors are small and power efficient but prone to noise like any other sensor. The real source of the inaccuracy is the nature of the algorithm, which integrates this noise while calculating position thus perturbing the position output. However this can be reduced with the help of signal processing.

Figure 6.13 Data in table 6.5 plotted as a probability distribution of the various users over the score classes for each interface. Its helps in showing how close each interface perform in a time sensitive application.

The results of this experiment however show that there is a room for improvement in the responsiveness of the device since its average and high scores are lower for mouse and keyboard based interfaces. But this is only due to the fact that mouse and keyboard have become the ubiquitous computer input devices, and thus virtually all users have exposure to them. The fact that the

IMVU has roughly the same user distribution as the keyboard and mouse interfaces, is a high note for its performance as shown in Figure 6.13. Its main objective was to distinguish between these three interfaces. Note that the moderately high performing attempts lie in the same band (See Figure 6.13) for all three interfaces, which shows comparable performance figures.

Table 6.1 shows the complexity of the keyboard-only interface, which requires a total of 12 keys mapped for each navigation action. The complexity of the keyboard interface is accentuated by the fact that so many of the keys that are not used in the interface are situated very close to the mapped keys. In a time-sensitive and intense application like 3D games where the user has to perform various actions in a very short duration, these unmapped keys get in the way of the interface.

Similarly, the mouse-only interface requires the use of a button to multiplex the various actions that is very unintuitive for the user, who can easily get confused and forget to press the multiplexing button. Also the use of the scroll wheel as a rotation control is quite unintuitive, as the user has a hard time adapting to the direction of the anticlockwise and clockwise direction of the rotation. This is because of the placement of the scroll wheel perpendicular to the view of the roll rotation on the screen.

The Experiment 2, as the results in Section 6.3.2 show, differentiates the three interfaces and quantifies the intuitiveness of the IMVU. The Experiment 3

quantified the performance of the interface on the overall basis, where the minute details and corrections by the user make only very little difference. These differences get lost when the data collected is looked at a macroscopic level, where users from diverse backgrounds are present and only the overall performance is under review. However, with the smaller more demanding track built for Experiment 2 and a select user group of three avid video game players, these minute differences become evident.

CHAPTER 7

DESIGN CHALLENGES

The use of inertial sensors for motion sensing makes such device highly portable and low cost but has many technical challenges that tend to reduce the device performance. These shall be discussed in this chapter.

7.1. CHALLENGES WHILE DESIGNING GF-IMU BASED INERTACTIVE DEVICE

The sensor signals integration process used in the 3 translational axes and the Z-axis rotation makes it difficult to provide precisely controlled motion-based interactions. The slight misalignment of the four sensors during physical attachment to the device can lead to ambiguous sensor signal interpretations. The misalignment will result in the sensor pair on the moment arm taking slightly different routes to the desired location. As a result, non-linearity in the sensor's output signal can cause unwanted spikes to be generated during signal subtraction and this can lead to incorrect interpretation of motion gestures. This problem can be alleviated with improved precision in the mechanical assembly and more sophisticated sensor signal matching strategies during the calibration process.

Due to the low frequency of the sensor signals, at times these may become slightly out-of-sync due to noise, especially when very slow gestures are applied. This causes the signals to be misinterpreted especially if an algorithm depends on their interdependence to interpret them. The IMVU algorithm was thus designed in such a way that it adapts to these cases of out-of-sync signals, and also has reduced signal interdependency. However this does not remedy the problem completely and thus is left for further research.

7.2. CHALLENGES WHILE DESIGNING GYRO-IMU BASED INERTACTIVE DEVICE

Gyro based IMU are mechanically quite simple to construct. However, due to the strap-down nature of the design, the orientation of the sensors is fixed to the body frame rather than the inertial frame. This has two significant effects on the algorithm, first that the orientation of the previous state should be known in order to calculate the orientation of the new state. This affects the calculation of the gravity vector. Secondly, it requires the algorithm to find the inertial frame quantities of their known body frame counterparts such as angular velocity and total acceleration, which in turn is dependent on the knowledge of the gravity vector and/or the magnetic north vector.

Figure 7.1 In (A) the total acceleration is equal to the gravity vector, thus the orientation about the gravity vector can be estimated. However in (B) The total acceleration is not equal to the gravity vector, which is the case when the object is accelerating. In this case the gravity vector is ambiguous, and can only be estimated with the help of angular velocity (which is a noisy signal). Thus the orientation calculation is not entirely dependent on the gravity vector alone.

Both of these are very interdependent problems and become significant because of the fact that during the executing of a rotational motion, the signals from both sensors are affected, i.e. angular velocity from the gyroscope and the total acceleration, which includes gravitational acceleration, from the accelerometers. Thus in order to find the gravity vector embedded in the total acceleration vector (from the accelerometers), one needs to know the correct orientation (See Figure 7.1B). On the contrary, one may depend on the measurement of the gravity vector (from the total acceleration measured by the accelerometers) to calculate the correct orientation (See Figure 7.1A).

These challenges are further complicated by the fact that the angular velocity measured by the gyroscopes is a highly dynamic signal, i.e. the signal is valid only when there is angular motion, caused by the the sensor's rotation. The accelerometers on the other hand have a very static signal, which settles slowly as far as the measurement of the gravity vector is concerned. This may cause

the two signals to go out-of-sync especially when slow rotations are applied.

All these issues complicate the design of the interface, one solution is to design with a focus on one particular type of motion that is essential to the application while ignoring the other motions afforded by the interface. This fact is particularly evident in the new motion sensing interfaces rolling out these days, such as the Sony's PlayStation™ Move, the Apple's iPhone 4 and SixSense's controller (which is only a prototype during the writing of this document).

7.3. DISCUSSION ON THE PERFORMANCE OF GYRO-IMU AND GF-IMU

Experimental observations of the GF-IMU and Gyro-IMU show that the Gyro-IMU is better than the GF-IMU. This is because the Gyro-IMU measures angular motion directly with the use of gyroscopes, while the GF-IMU uses accelerometers placed about a moment arm to measure angular velocity. Thus, GF-IMU requires quite precise alignment of the sensors as well as an optimum length of the moment arm, both of which cannot be achieved in our hand built prototype. Thus our prototype and the results gathered should not conclude that a GF-IMU performance can not approach that of a Gyro-based IMU, much more improvement can be made. The GF-IMU was more of a proof-of-concept of an accelerometer-only based six degree-of-freedom interface rather than a fully functional interface. Such an interface is novel in its own design and construction.

The GF-IMU has some advantages over the Gyro based IMU, as far as the design and development is concerned. Firstly, since it uses only accelerometers and thus its sensors are balanced. It suffers much less from out-of-sync problems than the Gyro-based IMU. Since there are many more accelerometers along each axis, the acceleration measurement is much more accurate using Kalman estimators than is possible in a Gyro-based IMU. Finally, the MEMS construction of the inertial sensors is such that the accelerometers are much more robust than the Gyroscopes [38], making the GF-IMU more reliable.

The Gyro-based IMU has shown an acceptable performance. The users have found the Gyro-IMU based interface more intuitive and responsive than the GF-IMU based interface. The main reason being that the Gyro-based IMU measures angular velocity directly with more sensitivity thus requiring much less signal processing than a comparable GF-IMU. It also employs half the number of sensors thus reducing the cost for manufacture. The cost saving may be offset by the fact that gyroscopes are more expensive than a comparable accelerometers.

CHAPTER 8

CONCLUSION AND FUTURE WORK

This document discussed issues related to the implementation of a novel interactive display unit for performing exploratory visualization and navigation of 3D datasets and virtual environments. We described the hardware design and signal-processing algorithms related to the proposed inertial sensing system and motivated the design of the accompanying visual interaction strategies that took into account the characteristics of this input modality. Results from the experimental studies suggest that the use of such motion-based device is generally intuitive, while providing a real world 3D interaction method for the user. Furthermore, the user's performance of various visual navigation tasks improves with limited exposure and training on the novel device.

8.1. CONTRIBUTIONS

The major contributions demonstrated in this thesis are as follows:

- Design and implementation of two configurations of 6-DOF IMU. A single configuration of GF-IMU was design with a few modifications for planar Human Input Devices. Another configuration of Gyro-based IMU was designed as a generic IMU for position measurement. These two prototypes were optimized for human interaction.
- The software and algorithms, as well as the signal conditioning

operations were designed to estimate 3D position and orientation information from the inertial sensing system. This algorithm was designed for small embedded system (with floating point unit) so that such a system can be targeted to the Human Input Devices (HID), which include gaming devices, portable devices, etc with a visual feedback from the embedded display unit.

- Interaction design for the motion gesture based exploration of simple 3D visual data was implemented. It was designed to allow the user to view and browse through 3D datasets intuitively.
- A comprehensive user study was conducted for the position tracking module in conjunction with the interaction design. The user study results showed that, of the two IMU configurations tested during the study, the Gyro-based IMU showed more promising performance parameters than the GF-IMU. Also, the user study showed that the IMVU gave a more intuitive means to manipulate the 3D view-space than the commonly used, conventional human input devices, especially when the input gestures require immediacy in the visual responses of the 3D view-space.
- Testing and researching potential applications for the IMVU was done. This is shown in the next section.

We have seen a recent resurgence into the 6-DOF based motion tracking interfaces such as Nintendo Wii-remote, Apple iPhone 4, Sixsense controller for PC, Sony Playstation Move, etc. However, with the exception of the Apple iPhone 4, all these methods of position tracking are not portable in nature. They

require setting up and space to operate, which includes the host module and base module (for example the Wii-remote requires its IR tracking base station and Nintendo game console to work; the Playstation Move also requires a similar setup). Also, they are not scalable to portable platforms which do not operate around any fixed base station.

Our IMVU provides a human input mechanism that is motion based and intuitive, while not being dependent on a particular operating area which is under the influence of a base station, i.e., it is truly portable. Although the Apple iPhone 4 has an accelerometer and a gyroscope (the iPhone 4 was not released during the course of this research), it does not have the software algorithm necessary to estimate position in its development framework or operating system. It is our belief that the algorithmic design of the IMVU can be ported to the Apple iPhone 4 without significant change and is not relevant as a research endeavor.

8.2. POTENTIAL APPLICATIONS

The video game industry has seen significant new developments in the interface/controller reinvention. The industry is making a conscious effort to redesign the legacy game controllers to smart motion sensitive controllers [39, 40, 41, 42, 43]. Thus the obvious applications for our IMVU are video games.

Other potential applications are scientific and medical visualization

applications. With the development of 3D MRI, CT and PET scans and also of newer procedures which allows doctors to treat parts of the body untreatable before due to the lack of appropriate information and visual aids, a new interface is necessary for these doctors to visualize these parts of the body and train themselves.

Another interesting application explored by us during the course of this project is the remote motion control of many-degrees-of-freedom robots [44]. With the development of small UAVs and ground based robots for various exploratory and task oriented applications, the legacy interfaces of the joystick, switches and buttons are becoming increasingly cumbersome. Our interface can provide a viable alternative to these legacy interfaces.

Figures 8.1 and 8.2 show the screen shots of an application demonstration for our interface. It is a free roam 3D application in which the user can navigate a room. The application also provides other models to be loaded and viewed instead as well. The user is given full control over all axes i.e. six degrees-of-freedom. The user also has the option of turning on and off the interaction design completely. The users were only asked to fill out a questionnaire after using the application.

Figure 8.1 A screen shot of the demo showing a simple 3D space with various assorted objects. The user can roam about freely and explore it while getting used to the interface.

Figure 8.2 A screen shot of the demo from a different angle.

8.3. FUTURE WORK

The work to date has proved that a viable visualization unit, allowing 6 degrees-of-freedom to the user can be built using inertial sensors. However the current performance of the IMVU has ample room for improvement. The placement strategies of sensors for Gyro-Free IMU in user interface applications will improve the performance and is therefore a worthwhile research endeavor. Similarly, algorithms optimized specifically for interactive application, which are robust to sensor misalignment issues, should be researched.

Better real-time filtering techniques to make the interface more responsive also need to be explored. Such techniques will allow for the reduction of the various noise thresholds in the current algorithm. Furthermore, with the development of new 3D applications and corresponding hardware such as GPU, Physics engines, etc, an interesting research endeavor would be to couple our device with such hardware for visualization and manipulation applications to measure the performance of the interface when coupled with fast, performance intensive applications.

A comparative study with other interfaces will also be an interesting research topic, which gauges the advantages and deficiencies of various interfaces. Also, control applications for robots while using acceleration hardware in the loop is a research interest worth pursuing, as it would significantly improve on the intuitiveness of the existing Human Machine Interfaces (HMI).

In short, our interface has shown to be an alternative interface for various scientific, commercial and industrial applications, which has potential for further research as well.

REFERENCES

- [1] Grewal, M. S., Weill, L. R. & Andrews. A. P. (2nd Ed) (2007). *Global Positioning Systems, Inertial Navigation and Integration*. New Jersey Wiley-Interscience.

- [2] Bekir, E. (2007). *Introduction to Modern Navigation Systems*. New Jersey World Scientific.

- [3] Groves, P. D. (2007) *Principles of GNSS, Inertial and Multi-sensor Integrated Navigation Systems*. Boston. Artech House.

- [4] Schuler, A. R., Grammatikos, A. and Fegley, K. A. (1967). Measuring Rotational Motion with Linear Accelerometers. *IEEE Transactions on AES*, 3(3), 465-472.

- [5] Brown, R. G. & Hwang, P. Y. C. (3rd Ed) (1997). *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley & Sons.

- [6] DiNapoli, L. D. (1965). *The Measurement of Angular Velocities without the use of Gyros*. Philadelphia. The Moore School of Electrical Engineering, University of Pennsylvania.

- [7] Edwan, E., Knedlik, S. & Loffeld, O. (2008). An Extended Kalman Filter for improving angular motion knowledge in a multiple distributed IMU set. In: *Symposium of Gyro Technology, Karlsruhe*.
- [8] Maybeck, D. S. (1979) *Stochastic Models, Estimation & Control, Vol 1*. New York. Academic Press.
- [9] Luinge, H. J. & Veltink, P. H. (2003). Inclination Measurement of Human Movement using a 3-D Accelerometer with Auto-calibration. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. 12(1), 112-121.
- [10] Liu, H. & Pang, G. (2001). Accelerometer for Mobile Robot Positioning. *IEEE Transactions on Industry Applications*. 37(3) 812-819.
- [11] Kim, A., Peng, Y. K., & Golnaraghi, M. F. (2004). A Sensor Fusion Algorithm for GPS, Strap-down and Gyro-Free Inertial Navigation Systems. In: *Proceedings of CSME Forum 2004*.
- [12] Vaseghi, S. V. (4th Ed) (2009). *Advanced Digital Signal Processing and Noise Reduction*. Chichester, U. K. John Wiley & Sons.
- [13] Ayob, K. (2008). *Digital Filters in Hardware: A Practical Guide for Firmware Engineers*. Victoria, BC, Trafford.

- [14] Chitode, J. S. (2nd Ed) (2007). *Digital Signal Processing*. Pune, Technical Publications.
- [15] Wang, Q., Ding, M., & Zhao, P. (2003) A New Scheme of Non-gyro Inertial Measurement Unit for Estimating Angular Velocity. In: *29th Annual Conference of the IEEE Industrial Electronics Society, IECON 2003*. 2, 1564-1567.
- [16] Chin, T.-L. (2008). Design and Analysis of a Fault Tolerant Coplanar Gyro-Free Inertial Measurement Unit. *Journal of Micro-Electromechanical Systems*, 17(1), 201-212.
- [17] Edwan, E., Knedlik, S., Zhou, J., and Loffeld, O. (2009) GPS/INS Inegration for GF-IMU of Twelve Mono-axial Accelerometers Configurations. In: *6th Workshop on Positioning, Navigation and Communication, 2009. WPNC'09, Hannover*. 183-188.
- [18] Jintao, Y. (2008). A Novel Design for NGIMU. In: *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing, Shanghai*. 267-269.
- [19] Bang, W.-C., Chang, W., Kang, K.-H. Choi, E.-S. Potamin, A., and Kim, D.-Y. (2003). Self-contained Spatial Input Device for Wearable Computers. In: *Proceedings of the Seventh IEEE International Symposium on Wearable Computers ISWC '03*. 26-34.

- [20] Tokoro, Y., Terada, T., and Tsukamoto, M. (2009). A Pointing Method Using Two Accelerometers for Wearable Computing. In *SAC '09, Honolulu, Hawaii*. 136-141
- [21] Kallio, S., Kela, J. Mantyjarvi, J., and Plomp, J. (2006). Visualization of Hand Gestures for Pervasive Computing Environments. In: *AVI '06, Venezia, Italy*. 480-483.
- [22] Martens, J.-B., Qi, W., Aliakseyeu, D., Kok, A. J.-F. and Liere, R. (2004). Experiencing 3D Interactions in Virtual Reality and Augmented Reality. In *EUSAI 2004, Eindhoven, Netherlands*. 84, 25-28.
- [23] Rauterberg, M., Fjeld, M., Kruger, H., Bichsel, M., Leohard, U. and Meier, M. (1997). BUILD-IT: A Computer Vision-based Interaction Technique for a Planning Tool. In: *The video program of CHI'98, ACM*.
- [24] de la Gorce, M., Paragios, N., and Fleet, D. (2008). Model-Based Hand Tracking with Texture, Shading and Self-occlusions. In: *Proceedings of the IEEE Conference in Computer Vision and Pattern Recognition (CVPR), Anchorage, AK*. 1-8.
- [25] Fedricksson, J., Ryan, S. B., and Fjeld, M. (2008). Real-Time 3D Hand-Computer Interaction: Optimization and Complexity Reduction. In: *Proceedings of NordiCHI 2008, Lund, Sweden*. 133-141.

- [26] Cho, I.-Y., Sumwoo, J., Son, Y.-K., Oh, M.-H., and Lee, C.-H. (2007). Development of a Single 3-Axis Accelerometer Sensor Based Wearable Gesture Recognition Band. *UIC 2007, LNC*, 4611, 43-52.
- [27] Ojeda, L., and Borenstein, J. (2007). Personal Dead-reckoning System for GPS-denied Environments. In: *IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR2007), Rome*. 1-6.
- [28] Randell, C., Djialis, C., and Muller, H. (2003). Personal Position Measurement Using Dead Reckoning. In: *Proceedings of the 7th IEEE International Symposium on Wearable Computers*. 166-173.
- [29] De Marziani, C., Urena, J., Hernandez, A., Mazo, M., Jimenez, A., Alvarez, F., and Villadengos, J. M. (2008). Relative Positioning System Using Acoustic Sensors for Ubiquitous Computing Applications.
- [30] Hazas, M., Kray, C., Gellersen, H., Agbota, H., and Kortuem, G. (2005). A Relative Positioning System for Co-located Mobile Devices. In: *Proceedings of the 3rd Conference on Mobile Systems, Applications and Services, Seattle, Washington*. 177-190.
- [31] Qian, L., Chen, W.-Y., and Xu, G.-P. (2009). Application of Novel Rotation Angular Model for 3D Mouse System Based on MEMS Accelerometers. *Journal of Shanghai Jiaotong Univ. (Sci)*. 14(2), 137-142.

- [32] Oppenheim, A. V., Willsky, A. S., and Hamid, S., (2nd Ed). (1996). *Signals and Systems*. Prentice Hall.
- [33] Mahony, R., Hamel, T., and Pflimlin, J., (2005). Complementary filter design on the special orthogonal group $SO(3)$. In: *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*. 1477-1484.
- [34] Zimmermann, M., and Sulzer, W., (1991). High Bandwidth Orientation Measurement and Control Based on Complementary Filtering. In: *Proceedings of SYROCO '91*. Symposium on Robot Control, Vienna Austria.
- [35] Piro, C., and Accardo, D., (2007). A Vertical Gyro Model Based on Particle Filters. In: *Proceeding of IEEE Aerospace Conference 2007, Big Sky, MT*. 1-12.
- [36] Pascoal, A., Kaminer, I., and Oliveira, P., (2000). Navigation System Design Using Time-Varying Complementary Filters. *IEEE Transactions on Aerospace and Electronic Systems*. 36(4), 1099-1114.
- [37] Baldwin, G., Mahony, R., Trumpf, J., Hamel, T., and Chevion, T., (2007). Complementary Filter Design on the Special Euclidean Group $SE(3)$. In: *Proceedings of the European Controls Conference 2007*.

- [38] Nasiri, S., (2005). *A Critical Review of MEMS Gyroscope Technology and Commercialization Overview*. InvenSense Inc.
- [39] Apple Website Event Page. (2010, Oct 11.). *Apple WWDC '10 Key Note Address*. [On-line Video]. Available: www.apple.com/apple-events/
- [40] E3 Gamespot.com coverage. (2010, Oct 11.). *Sony E3 Press Conference*. [On-line Video]. Available: <http://e3.gamespot.com/press-conference/>
- [41] Sony Entertainment Website. (2010, Oct 11.). *Sony Playstation™ Move*. [On-line]. Available: <http://us.playstation.com/ps3/playstation-move/>
- [42] Apple's Website. (2010, Oct 11.). *Apple's iPhone 4 Specification Page*. [On-line]. Available: <http://www.apple.com/iphone/specs.html>
- [43] Sixense Website. (2010, Oct 11.). *TrueMotion® Motion Sensing Gaming Interface*. [On-line]. Available: <http://www.sixense.com/>
- [44] Hussaini, S. and Goh, W. B., (2010). A Motion-based Visual Interface for 3D Visualization and Robotic Control Applications. In: *Proceedings of the 11th International Conference on Automation, Control and Robotics (ICARCV '10) Singapore*.
- [45] van Rhijn, A., and Mulder, J.D. (2006). Spatial input device structure and bimanual object manipulation in virtual environments. In: *Proceedings of the*

ACM Symp. on Virtual Reality Software and Technology, Limassol, Cypru. 51 – 60.

[46] New York University (NYU) Website. (2011, Jan 21). Wii-remote [Online]. Available: <http://homepages.nyu.edu/~jq271/hw6.html/>

[47] Gadget Venue Website. (2011, Jan 21). Apple to investigate slow iPhone 3G iOS 4 problems. [Online]. Available: <http://www.gadgetvenue.com/apple-investigate-slow-iphone-3g-ios-4-problems-07295915/>

[48] Inhabitat Website. (2011, Jan 21). Is it green?: The Apple iPad [Online]. Available: <http://inhabitat.com/how-green-is-apples-ipad/>

[49] Game Guru Website. (2011, Jan 21). Sixaxis getting ready to rumble [Online]. Available: <http://www.gameguru.in/sony-playstation-3/2007/15/sixaxis-getting-ready-to-rumble/>

APPENDIX A

OLDER PROTOTYPES

A.1 VERSION 0.1

The objective of the experiments was not only to test the sensors and familiarize us with their usage, but also to gather “raw” data from the sensors for various characteristic motions and forces applied to these inertial sensors. To achieve this goal we required a minimalistic hardware that could sample sensor data at a good rate with minimal introduction of electrical noise and aliasing artifacts. This section would describe the hardware that interfaces to the inertial sensors, samples them and transmits the sampled data to the computer. The following section would describe the code at the computer end that receives the sent data and gathers it in a file.

Figure A.1: The sensors and interface hardware that samples and transmits the data via a USB link.

A.1.1 Sensor/Accelerometers

We are currently using Dimension Engineering's DE-ACCM3D board that employs a single Analog Devices ADXL330 chip. Thus we have three orthogonally planted accelerometers whose range is about 3g's. The DIP form factor allows for easy access to the interface pins and wire-wrapping techniques can be used to connect wires to propagate signals from the chip to the microcontroller. The chip is mounted over a soft sticky dough type material to dampen some parasitic vibrations during motion especially due to friction across surfaces. The board provides direct voltages from the sensors that can be sampled by an ADC. The accelerometers are being operated at 3.3V by an on-board regulator. This provides a ceiling for the Accelerometer values. For more details please refer to [15].

A.1.2 Microcontroller

The Microcontroller used is an Atmel ATMEGA168 AVR 8-bit microcontroller that is a prototyping-friendly chip with a 28 pin DIP packaging. The chip is mounted on an evaluation board from Arduino called Arduino Deicimilia that boasts a very accessible and quick development features due to the FTDI chip which converts the on-board serial port on the ATMEGA168 into a fully functional USB port. Thus there is no need for Driver development or installations. The Arduino IDE also quickens the development and prototyping process by providing a C like library of simple functions to control and operate the various peripherals present on the ATMEGA168 chip such as the ADC and the serial ports. An added bonus is the available sample code for various

applications. In fact, we are using a slightly modified version of such a sample in these experiments.

A.1.3 Interface

The Arduino board provides a USB interface that is easy to use. This interface can be turned into a VCP (Virtual COM Port) that can then be addressed like any common legacy serial ports. These ports support faster baud rates like 115kbps. The applications developed are simple but require fast data transfer rates. Thus 115kbps baud is used to ensure quick steady flow of data without worrying about frame mismatches.

A.1.4 Software (Remote Device)

The ATMEGA168 was programmed using the Arduino IDE and its pseudo language as it provides a simple and quick path to a working prototype without the hassle of debugging and verifying. The operation of the remote device is fairly simple, it constantly sample the three sensors via multi channel ADC and output the data in a string format through its serial port which is attached to the serial to USB chip. The ADC data is 10-bit integers from each channel that are referenced at 5V instead of the more intuitive 3.3V.

A.1.5 Software (Data Acquisition and Record, Ver. 1.0)

The Software running on the PC end is very simple Data Acquisition and

recording software. The software was developed using Microsoft Visual Studio Development environment. The program is a console application developed using C++. Although this was a quick development process, a library for Serial Interface with rudimentary yet scalable functions was written. This software displays the acquired data in textual format using the standard libraries and also records the said data on user's request in a simple text file. Graphics formatting and display of this data is then done in a separate program using MATLAB off line at the user's leisure.

A.1.6 Software (Data Acquisition and Visualization Ver. 1.1)

As our goal is to ultimately use the sensors to manipulate the user's perspective of data, it is thus prudent to monitor how this data will affect visual objects. This task is accomplished by the second version of our Data acquisition software that not only provides us with the means to view this sensor data but also a platform on which to construct our future visualization application and interface. Therefore to achieve the latter, we employed the use of 3D Hardware Acceleration Libraries such as DirectX 9/10. The program is written such that the further, more complex applications would not need to re-write the Display Hardware Interfacing code and reuse the current code while building more essential features into it. The software shows angles from the x and y-axes in visual form.

Figure A.2: The visual application for viewing of tilt data from the sensors.

A.1.7 Test Hardware

Most of the tests were conducted on the SCROBOT TM, a robotic arm that provided the six degrees of freedom to the sensors. The robot provided both programmed motions as well as manually controlled. A picture of the robot is shown in Figure A.3. The robot was used to gather sensor signals when under the influence of various translational and rotational motions.

A.2 VERSION 0.2

The new system developed is an intermediate test bed for all our future endeavors. It comprises of two accelerometers placed on a moment arm on x-axis. These accelerometers are connected to the Aduino board that simply samples the sensors at a high rate (worst case rate of 31.25 Hz, best case rate of

62.5Hz) and sends data packets to the computer. The new Serial Interface (courtesy CodeProject.com) does some error checking and passes the data to the algorithm.

Figure A.3: The robotic arm that was used to test the sensors.

A.2.1 Remote Device

The remote device has two accelerometers this time around; the two accelerometers require a calibration and matching step that was not present in the previous device. This is done in software rather than hardware. However, unlike previously, when we were transferring data at a really slow rate intentionally, we needed to transfer data faster this time around. But as monitored in our experiments, at a faster rate, the data was getting corrupted by transmission errors and framing errors. Thus an error-encoding scheme was

necessary as well with the modifications to accommodate the second accelerometer. The device is simply calculating a checksum and placing a terminator to eliminate the transmission and framing errors. However no retransmission is accommodated on packet loss. The packet is simply dropped if an error is found and the next sampled values are passed.

A.2.2 Computer-end Algorithm and Display Software

This software has major differences from its previous counterpart. The differences occur in almost every section of the design. The serial interface was changed to a more standard and fully functional code found at CodeProject.com website. The application end receiving function is capable of some rudimentary error checking. It drops the packet instead of getting it retransmitted, thus saving time from retransmissions and increasing throughput of the sampled data. This approach is used in favor of throughput because our application requires data in real-time at a constant rate. Furthermore, a very low packet drop was noted in the experiments. Therefore such an approach is applicable.

The second major change is the introduction of a simplified class for Accelerometers called CAXcel. These, not only caters for the initialization of certain parameters for the accelerometers, but also do advanced tasks such as calibration and matching of accelerometer parameters, the execution of the main algorithm for orientation-position update, etc. The unique feature is that internally, the class objects are linked together in a linked list, but externally appear to be separate entities. This allows for the algorithm to keep track of the

accelerometers, while allowing the user some level of abstraction from the system.

Figure A.4: The device with out the monitor. Notice the circular stands for the monitor.

Figure A.5: The system with monitor.

APPENDIX B

STATE SPACE DESCRIPTION

We shall start with the description of the inputs and outputs of our state space description. The input $\mathbf{u}_k$ is a column vector composed of the measured acceleration components.

$$\mathbf{u}_k = \begin{bmatrix} a_{c,x} \\ a_{c,y} \\ a_{c,z} \end{bmatrix}$$

The column vector is a 3×1 matrix composed of the components of the acceleration compensated for gravity described in Eq 3.8. The output column vector is described as follows:

$$\mathbf{y}_k = \begin{bmatrix} \vec{v} \\ \vec{p} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ p_x \\ p_y \\ p_z \end{bmatrix}$$

The output describes the position $\vec{p}$ and velocity $\vec{v}$ estimated during the current iteration to calculate the new state. The state vectors are described as follows.

$$\mathbf{x}_k = \begin{bmatrix} \vec{v} \\ \vec{p} \end{bmatrix}$$

Thus the state vector and the output vector description dictates the output transition matrix $\mathbf{C}$ be an identity matrix of the order 6×6.

$$\mathbf{C} = \mathbf{I}_6 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Where $\mathbf{I}$ describes any general identity matrix. As the state vectors are of the order 6×1 and the input vector is of the order 3×1 , a rectangular input transition matrix of the order 6×3 is required, described as follows.

$$\mathbf{B} = \begin{bmatrix} \mathbf{I}_3 \\ \mathbf{I}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where $\mathbf{I}_3$ denotes an identity matrix of the order 3×3 . Since identity matrix are assumed to be square matrix, only one 3 in the subscript is used. This matrix essentially duplicates the acceleration input vector to be added separately to the position and velocity state vector sub-components. Thus this poses as a resource/memory and operation waste when implemented. However this is necessary for any state space description, and any data structure optimization techniques may be applied to conserve this resources.

The final matrix to be described is the state transition matrix, which is slightly complicated. For its description, we shall use the Eq 3.9 and 3.10.

$$\vec{v}_k = \vec{v}_{k-1} + (t_k - t_{k-1}) \cdot \vec{a}_c$$

$$\vec{p}_k = \vec{p}_{k-1} + (t_k - t_{k-1}) \cdot \vec{v}_{k-1} + (t_k - t_{k-1})^2 \cdot \vec{a}_c$$

The above equations can be easily converted into a state space description. However a slight modification to the input transition matrix will be required as evident from the above equations. The State transition matrix is given by.

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \\ T_s \cdot \mathbf{I}_3 & \mathbf{I}_3 \end{bmatrix}$$

Where

$$T_s = \frac{1}{f_s} = t_k - t_{k-1}$$

$\mathbf{0}_3$ is the square zero matrix of the order 3×3 and T_s is the sampling time which is the reciprocal of f_s (sampling frequency). The modified input transition matrix $\mathbf{B}$ is given by.

$$\mathbf{B} = T_s \cdot \begin{bmatrix} \mathbf{I}_3 \\ \mathbf{I}_3 \end{bmatrix}$$

APPENDIX C

PUBLICATIONS ARISING FROM M.ENG RESEARCH

Please turn the page.

A Motion-based Visual Interface for 3D Visualization and Robotic Control Applications

Safiullah Hussaini

School of Computer Engineering
Nanyang Technological University
Singapore
SAFI0003@ntu.edu.sg

Wooi-Boon Goh

School of Computer Engineering
Nanyang Technological University
Singapore
aswbgo@ntu.edu.sg

Abstract—The exploratory visualization of 3D data sets or multiple axis motion control of a monitoring camera mounted on a six-axis robot often require the user to control the viewing orientation and position of a camera view within a 3D space. This paper describes the design of an inertial motion sensing system, which uses only accelerometers to provide interactive 3D control of both orientation and position in such systems. We describe the interaction design strategies for exploratory visualization of 3D data using the proposed input device, which is mounted beneath a portable display. Experiments on the use of the proposed device suggest that with some training, the users are able to improve the speed with which they can navigate the orientation and position of a virtual camera to a desired target view in the 3D visual space

Keywords—Inertial-based Interaction, Man-machine Interface, Visual Interface, Accelerometer-based Input Device

I. INTRODUCTION

Before a neurosurgeon takes careful measurements of a brain tumor, he would first need to visualize the size, shape, position and orientation of the tumor. Given that most neurosurgeons have a good visual notion of the internal 3D structure of healthy tissues, an exploratory visualization tool such as the “virtual MRI scanner” shown in Fig. 1 could help him acquire such a visual map quickly. Such a visualization unit allows him to perform interactive 3D exploratory visualization of the tumor region by seamlessly specifying the path and orientation of a 2D slice, which “acquires” and displays real-time 2D MRI images at each corresponding 3D location.

Figure 1. The inertial-controlled display unit. Touch-buttons on the left and right allow the user to turn motion sensing on or off at any time. The image on the I-CVU's display is extracted from a 2D visual slice whose 3D position and orientation can be controlled.

Such a motion-based visual interface could also be useful in helping a user control a monitoring camera mounted on a six-

axis robot to view an object remotely (see Fig. 2). The user's intuitive 3D manipulation of the inertial-controlled visualization unit (I-CVU) could be used to move the six-axis robot in a corresponding manner. This paper describes our development of such a motion-based visualization interface.

Figure 2. The motion-based visual interface for controlling the 3D position and orientation of a camera mounted on a robotic arm that is used in a remote monitoring application.

II. RELATED WORK

There are many approaches to the problem of interactive visualization of 3D volumetric data, especially those related to data obtained from volumetric MRI scans. One approach is to provide user the ability to easily visualize the volumetric MRI scan images by slicing away appropriate sections of the 3D data using mouse and keyboard interactions. David Gering's 3D Slicer software is a classic example [1]. However, exploratory visualization through this 3D image space is difficult with the traditional mouse/keyboard user interface and GUI (e.g. slider bars, etc). Fröhlich et al. introduce a novel interface call the Cubic-mouse [2] to allow the user to interactively navigate through the selected visualization position in the 3 imaging planes by pushing and retracting pencil-like control bars. Due to the distinct separation of physical control in each of the spatial axes, specifying arbitrary X-Y-Z combinations of visualization traversal paths (or point sequences) is difficult to do in an intuitive manner. This is because the executions of the required physical motion on the control bars do not correspond well to the nature of the 3D exploratory trajectory executed by the user. The study of by Van Rhijn and Mulder [3] confirms that the physical structure of the interactive device influences the stimulus-response (S-R) compatibility. Performance improves when an input device is structured such that the motion type and frame of reference of a user's action matches the frame of reference and motion type of

the virtual object being manipulated. In other words, if an input device is moved by the user in a certain way, the visualization changes in the virtual world must move in a corresponding manner. This is a fundamental design principle used in our development of the proposed I-CVU.

The inertial-sensing system in the I-CVU uses only accelerometers for motion sensing. Work on gyro-free systems can be traced back to the seminal paper by Schuler [4]. Inertial sensing has been used in numerous human computer interaction applications. They have been proposed for use as inputs for recognizing hand gestures for pointing [5, 6, 7]. Inertia sensing systems have also been stretched beyond its typical use and applied to positional tracking applications such as in [8]. Accurate tracking of positional changes with accelerometers are problematic due to the need to integrate the output of the sensor since what is measured by the accelerometer is acceleration and not displacement. The use of any dead reckoning algorithms to compute displacements over long distances will suffer from bias drifts. This problem was overcome in [8] using a Zero Velocity Update. Qian et al. [9] describes a method to control a virtual object's orientation by using a 3-axis accelerometer configuration.

In this work, we proposed a motion-based visual interface that would facilitate the development exploratory 3D visualization and control systems. The device should recognize and execute predefined intents of the user through appropriate motion gestures applied to the device. Motion initiated by the user will be appropriately mapped into corresponding visual changes in the visual space or a control system in case of a physical system, to maintain S-R compatibility.

III. THE INERTIAL SENSING SYSTEM

This section describes the hardware design of the I-CVU and the associated signal processing algorithms employed to interpret these inertial sensors' outputs in order to compute 3D position and orientation changes.

Figure 3. Placement of the four accelerometers a1 to a4 with respect to the 3 spatial axes of the display plane. Sensors a1-a2 and a3-a4 form two moment arms 1 and 2 along the X and Y axis respectively. The radius, r is the uniform distance from the device center to each sensors. During rotation about the Z-axis, both the moment arms are under the influence of tangential and centripetal forces. For clarity, only the centripetal (dotted) and tangential (dashed) forces for moment arms 1 and 2 respectively are shown, along with their typical time-varying magnitude at the start of the rotation gesture.

A. Hardware Design

We employ four 3-axis accelerometers placed at appropriate distances from the center of the device along the designated +X, -X, +Y and -Y axes respectively as shown in

the Fig. 3. The physical placement of the each accelerometer is important as it will dictate the nature of the forces acting on each individual sensor and at the corresponding output signal magnitude in each of their three axes. Without gyroscopes in our configuration, we need to capture the various rotational components applied across the three axes of the device. This is done by placing two sensors on each of the device axis to act as a moment arm (see Fig. 3) and capturing the applied tangential and centripetal forces. An ARM7 processor board is used for sensor data acquisition and the data is transferred to the computer using the serial communications port.

B. Signal conditioning and algorithms

The algorithm which estimates the position and orientation from the accelerations is a modified version of the traditional dead reckoning algorithm which employs double integration, much similar to [9]. Our modified algorithm is designed for human computer interaction and thus caters for the types of forces a user would apply to the device while holding it in front to view the display. This assumption allows us to adopt certain strategies, such as Zero Velocity Compensation [9] to reduce the duration for which integration is performed, thus bounding the resultant signals. As a result, the user can use the device for a longer period without observing significant bias drifts. The output signals from the accelerometers are first filtered using a filter to reduce the amount of noise in the signal [10]. The filtered output signal from each of the 3 axis of each accelerometer is notated as a_{nm} where $n \in [1,2,3,4]$ defines which of the four labeled sensor and $m \in [x,y,z]$ defines one of the 3 orthogonal spatial axis with respect to the display plane of the I-CVU (see Fig. 3).

The signals from the two accelerometers on the same device-axis are subtracted to capture centripetal and tangential accelerations applied to the respective device-axis during rotation. With two mutually perpendicular moment arms across the length and breadth of the device, any rotation applied by the user can be easily decomposed into its axial components as rotation about X, Y and/or Z axes. The total angular velocity is computed by integrating the tangential accelerations acting on each moment arm. Since there is no moment arm for the Z-axis, a combination of tangential forces along the X and Y axes will indicate a rotation is being performed about the Z-axis. The resultant combined angular velocity (described as a 3-component vector) is given by

$$\vec{\omega}(t) = r \int a_{T1z}(t) dt \cdot \hat{i} + r \int a_{T2z}(t) dt \cdot \hat{j} + r \int \sqrt{a_{T2x}(t) \cdot a_{T1y}(t)} \cdot \hat{k} \cdot dt \quad (1)$$

$$\text{where } \begin{aligned} a_{T1y}(t) &= a_{1y} - a_{2y} & a_{T2x}(t) &= a_{3x} - a_{4x} \\ a_{T1z}(t) &= a_{1z} - a_{2z} & a_{T2z}(t) &= a_{3z} - a_{4z} \end{aligned}$$

The unit vectors $\hat{i}, \hat{j}, \hat{k}$ correspond to the respective directions in the X, Y and Z spatial axes of the I-CVU. r is the distance of the sensor to the device centre (see Fig. 3). The values a_{T1z} and a_{T2z} correspond to the tangential forces applied on the Z-axis on the moment arms 1 and 2 respectively. The values a_{T2x} and a_{T1y} are the tangential forces applied on the X-axis of the moment arm 2 and the Y-axis of the moment arm 1 respectively. Once the angular velocities are known, the orientation of the display plane about

the X, Y or Z-axis, described as pitch (α), yaw (β) or roll (γ) respectively, can be compute using

$$\theta_\mu = \int_{t_i}^{t_f} \omega_\mu(t) dt \quad (2)$$

where θ_μ , with $\mu \in [\alpha, \beta, \gamma]$ is the current angular displacement of the display plane about the respective spatial axes from the initial orientation before tracking was initiated. The associated angular velocity ω_μ is derived from the corresponding individual vector component given in (1). Tracking the current orientation of the display using tangential forces works only when the I-CVU is moving with a non-constant velocity. As such, it is unable to give high precision estimates of absolute orientation.

Measuring orientation changes using the changes in the earth's gravitational force on the sensors can be done more precisely and against an absolute reference. This is the method employed in devices like the iPhone and Wii-Remote. Unfortunately, this more precise method of measuring orientation changes is adopted for just yaw and pitch as variations in gravitational force only occurs during rotations about the X and Y axes. For rotation about the Z-axis (roll), we employed the dead reckoning method using the tangential accelerations described earlier. Since yaw and pitch controls directly manipulate the orientation of the respective axes, the resultant signals have high amplitude, low frequency components. We term these components *static information*. Roll control and translational displacement of the I-CVU in all three axes are dependent on transients in the signal, which we term *dynamic information*. The various signals are filtered using a FIR band pass filter to extract the dynamic information and a low pass filter to extract the static information. Changes in gravity-based static information vary from very low to mid-range frequencies, while the dynamic information mostly occupies the mid-range frequencies. This unfortunate overlap makes disambiguation of rotation and translational gestures more complex and requires us to make informed use of the centripetal and tangential signals for the X and Y axes. Activity in the centripetal channel for X-axis and the tangential z-component of the same moment arm would indicate a rotation is being performed. The same is the case for the other moment. Thus (1) and (2) can be combined in quaternion form as

$$\dot{\mathbf{q}}_{k+1} = \mathbf{q}_k \cdot \boldsymbol{\omega}_{k+1} \quad (3)$$

where $\dot{\mathbf{q}}_{k+1}$ is the difference quaternion representing the new orientation state and $\mathbf{q}_k$ is the quaternion representing the orientation of the previous state, thus making the above equation a first order difference equation. The $\boldsymbol{\omega}_{k+1}$ is a zero-scalar quaternion presenting the angular velocity measured. The angles for X and Y axis are derived from the gravity-based static information and are described in (4). θ_{α_k} , obtained when $\mu = \alpha$, in (2), is the current Euler angle about X-axis (pitch) which is a component of $\mathbf{q}_{k+1}$ and is given by

$$\theta_{\alpha_k} = \begin{cases} \cos^{-1} \left(\frac{a_z}{\sqrt{a_z^2 + a_x^2}} \right), & |\dot{a}_z| < (3 \cdot E[\dot{a}_{z_n}]) \vee |\dot{a}_x| < (3 \cdot E[\dot{a}_{x_n}]) \\ \theta_{\alpha_{k-1}}, & |\dot{a}_z| > (3 \cdot E[\dot{a}_{z_n}]) \wedge |\dot{a}_x| > (3 \cdot E[\dot{a}_{x_n}]) \end{cases} \quad (4)$$

If the translational components are present, the orientation is not changed and is kept same as the previous value denoted by $\theta_{\alpha_{k-1}}$. Otherwise it is calculated using the top part of the function given in (4). Here a_x , a_y and a_z are the low frequency axial components of the total applied force which includes the gravity vector and $\dot{a}_x$, $\dot{a}_y$ and $\dot{a}_z$ are their differential signals respectively. The n subscript denotes that the signal was not under the influence of any external force when the variance was calculated and therefore contained only noise. A similar approach is applied to the current orientation angle θ_{β_k} about the Y-axis (yaw). Angular velocity about these axes can be easily calculated by differentiating (4).

The translation calculations for the device can be obtained by a double integration of the average of band pass filtered accelerations from the four sensors. The translation calculations give a position vector relative to the initial position of the device at the moment the tracking button is activated and is given by

$$\vec{P}_r = \iint_{t_i}^{t_f} \vec{a}_{AV_f}(t) \cdot dt \quad (6)$$

where $\vec{a}_{AV_f}(t)$ is the average of the band pass filtered signals (using an FIR filter) and the pre-filtered signal is obtained from the four inertial sensors and is given by

$$\vec{a}_{AV}(t) = \sum_{k=1}^4 \vec{a}_k(t) \quad (7)$$

where $\vec{a}_k(t)$ is the signal from each of the sensor k . The signal integration process is triggered by thresholds based on the standard deviations of the noise in the signal and calculated during calibration process of the I-CVU. The integration is triggered based on the following conditions

$$\text{if } |f| > (3 \cdot E[f_n]) \ \& \ \left| \frac{df}{dt} \right| > \left(3 \cdot E \left[\frac{df}{dt_n} \right] \right) \quad (8)$$

where f is any signal in under consideration and $|f|$ is the absolute value of the signal. $E[f_n]$ is the standard deviation of the noise in the signal f . Similarly $\left| \frac{df}{dt} \right|$ is the absolute value of the difference signal, while $E \left[\frac{df}{dt_n} \right]$ is its variance of the noise contained in the signal. The signal is assumed to have a zero mean in this case, however if the signal has a stable mean value, that is also calculated during calibration and subtracted out of the signal. The threshold is three times of the standard deviation as it is assumed that the noise in the signal has a Gaussian distribution. The integration is performed on the signal f as long as the relation in (8) holds true. This allows us to minimize the duration of integration and thus reduce the influence of bias drifts.

IV. INTERACTION DESIGN

This section describes the considerations in our design of the interactive exploratory visualization features of the I-CVU.

A. Display ergonomics

Firstly, the I-CVU is designed with a flat LCD display with viewing angle of $\pm 60^\circ$ in left-right direction and $\pm 30^\circ$ in the top-bottom direction. However, from our empirical studies with

several users, the comfortable natural inclination about the X-axis (see Side view in Fig. 4) is about 30° . Ergonomically comfortable wrist flexing about the X-axis is about $\pm 30^\circ$ about this natural inclination, which corresponds well with the top-bottom viewing angle of the LCD display used. Rotation limits about the Y-axis (see Front view in Fig. 4) was set to $\pm 45^\circ$ about the horizontal as further tilting produces too much visual depth differential between the left and right edges of the display screen. Fig. 4 shows the limits of the various rotational gestures adopted in interaction design strategies.

Figure 4. Rotational limits used in the interaction design.

B. Sensor constraints

There are basically two categories of sensor information produced by our inertial sensing system. The first is *static information*, which changes when the sensor orientation changes with respect to earth's gravitational force. They are useful in relaying absolute attributes such as the current tilt of the I-CVU with respect to the floor and is exploited in providing the precision control features of the I-CVU. The other type of sensor information is the *dynamic information*, which is solicited from the forces acting on the sensor during rotation (tangential forces) or non-constant translational motion of the sensor. Since such information are only present when the user apply some kind of motion to the I-CVU, this dynamic information is exploited in our design of gesture-based interactions, which will support our implementation of the coarse control features.

C. Visual interaction modes

Due to the display ergonomics and the different types of sensor information available, we have proposed three modes for visual interaction with the proposed I-CVU. They are the Browse, Follow and Jog modes. These modes can be activated once the I-CVU is put in tracking mode by turning toggling either one of the two activation buttons.

Browse mode - Consider a case in which the user wishes to rotate the current view by a large angular change of 90° about the Z-axis. Given the ergonomically limited rotation that can be applied to the I-CVU, this desired angular change can be initiated by executing a strong long jerk-like rotation of the I-CVU in the required angular direction (see Fig. 5a). The displayed image will then execute a continuous rotation in the specified direction. The user stops the rotation by applying a gentle rotation to the I-CVU in the reverse direction (see Fig. 8b). Nuances such as the speed of continuous rotation can be controlled by the strength of the initial forces applied to initiate the browse mode. This same browse mode gestures

can be applied to rotations in other 2 axes or translations in any of the 3 spatial dimensions. In short, browse mode allows the user to move through large portions of the 3D data set where detailed visual analysis is not required and the goal is to get to a visual region of interest quickly and but in a visually exploratory manner.

Figure 5. Z-axis rotation example. (a) Starting continuous rotation in the Browse mode with strong rotation gesture. (b) Stopping Browse mode with a gentle rotation gesture in the opposite direction.

To illustrate the difference in implementation of the interaction design for different axes, we describe the Z-axis and the X-axis rotations. As noted, the Z-axis rotation uses dead reckoning of the tangential forces to estimate the angular displacement. On the other hand, the X-axis rotation uses the change in the direction of gravity to calculate the orientation, which therefore allows the user to control the absolute orientation. The Browse mode allows the user to visually parse through the 3D dataset quickly in order to find a region of interest and the rate of browsing (usually associated with the user's familiarity with the dataset) is controlled by the angular velocity about the specified axis. The condition to deactivate and activate Browse mode for Z-axis rotation and the subsequent angular velocity is governed by

$$\omega_z = \begin{cases} A \cdot a_{T2x_1}, & a_{T2x_1} > D_{T2x} \\ 0, & \hat{a}_{T2x_k} \neq \hat{a}_{T2x_1} \wedge \hat{a}_{T2x_k} \neq 0 \end{cases} \quad (9)$$

where a_{T2x_k} is the current tangential acceleration for the x -component in moment arm 2 (see Fig. 3) once Browse mode is activated and is converted into an appropriate angular velocity via a pre-defined scale factor A . The initial force a_{T2x_1} activates the Browse mode when it exceeds a user definable threshold of D_{T2x} and this can be realized by simply applying enough tangential force via a strong jerk-like rotation gesture about the Z-axis. Once Browse mode is active, the angular velocity of the displayed rotation about the Z-axis will be at a constant value proportional to the strength of the initial rotational jerk gesture. The conditions to exit Browse mode are described in the lower portion of (9). The unit vectors for the initial force $\hat{a}_{T2x_1}$ and the current force $\hat{a}_{T2x_k}$ must be non-zero (i.e. rotation gesture is present) and the current force is in the opposite direction from the initial force. To switch out of Browse mode, the user simply has to apply any amount of force in the opposite direction from the initial force. The described implementation is applicable to other motions using dead reckoning, namely translations in the X, Y and Z axes.

The Browse mode for rotation about the X and Y axes is implemented differently from what was described for the Z-axis as we can take advantage of the absolute orientation

sensing capabilities about these two axes. The angular velocity about the X-axis is governed by

$$\omega_x = \begin{cases} B \cdot \theta_x, & |\theta_x| > D_R \\ 0, & \theta_x \approx N_x \end{cases} \quad (10)$$

where the θ_x is the orientation about the X-axis measured relative to the neutral orientation position given by N_x (see Side view in Fig. 4). The positive value B is a pre-defined scale factor. The Browse mode is triggered when the user changes the orientation of the display plane beyond a threshold value of D_R about the neutral orientation position N_x . In our experiments, we set D_R to about 15° . Once the device is in Browse mode, the browsing rate can be changed by simply varying the orientation of the display plane about the X-axis. We exit Browse mode when the user brings the device back to its neutral position, N_x . The same method is used to design the Browse mode for the Y-axis rotation.

Follow mode - The Follow mode uses the changes in the static information produced by the inertial sensors to allow the user to have direct control of the current orientation of the 2D visualization plane. In the Follow mode, slow deliberate tilt of the I-CVU about the X and Y axes will provide corresponding changes to the absolute orientation of the 2D visual plane; provided the current orientation θ_x (or θ_y for the Y-axis) keeps within the D_R threshold given in (10). This is part of the precise control feature of the I-CVU, which allows the user to make detailed visual analysis of the image generated by the 2D visualization plane when it is carefully adjusted to the desired 3D orientation in the visualization space. Follow mode is available for rotation about the X and Y axes only.

Jog mode - The Jog mode attempts to emulate the Follow mode visualization capabilities so that the user can bring and then hold the 2D visual plane in any arbitrary 3D position and orientation. However, because several spatial axes do not produce meaningful static information, the user's dynamic gestures are required to make incremental angular and positional changes to the 2D visual plane. In the Jog mode, small light jerks in the required direction that do not exceed the threshold of D_{T2x} given in (9) are used to move the 2D visual plane with a predefined velocity in the corresponding direction. The motion of the visual plane will be maintained until the user's jerk gesture comes to a complete halt. This is not a very efficient way to get to a precise 3D position as short incremental jerks make the process slow and long incremental jerks prevents fine grain positioning and may cause an overshoot of the intended target position. Selection of the predefined jog velocity also requires some manner of compromise. A faster velocity allows the user to move to the desired position quickly but may make the jog control too sensitive and difficult to control, leading again to frequent target overshoot. However, with some familiarization of the behavior of the visual plane motion and its relationship to the duration and strength of the small jerk gestures, users can be easily use the Jog mode to navigate to the intended viewing position. Fortunately, the Jog mode is implemented for visualization operations that normally require coarse control.

V. EXPERIMENTAL RESULTS

A series of experiments were done on an IBM® Pentium™ 4 PC with 2 GB of memory and running at 3.0 GHz. A custom rendering engine was built using Microsoft® DirectX® 9.0 with a nVidia® QuadroFX Graphics Processor Unit.

A. Experimental design

The experiments were designed to gauge the required learning curve for the users to adapt to the device. A user study comprising of a user group of six individuals (4 males and 2 females) was conducted. Each user, with only a rudimentary demonstration of the device was asked to perform three tasks. Each task was to be performed three times and the completion time was logged. The user upon the start of the experiments would perform each of the three tasks without any training. The metric for successful completion is the completion of the task with 85% of accuracy (computed from the percentage overlap between object and target outline). However, before the 2nd and 3rd attempts, the users were allowed to train on the device for about 10-15 minutes. The three tasks are detailed in Fig. 6. In each case, the user has to align and match the size of the object (letter 'E') to a reference outline and hold the view for about 2 seconds. A questionnaire was used to evaluate their experience during the user study. It evaluates four key performance indexes, namely response sensitively to intended gesture, ease of use, response intuitively to intended action and effects of training on improved performance. A rating of 1-10 was requested from the user, with a value of 1 being most negative (see Table 1).

Figure 6. (a) Screen shot of the task 1 specifically designed to manipulate multiple axes of freedom, namely X translation and Z rotation. The users have to match the black outlined letter 'E' with the yellow 'E'. Screen shots of (b) task 2, which is specifically designed to manipulate multiple axes of freedom, namely X translation, Z translation and X rotation, and (c) task 3, which is specifically designed to manipulate all rotational axes, namely X rotation, Y rotation and Z rotation.

B. Results of user performance

The results obtained for tasks 1 to 3 are shown in Fig. 7-8. Task 1 is considered easier to perform than tasks 2 and 3. All users make significant improvements in their performance in manipulating the I-CVU after having a few minutes of practice on the device, evidenced by the sharp drop in the average time taken to complete the task 1 from attempt 1, which is with no training, compared to attempts 2 and 3, which are the two attempts after spending some time trying out the device (see Fig. 7). The standard deviations among the six users were rather high, suggesting that some people are better at using inertial or motion-based interactive devices than others. However, with some exposure to the use of such devices, the variability in proficiency among the different users decreased. This is true of the simpler task 1 but with more the difficult tasks 2 and 3 (see Fig. 8), short training periods during the

study had little impact on variability in user's proficiency. However, all users showed improvement in performance for tasks 2 and 3 after some amount of training compared to the performance when they first used the device. In summary, it is encouraging to observe that given some training in using a novel interactive visualization devices like the proposed I-CVU, the majority of users are able to improve their performance when navigating in a 3D visual environment and controlling the virtual 3D camera to a desired 3D orientation and position. This suggests such devices have the potential to be used as effective interactive device for exploratory visualization albeit with some measure of acclimatization.

Figure 7. The graph shows the average time taken (in milliseconds) and its standard deviations for completing task 1 versus the attempt number. The data shown is the average of all six users in the user study for each of the 3 attempts. The first attempt is done without training and the second and third attempts are done after some training.

Figure 8. Average time taken (in msec) and its standard deviations for performing (a) task 2 and (b) task 3 versus the attempt number.

Table 1. The table shows average values of various parameters gauged in our questionnaire. The device scores highest for sensitivity and intuitive design.

Ease of use	Effect of training	Intuitiveness	Sensitivity
6.25	6	7.25	6.75

Table 1 shows the summary of replies to the questionnaires collected from our user group. User feedback suggest that most of them have better than neutral opinions of the various aspect of the I-CVU, namely its ease of use, their performance improvement after training, its intuitiveness and its sensitive response to their intended gestures. The most positive feedback was given to fact that the visualization device responded intuitively to intended action of the user. This characteristic is important as it speaks well of the fundamental concept driving the design of the proposed inertial-controlled visualization unit. In essence, with the I-CVU, the user's

action matches well with the resulting motion of the virtual object in the visual space and this strongly influences the S-R compatibility rating of the device [3].

VI. CONCLUSIONS AND FUTURE WORK

This paper discussed issues related to the implementation of an interactive display interface for performing exploratory visualization and camera view control in 3D environments. We described the related hardware design and signal processing algorithms and motivated the design of the accompanying visual interaction strategies that took into account the characteristics of this input modality. Results from the experimental studies suggest that the use of such motion-based device is generally intuitive and the user's performance of various visual navigation tasks significantly improves with limited exposure and training on the novel device.

The use of only accelerometers for motion sensing makes such device highly portable and low cost but many technical challenges still remain in providing seamless and responsive motion tracking. The sensor signal integration process used in the 3 translational axes and the Z-axis rotation makes it difficult to provide precisely-controlled motion-based interactions. Furthermore, the slight time lag introduced by our current implementation of the FIR band pass filters produces a delayed response between motion and view change, giving rise to frequent overshoot in getting to the target 3D position and orientation. This limits the interactive control significantly. We are currently looking at reducing this time lag using an improved filtering algorithm.

REFERENCES

- [1] Gering, D. 3D Slicer. <http://www.slicer.org/>
- [2] Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems .
- [3] van Rhijn, A., Mulder, J.D. 2006. Spatial input device structure and bimanual object manipulation in virtual environments. In Proceedings of the ACM Symp. on Virtual Reality Software and Technology, 51 – 60.
- [4] Schuler, A.R. 1967. Measuring rotational motion with linear accelerometers. IEEE Transactions on AES, 3,367-382.
- [5] Bang, W.-C., Chang, W., Kang, K.-H., Choi, E.-S., Potanin, A., Kim, D.-Y. 2003. Self-contained spatial input device for wearable computers. In Proc.7th IEEE Intel Symp. on Wearable Computers.
- [6] Tokoro, Y., Terada, T., Tsukamoto, M. 2009. A pointing method using two accelerometers for wearable computing. In Proc. of the ACM Symp. on Applied Computing.
- [7] Yang, J., Choi, E.-S., Chang, W., Bang, W.-C., Cho, S.-J. Oh, J.-K., Cho, J.-K., Kim, D.-Y. 2004. A novel hand gesture input device based on inertial sensing technique. In the 32nd Annual Conf. of the IEEE Industrial Electronics Society.
- [8] Ojeda, L., Borenstein, J. Personal dead-reckoning system for GPS-denied environments. 2007. In IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR'07). 1-6.
- [9] Qian, L., Chen, W.-Y., Xu, G.-P. 2009. Application of novel rotation angular model for 3d mouse system based on MEMS accelerometers. Journal of Shanghai Jiaotong University (Science), 14(2):137-142.
- [10] Kay, S.M.. 1993. Fundamentals of statistical signal processing: estimation theory. Prentice Hall, New Jersey.