Digital Processing of Magnetic, Angular-Rate and Gravity Signals for Human-Computer Interaction

Neeranut Ratchatanantakit

Florida International University, nratc001@fiu.edu

Follow this and additional works at: https://digitalcommons.fiu.edu/etd

Recommended Citation

https://digitalcommons.fiu.edu/etd/4891

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact dcc@fiu.edu.
FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

DIGITAL PROCESSING OF MAGNETIC, ANGULAR-RATE AND GRAVITY SIGNALS FOR HUMAN-COMPUTER INTERACTION

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Neeranut Ratchatanantakit

2021
To: Dean John L. Volakis  
College of Engineering and Computing  

This dissertation, written by Neeranut Ratchatanantakit, and entitled Digital Processing of Magnetic, Angular-Rate and Gravity Signals for Human-Computer Interaction, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

_______________________________________  
Jean Andrian  

_______________________________________  
Malek Adjouadi  

_______________________________________  
Wensong Wu  

_______________________________________  
Armando Barreto, Major Professor  

Date of Defense: November 5, 2021  

The dissertation of Neeranut Ratchatanantakit is approved.

_______________________________________  
Dean John L. Volakis  
College of Engineering and Computing  

_______________________________________  
Andrés G. Gil  
Vice President for Research and Economic Development and Dean of the University Graduate School  

Florida International University, 2021
DEDICATION

To my professors, friends, and family
ACKNOWLEDGMENTS

The process of earning a doctorate is long and arduous. The completion of this study could not have been possible without the expertise and support of Dr. Armando Barreto, my major advisor. He has constantly encouraged, given significant guidance, and braced me through the research process. Honestly, without his support and patience, I would give up this degree two years ago. Thank you very much, professor. I would also like to recognize and thank my committee members Dr. Jean Andrian, Dr. Malek Adjouadi, and Dr. Wensong Wu, for their advice and expertise.

I would certainly be remiss not to mention my senior, tutor, colleague, friend, and “brother” (I would say), Dr. Nonnarit O-larnnithipong. Over the past ten years, he has been my big support in all angles. His encouragement and guidance facilitated challenges that I encountered since in bachelor’s through my Ph.D. program. During my study, I am fortunate to be supported and comforted by many friends here at FIU: Praew Chantarasinlapin, Lukkamol Prapkree, Nalat Sornkhampan, Julian Gil, Steward Schwarz, Kelvin Gomez, Daniel Walls, Thodsapon hunsanon, Pontakorn Sonchan, and many more.

I would like to extend my love and special thanks to my family and friends in Thailand, especially Somsunee Kiatwacharawit and Pirasda Wongthongsuk, who always stick up for me. Without their enduring supports and encouragement, this accomplishment would never have been possible.

The research in this dissertation was supported by National Sciences Foundation grants CNS-1532061, CNS-1920182, and the FIU Graduate School Dissertation Year Fellowship.
ABSTRACT OF THE DISSERTATION

DIGITAL PROCESSING OF MAGNETIC, ANGULAR-RATE AND GRAVITY SIGNALS FOR HUMAN-COMPUTER INTERACTION

by

Neeranut Ratchatanantakit

Florida International University, 2021

Miami, Florida

Professor Armando Barreto, Major Professor

This dissertation pursued the definition and evaluation of a processing approach for robust real-time orientation estimation of a miniature Magnetic, Angular-Rate, Gravity (MARG) module for use in a human-computer interaction system that also uses a 3-camera IR-video module for position estimation. The proposed algorithm introduces the novel idea of spatially mapping the level of trustworthiness of the magnetometer-based potential corrections to the orientation estimate. This trustworthiness level is used to reduce the strength of magnetometer-based corrections of the orientation estimate where the magnetic field distortion invalidates the assumptions necessary for those corrections.

The new algorithm addresses the three research questions posed in this dissertation by 1.) Compensating for the gyroscope drift error 2.) Creating a voxel map with the values of magnetic distortion in specific regions of the operating space of the system, and 3.) Combining two different types of data to accurately track hand motion through adaptive quaternion interpolation.

The algorithm was evaluated in an experiment with thirty human subjects, processing signals from one MARG module and a 3-camera IR video system. The
results verified that the new algorithm, using the Gravity Vector and the Magnetic North vector with Double SLERP interpolation (GMV-D), reduced the drift of the orientation estimates in areas with and without magnetic distortion.

The Kruskal-Wallis test, with the error in the Phi, Theta, and Psi Euler angles as dependent variables, was used to study 3 orientation estimation methods: Kalman Filtering, GMV-D, and its precursor, GMV-S (which uses a single SLERP operation). In the magnetically undistorted area, there were no significant differences for the Phi and Theta angles. However, in the magnetically distorted area, significant differences in method performance were found for all three Euler angles, with GMV-D consistently reporting the lowest mean rank and the Kalman Filter reporting the highest.

The proposed GMV-D method makes the MARG orientation estimation more robust by fully taking advantage of the MARG operating conditions in a typical human-computer interaction application and by comprehensively utilizing all the sensing modalities available in the MARG module.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 1 - INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation and Statement of the Problem</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Hand Gestures</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Hand Tracking using IMUs – Limitations and Concerns</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Problem Statements and Hypotheses</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Literature Review on Gyroscope drift correction and IMUs limitations</td>
<td>6</td>
</tr>
<tr>
<td>1.5.1 General Motion Tracking</td>
<td>6</td>
</tr>
<tr>
<td>1.5.2 IMUs tracking body motion</td>
<td>8</td>
</tr>
<tr>
<td>1.5.3 Real-time Hand tracking by any method</td>
<td>10</td>
</tr>
<tr>
<td>1.5.4 Alternative algorithms for gyroscope drift corrections</td>
<td>11</td>
</tr>
<tr>
<td>CHAPTER 2 - TRANSFORMATIONS IN THREE-DIMENSIONAL SPACE</td>
<td>17</td>
</tr>
<tr>
<td>2.1 Three-Dimensional Coordinate Systems</td>
<td>17</td>
</tr>
<tr>
<td>2.2.1 The Inertial Frame</td>
<td>18</td>
</tr>
<tr>
<td>2.2.2 The Body Frame</td>
<td>19</td>
</tr>
<tr>
<td>2.2.3 The Earth-Centered-Earth-Fixed Frame</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Transformations</td>
<td>20</td>
</tr>
<tr>
<td>2.3.1 Direction Cosines</td>
<td>21</td>
</tr>
<tr>
<td>2.3.2 Euler Angles and Euler Rotation</td>
<td>22</td>
</tr>
<tr>
<td>2.4 Quaternions</td>
<td>24</td>
</tr>
<tr>
<td>2.4.1 Quaternion Definition</td>
<td>24</td>
</tr>
<tr>
<td>2.4.2 Basic Definitions and Properties of Quaternions</td>
<td>24</td>
</tr>
<tr>
<td>2.4.3 Quaternion Transformations</td>
<td>27</td>
</tr>
<tr>
<td>2.4.4 Conversion between Euler Angles and Quaternion</td>
<td>29</td>
</tr>
<tr>
<td>CHAPTER 3 – MEMS MAGNETIC, ANGULAR-RATE, GRAVITY (MARG) SENSORS</td>
<td>31</td>
</tr>
<tr>
<td>3.1 Gyroscopes</td>
<td>31</td>
</tr>
<tr>
<td>3.2 Accelerometers</td>
<td>33</td>
</tr>
<tr>
<td>3.3 Magnetometers</td>
<td>34</td>
</tr>
<tr>
<td>3.4 Errors and Limitations in MEMS Inertial Sensors</td>
<td>35</td>
</tr>
<tr>
<td>3.4.1 Gyroscopes Errors</td>
<td>35</td>
</tr>
<tr>
<td>3.4.2 Limitations of Accelerometers</td>
<td>38</td>
</tr>
<tr>
<td>3.4.3 Limitations of Magnetometers</td>
<td>38</td>
</tr>
<tr>
<td>CHAPTER 4 - SENSOR FUSION AND ORIENTATION CORRECTION ALGORITHM</td>
<td>41</td>
</tr>
<tr>
<td>4.1 Gravity-Magnetic Vector Compensation (GMV)</td>
<td>41</td>
</tr>
<tr>
<td>4.1.1 Spherical Linear Quaternion Interpolation</td>
<td>45</td>
</tr>
<tr>
<td>4.2 Gravity-Magnetic Vector Compensation with Double SLERP (GMV-D)</td>
<td>46</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1.1 Functional parameters of different types of IMU components from Foxlin [13].</td>
<td>4</td>
</tr>
<tr>
<td>Table 4.1 Four possibilities of the final estimate orientation from the Double SLERP method.</td>
<td>47</td>
</tr>
<tr>
<td>Table 4.2 Relations between the current magnetic North vector ( (m_0) ) and the corrected quaternion, ( \mu q_{Gpost} ), given the Magnetic Correction Trustworthiness ( (\mu) ) value.</td>
<td>52</td>
</tr>
<tr>
<td>Table 5.1 Specifications of Yost Lab 3-spaceTM sensor Micro USB [71].</td>
<td>62</td>
</tr>
<tr>
<td>Table 6.1 Estimated means and Standard Deviation of the orientation output. (In degree)</td>
<td>77</td>
</tr>
<tr>
<td>Table 6.2 Tests of Normality: Kolmogorov-Smirnov and Shapiro-Wilk</td>
<td>80</td>
</tr>
<tr>
<td>Table 6.3 Levene's Test of Equality of Error Variances.</td>
<td>82</td>
</tr>
<tr>
<td>Table 6.4 Summary results of Kruskal-Wallis Test for the orientation errors in all three angles (Phi, Theta, Psi), across three different methods in the non-magnetically distorted area.</td>
<td>84</td>
</tr>
<tr>
<td>Table 6.5 Summary results of Kruskal-Wallis Test for the orientation errors in all three angles (Phi, Theta, Psi), across three different methods at the magnetic distortion area.</td>
<td>89</td>
</tr>
<tr>
<td>Table 8.1 Three model of 3-Space sensors provided by Yost Labs</td>
<td>100</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

FIGURE PAGE

Figure 1.1 Inertial Motion tracking process ................................................. 4

Figure 2.1 The Cartesian Coordinate in Three-Dimensional Space .................. 18

Figure 2.2 Body frame of an aircraft ............................................................... 19

Figure 2.3 Earth-Centered-Earth-Fixed Reference Frame. Adapted from Mathworks [55]. .......................................................... 20

Figure 3.1 A classic gyroscope (left), A structure of MEMS gyroscope from Howtomechatronics.com [62] (right) ......................................................... 32

Figure 3.2 Micro-structure of MEMS accelerometer, at rest position (left) and applied acceleration (right). .................................................. 33

Figure 3.3 The Hall Effect principle ................................................................. 34

Figure 3.4 Types of Inertial sensors based on their bias Stability and Scale Factor stability. [65] ................................................................. 35

Figure 3.5 (Top) Raw gyroscope with Bias offset error. (Bottom) Integrated result in angle with drift .................................................. 36

Figure 3.6 (a) Office environment space where the experiment took place. (b) 3D plot of Magnetic Vector recordings in the 5x5x5 Dense Grid (separation between nodes in 1’ in all directions) .................................................. 39

Figure 3.7 The bar chart of mean error at the most critical point (1,1,5) compared with the least critical point (3,5,1) from day 1 to day 30 ...................................... 40

Figure 4.1 The flow chart explained Gravity-Magnetic Vector Compensation (GMV) Algorithm ................................................................. 41

Figure 4.2 The output (o) from the interpolation between two quaternions (q0 and q1) with the control parameter (h) .................................................. 45

Figure 4.3 Calculated alpha with Four different slopes (mα), smallest value of slope = Blue, Largest value of slope = Violet .................................................. 49

Figure 4.4 Calculated MU with Four different slopes (mµ), smallest value of slope = Blue, Largest value of slope = Violet .................................................. 53

Figure 4.5 All the panels display the evolution of the output quaternion, qOUT, of the Single SLERP vs Double SLERP with varying Slope mα and mµ at 4 Scenarios (a) Rotation without Magnetic Distortion, (b) Rotation with Magnetic
Distortion, (c) Translation without Magnetic Distortion, and (d) Translation with Magnetic Distortion. ................................................................. 55

Figure 4.6 The cubic grid with voxels in the working space. Gray voxels indicate \( \mu = 0 \). White voxels indicate non-zero \( \mu \), i.e., no significant magnetic distortion. 57

Figure 5.1 The infrared camera model V120: Trio. From OptiTrack, NaturalPoint, Inc., March 2021 [69]. ................................................................. 59

Figure 5.2 The operation volume size of the OptiTrack V120: Trio from side view (left) and top view (right). [70] ................................................................. 59

Figure 5.3 A wooden box with Infrared-reflective markers. ........................................ 60

Figure 5.4 Yost Lab 3-spaceTM sensor Micro USB with dimension ................................ 61

Figure 5.5 The non-ferromagnetic box with Yost Lab 3-spaceTM sensor attached. ... 63

Figure 5.6 The illustration of the hand inside the fabric glove with the sensor attached. ........................................................................................ 64

Figure 5.7 The subject performs the experiment at the testing area. ....................... 65

Figure 5.8 The experiment station compares with MU plot in 3D cube voxel. Red cube indicated the magnetic distortion (MU<0.9). ............................ 66

Figure 5.9 Block diagram of the orientation correction algorithm using the gravity vector and magnetic North vector with Double SLERP. ......................... 67

Figure 5.10 Flowchart showing the implementation of Gravity Magnetic Vector using Double SLERP corrections algorithm for one iteration, where pos.X, pos.Y and pos.Z obtained from OptiTrack’s Unity Plugin. ........................................ 69

Figure 5.11 The movement guide animation in Non-magnetic distorted and Magnetic distorted area showing identical rotation. .................................. 70

Figure 6.1 The evolution of the 4 quaternion components from 4 orientation estimation methods for a complete duration of experiment with interval sampling rate = 10ms ........................................................................................................ 74

Figure 6.2 3D hand rendering oriented according to the orientation estimation output from 4 different methods while in the non-magnetically distorted area (Poses 1-5) ........................................................................................................ 75

Figure 6.3 3D hand rendering oriented according to the orientation estimation output from 4 different methods while in the magnetically distorted area (Poses 6-10). ........................................................................................................ 76

Figure 6.4 Estimated Marginal means of the orientation errors for Phi (In degrees) ... 78
Figure 6.5 Estimated Marginal means of the orientation errors for Theta (In degrees) 78

Figure 6.6 Estimated Marginal means of the orientation errors for Psi (In degrees) 79

Figure 6.7 The Normal Q-Q plot of Phi 80

Figure 6.8 The Normal Q-Q plot of Theta 81

Figure 6.9 The Normal Q-Q plot of Psi 81

Figure 6.10 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle Phi, across three different methods in the non-magnetically distorted area 85

Figure 6.11 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle Theta, across three different methods in the non-magnetically distorted area 86

Figure 6.12 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle Psi, across three different methods in the non-magnetically distorted area 87

Figure 6.13 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle Phi, across three different methods in the magnetically distorted area 90

Figure 6.14 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle Theta, across three different methods in the magnetically distorted area 91

Figure 6.15 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle Psi, across three different methods in the magnetically distorted area 92

Figure 8.1 Alpha value obtained from Gyroscope data (top), and Accelerometer data (bottom) 103

Figure 8.2 Comparison between Stillness vs Motionlessness with Average method (top) and after applied offset product (bottom) 104

Figure 8.3 Square wave pulse showing on/off Accelerometer Correction periods 106
CHAPTER 1 - INTRODUCTION

1.1 Motivation and Statement of the Problem

Today, computers are part of many aspects of human activity. There is great interest in research that seeks to enhance human-computer interactions, developing new ways in which users can provide input to computer systems and mechanisms by which the users can perceive the output from computers. In particular, there is a current interest in developing systems that could allow users to provide input to the computer in ways that are more natural and intuitive. Several studies have attempted to develop alternate methods to achieve input and output to and from computer systems, beyond the general input devices (e.g., keyboard, mouse, and etc.). Voice command systems have been developed by numerous research groups since 1952 [1] and are becoming one of the most commonly used input methods, in both computer and mobile systems, providing a natural way to give input commands and dictation to computer systems. Similarly, 3D graphics have become widely adopted by some computer industries [2], such as gaming and designing [3] [4], pursuing the representation of computer objects in highly realistic ways. As the realism of computer representations has been enhanced, the popularity of Virtual Reality (VR) and Augmented Reality (AR) has increased in many applications [5]. In this current context, it would be highly beneficial for computer systems to have the real-time ability of capturing hand movement in terms of both position and orientation. This kind of capability would enable the development of natural human-computer interaction systems that could operate intuitively in 3D space [6].

One of the key objectives of using VR and AR is to create a simulated environment and make the user feel immersed in that 3D world, giving him/her the
ability to interact with that world in a natural and effortless fashion. VR and AR developers endeavor to provide simulated stimulation to several senses, such as vision, hearing, touch, or even smell, to foster the sensation of presence in the user. In these types of environments, hand motion tracking is an interesting choice for users to interact more naturally with 3D interfaces than regular input devices like mice, game pads, joysticks and wands, which require the user to perform highly artificial sequences of actions. These unnatural actions may play an important role in reducing the sensations of immersion and presence experienced by the users. In contrast, hand motion tracking can be used to implement a more intuitive way to provide input to the computer and provide the computer with a mechanism to gauge the user’s body language as the user grabs and holds objects, moving objects from place to place in the surrounding environment. Thus, an improvement in the computer’s capability to perform real-time hand tracking may be a crucial step towards the development of the next generation of human-computer interaction systems.

1.2 Hand Gestures

Several studies present different methods to capture hand gestures, such as video-based systems, and systems based on resistor strips, or Inertial Measurement Units (IMUs). Video-based systems are computer vision systems that utilize image processing algorithms to perceive the motion of the hand and its configuration. Many studies propose methods using image processing to perceive and record gestures of the hand [7]. These systems may provide hand tracking capabilities on the basis of images from one or several cameras. However, the principle of operation for these systems requires the hand to be always visible by the cameras. In addition, these systems also
need high resolution cameras and can tolerate only a minimal amount of interference (noise). Furthermore, the system needs enough lighting to capture the hand clearly. [8] Force Sensing Resistors, usually available as resistor strips, physically measure the movement of the fingers and the configuration of the hand posture by detecting the resistance changes due to stretching of the strip as the fingers bend [9] [10]. The resistor strips do not measure the orientation or the position of the hand. They can detect some actions of the fingers, such as grabbing, flexing, and pointing. Many studies that attempted to record complex hand gestures have resorted to combining resistor strips with IMUs to simultaneously acquire hand orientation. [11].

The use of miniaturized MEMS Inertial Measurement Units (IMUs) is one of the emerging approaches for hand motion tracking. The gyroscopes in IMUs provide the angular velocity of the moving hand parts and, by accumulation, can yield their orientation angle. Different types of IMUs are used in many industries and have a wide range of accuracy levels that tend to be in proportion to their price, from Navigation Systems in aviation to the smaller size IMUs used in portable devices (e.g., mobile phones). Therefore, it could be expected that IMUs might be utilized to determine the orientation of the human hand for interaction in virtual 3D spaces.

1.3 Hand Tracking using IMUs – Limitations and Concerns

Micro-Electro-Mechanical Systems (MEMS) accelerometers were first introduced in the 1990s and MEMS gyroscopes in 2009 [12]. The combination of MEMS accelerometers and gyroscopes are called Inertial Measurement Units (IMUs). MEMS magnetometers are included in some models. In that case, the sensor module is called a Magnetic, Angular-Rate, Gravity (MARG) module. The angular velocity given from the gyroscopes in IMUs can, in principle, be accumulated through numerical
integration to obtain the orientation (angle) of an object in space. By mounting the MEMS IMU sensor on a moving object (which defines the “Body Coordinate Frame”), it can measure the object’s acceleration and angular change. This method can be used to determine the current position of the object with respect to a fixed frame of reference (“Inertial Coordinate Frame”) by using double numerical integration of the acceleration to determine position. Figure 1.1 shows the tracking process from the inertial motion using a tri-axial gyroscope and a tri-axial accelerometer.

Manufacturers provide many types of IMUs for different purposes. Each type is unique in terms of its level of accuracy, which varies with the price range of the different types. The different grades of IMU have significantly different error and noise characteristics, as shown in Table 1.1, listing the accuracy of IMUs utilized for strap-down inertial position tracking (from Foxlin’s study [13].)

Table 1.1 Functional parameters of different types of IMU components from Foxlin [13].

<table>
<thead>
<tr>
<th>Component</th>
<th>Commercial-Grade</th>
<th>Tactical-grade</th>
<th>Navigation-grade</th>
<th>Strategic-grade</th>
<th>Geophysical limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gyro bias stability</td>
<td>150°/hr</td>
<td>15°/hr</td>
<td>0.015°/hr</td>
<td>0.000015°/hr</td>
<td>0°/hr/deg</td>
</tr>
<tr>
<td>Gyro bias initial uncertainty</td>
<td>0.15°/hr</td>
<td>1.5°/hr</td>
<td>0.0015°/hr</td>
<td>0.0000015°/hr</td>
<td>0°/hr/deg</td>
</tr>
<tr>
<td>Accel bias stability</td>
<td>1 mg/°/hr</td>
<td>100 μg/°/hr</td>
<td>10 μg/°/hr</td>
<td>0.5 μg/°/hr</td>
<td>0 μg/°/hr</td>
</tr>
<tr>
<td>Accel bias initial uncertainty</td>
<td>0.25 mg</td>
<td>10 μg</td>
<td>1 μg</td>
<td>0.1 μg</td>
<td>0.1 μg</td>
</tr>
<tr>
<td>Initial orientation alignment</td>
<td>1 arcsecond</td>
<td>1 arcsecond</td>
<td>1 arcsecond</td>
<td>1 µrad</td>
<td>0.01 µrad</td>
</tr>
</tbody>
</table>

Table 1.1 lists the uncertainty and stability of the readings obtained from different types of gyroscopes. This type of error is called “the bias offset error,” and is generated in the gyroscopes by providing a non-zero output when the sensors are static,
and no actual physical rotation is taking place. The bias offset error can result in high levels of orientation tracking error, as the angular velocity readings have to be integrated over time. This type of orientation error, called “drift,” is a common effect that continues to grow through time as the sensor operates and keeps integrating the rotational speed measurements to yield current orientation values (an approach also known as “Dead Reckoning”).

This dissertation proposes to use MEMS modules that contain three accelerometers, three rate gyroscopes, and three magnetometers, in combination with IR cameras to obtain a real-time estimate of the orientation and position of the hand of a human subject. In order to overcome the problems of orientation measurement through the use of gyroscopes that were described in previous paragraphs, an algorithm is introduced to fix the drift error utilizing the signals from all the sensors available in the MEMS modules.

1.4 Problem Statements and Hypotheses

Question 1: Can the gyroscope drift problem in the MARG module be corrected using high-level, adaptable MARG module orientation corrections?

Hypothesis 1: The gyroscope drift problem in the MARG module will be corrected using the proposed algorithm, which determines and compensates the bias offset error and performs orientation correction on the basis of the gravity vector measured by the accelerometer and the magnetic North vector measured by the magnetometer.

Question 2: Can we map the magnetic distortions within the working area and use that mapping to adapt the relevance of magnetometer-based correction in the calculation of the overall orientation estimates produced?
Hypothesis 2: The system will create a voxel map and indicate the value of magnetic distortion in the specific cubic voxels visited by the sensor, determined by comparison of orientation estimates derived from magnetometer readings and from accelerometer readings, for intervals when the sensor is determined to be static. The number of voxels with an assessment of magnetic distortion level will grow as the system continues in operation through time.

Question 3: Can we build the system to track the hand motion using several sensor modalities (gyroscope, accelerometer, and magnetometer)?

Hypothesis 3: The proposed hand motion tracking system will make it possible to efficiently track the human hand movement at each movable segment of the hand, attaching one MARG module to it. The system will make it possible to accurately track the hand motion by combining two different sources of data acquisition.

1.5 Literature Review on Gyroscope drift correction and IMUs limitations

Many systems use IMUs to track orientation in diverse applications, where the drift experienced in the dead reckoning approach is a common error known by the researchers. There are several studies and developments on denoising and error correction of IMUs to address this problem. Each method employs a different type of model and calculation. This section presents some examples of denoising methods for various types of applications that attempt to track parts of the body with IMUs. In addition, some specific technologies for real-time hand tracking are listed in this section.

1.5.1 General Motion Tracking

Kalman Filtering is a popular method used in many applications for gyroscope drift error correction. Zhang [14] integrates a low-cost IMU with an odometer for the
purpose of mapping an underground 3-D pipeline. Applying Extended Kalman Filtering, the outlier measurements from the odometer, due to the wheel-slip, were removed. His system, which included robust Kalman Filter, showed comparable results with an expensive Duct Runner pipeline mapping system in the market. DW. Qiu applied Kalman Filter to improve the running time and reduce IMU drift for his binocular vision mobile measurement system. With the combination of cameras and an IMU system, his work could improve the measurement accuracy of stereoscopic vision by 84.0% [15]. In Attitude and Heading Reference and System (AHRS) estimation, Yang has introduced a fusion algorithm of Fast Euler Singular Value Decomposition Cubature Kalman Filter. His algorithm improves the filter accuracy, compared with the Cubature Kalman Filter alone, in both low and high dynamic flight conditions [16]. All three studies above have shown the improvement of their systems to achieve an individual task, but using Kalman Filter led to increases in complexity that they had to face. For example, it was essential to properly estimate the covariance matrices involved in the model.

To avoid the difficulty of Kalman Filter, Khankalantary applied an adaptive constrained type-2 fuzzy Hammerstein neural network (T2FHNN) on his Strap-down inertial navigation system / Global navigation satellite system (SINS/GNSS) approach [17]. During the GNSS outage, MEMs measured the vehicle acceleration and rotation rate using gyroscopes and accelerometers, which represent a SINS Dynamic. This proposed algorithm consists of four layers, identified as: input, nonlinear static gain, linear dynamic, and output layer. The stability of the algorithm depended on the learning rate of the HNN. A low learning rate can delay the system; on the other hand, a high learning rate can lead to the un-stability of the HNN, which related to the vehicle maneuvers. A type-2 Fuzzy system was applied to set the learning rate for HNN
weights. To overcome the drift error from the IMUs, GNSS was picked to compensate for the error. However, since GNSS is based on satellite data, the signal may be obstructed. Brossard [18] developed a Deep Learning method for denoising IMU gyroscopes for Open-Loop Attitude Estimation. The neural network computed the gyro correction from the leveraging information present in the past local window, where the learning problem of using time-varying data for calibration parameters remained. His proposed algorithm obtained remarkably accurate attitude estimates, which offered a new perspective for inertial learning methods. However, the performance of the algorithm is based on a prudent design and feature selection of a dilated convolutional network, and an appropriate loss function leveraged for training on orientation increment at the ground truth frequency.

1.5.2 IMUs tracking body motion.

Ergonomists have proposed different ways to measure body movement, and they consider a direct measurement method as the most efficient tool. Using IMUs is one of the approaches followed to correct kinematics. Qilong Yuan [19] studied dynamic motion from 4 activities: walk, run, jump, and swing. He mounted IMUs on the lower limb, the right thigh, the right shank, and the right foot of the subject to capture the lower limb's motion. He presented an uncertainty-based fusion approach with a model based on magnetic and acceleration measurements for an Extended Kalman Filter orientation estimator. This method achieved some success in tracking the different types of motion with and without acceleration shocks, but it still showed need for improvement to study long-term and extremely dynamic situations. Ming Hao [20] proposed a Smoother-Based method for 3-D foot trajectory estimation using IMUs attached to the heel of each shoe. The Zero Velocity Update (ZUPT) method was applied, followed by smoothing, and then an Extended Kalman Filter was used to
reduce drift-related error. The result showed a good performance among similar methods and can improve the accuracy of foot trajectory both on stride length and stride width. However, this method is still limited under some indoor assumptions and did not correct the drift error of yaw due to the sensitivity of magnetometers with indoor environments. ChandraTjhai [21] has also studied the walking motion. In this approach, IMUs were mounted at the shank and foot for each side. Rauch Tung Striebel (RTS) filter is used to yield motion and eliminate drift, while IMUs needed pre-calibration to remove bias before use. RTS also has a reverse loop, which improves the robustness. This proposed system presented similar movement result as the high-precision robot arm that was used as a reference. Nevertheless, the RTS filter required a proper initialization, and the sensor frames have to be aligned with the vertical gravity axis and the patient's direction of motion for the setup. Kiyoshi Irie [22] studied the swing motion in sports (for example, in table tennis) measured using an IMU and a monocular camera. The clocks from both sensors must be precisely synchronized at the beginning of the recordings. To overcome the IMU error, the Least-Squares method with Gauss-Newton, Hessian matrix, and Newton methods are applied to Loop-closure for Attitude, velocity, and position, respectively. The proposed method significantly improved the accuracy of the inertial motion estimation, compared to similar approaches, which will be useful for sports skill evaluations. However, the process of synchronizing the clock has to be careful since it is the key to the system, and it is highly sensitive.

Operation of modern prosthetics requires real-time feedback from the artificial body parts to enhance their performance. S. Shaikh [23] developed a Stump Angle Measurement (SAM) system using the accelerometers in IMUs to measure the “tilt” angle of the residual stump during various phases of the gait of a low-cost electronic
knee prosthesis. The sensor was placed and measured the thigh angle. He applied a Kalman Filter to improve the accuracy of IMU estimations. In addition, the SAM system tried to remove high-frequency noise from the gait events. The results from IMU provided a reliable feedback signal and successfully detected the gait events with 93% accuracy rate but only under limited movement conditions. Application to wider mobility conditions will be studied in the future. Clement Duraffourg [24] also studied the real-time estimation of the pose of a lower limb prosthesis, placing the IMU at a shank. The data was captured during a swing of the gait cycle. The system used a nonlinear complementary filter with a variable gain to estimate attitude. The algorithm did not use the magnetometer in the IMU and assumed a drift-free operation in the experiment. This system allowed a better real-time estimation of the trajectory, potentially allowing faster gait mode detection.

1.5.3 Real-time Hand tracking by any method

To capture human hand gesture and motion, several studies have introduced systems using different types and numbers of sensing units. This section presents some examples of hand motion tracking systems and technologies.

Haihan Duan [25] tracked hand gestures using an ambient light-based system. The system relied on ubiquitous ambient light and low-cost photodiodes. A recurrent neural network (RNN) was used to process the signal data, which gave an accurate recognition rate [26] of 99.31%. The system only captures the shape of the hand but not its orientation. Dinh-Son Tran also worked with a neural network (3D convolutional) for spotting hand gestures in real-time using a camera-based system from the Kinect RGB-D camera. Kinect provides a skeletal tracker, which may be useful for tracking finger and hand shape. The system first extracts the hand region from the depth image, and then points the position of a fingertip using the K-cosine
corner detection algorithm. The data is then processed with a 3D convolutional neural network to recognize the gesture. The experiment was done with 50 participants, which generated 5250 samples. The researchers randomly selected 70% of data as training data, 10% for validation, and 20% as testing data. The results with 3DCNN models gave an accuracy rate of more than 90% but required training times of more than an hour and could be negatively impacted by changes in ambient lighting. Gabyong Park [27] combines a depth camera and a gyroscope to enhance the 3D hand tracking feature for fast movement. A matrix was constructed in order to alleviate the drift problem from the gyroscope. The time delay and pose offset between a depth camera and a hand-worn gyroscope have to be calibrated before the beginning of the system operation. With this sensor-fusion method, the system successfully captures hand motion during fast movements even if the image is distorted due to excessive motion blur. Shuo Jiang [28] attached a stretchable e-skin patch on the back of the hand to recognize hand gestures by estimating skin strain. Each finger flexion changes the strain of the muscle at the back of the hand. Therefore, his study used this approach to classify the hand gestures for simple American Sign Language signs (e.g., numbers 0-9). The results showed classification accuracies of more than 90%. However, there are two key challenges in this research that must be overcome, the skin strain complexities and the sensitivity of the sensor.

1.5.4 Alternative algorithms for gyroscope drift corrections

Over the last several decades, there have been numerous approaches proposed for orientation estimation fusing measurements from gyroscope and accelerometer readings (i.e., from IMU modules) first, and later including the additional fusion of magnetometer readings. Nazarahari and Rouhani [29] offer a fairly comprehensive and systematic cataloguing of those efforts, identifying 3 major families of approaches:
Vector Observation (VO), Complimentary Filter (CF) and Kalman Filters (KF). Further, these same authors did an experiment to compare 36 Sensor Fusion Algorithms from their survey which could be categorized into 7 major groups: Linear Complementary Filter (LCF), Nonlinear Complementary Filter (NLCF), Linear Kalman Filter (LKF), Extended Kalman Filter (EKF), Complementary Kalman Filter (CKF), Square-root Unscented Kalman Filter (SRUKF), and Square-root Cubature Kalman Filter (SRCKF) [30]. They concluded that the method proposed by Sabatini [31] showed lower errors when time is not a factor, while the methods from Hua et. al [32] or Justa et. al [33] were found to give the best results when the execution time is a factor.

Harindranath [34] created a simulation platform in MATLAB that compared another four popular orientation algorithms in a 9-axis MARG unit which are Madgwick Algorithm [35], Mahony Algorithm [36], Extended Kalman Filter and Two Stage KF-Q update. It was found that the Mahony filter performed best in the low level of noise while the Two Stage KF-Q worked well in a higher noise level.

The Kalman Filtering, envisioned by Dr. Rudolf E. Kalman [37], is one of the most popular methods to estimate the state of a dynamic system for which there is a model and instantaneous observations [38]. In its simple form the Kalman Filter obtains an enhanced (“posterior”) estimate fusing the results predicted from a model (used as “prior” estimate) with instantaneous measurements through Bayesian Estimation. It is noteworthy that in the Kalman Filter the predicted estimate resulting from the model is “corrected” in the second stage of the process by involvement of the instantaneous measurements. The final state estimate that results has fused the result from the model and the information derived from the instantaneous measurements according to the level of “trustworthiness” of both sources (represented by their
covariance matrices). Although the original Kalman Filter was developed on the basis of a linear state transformation and the assumption of Gaussian distributions for all the quantities involved, subsequent versions (e.g., “Extended” and “Unscented”) of the Kalman Filter concept have been adapted for application to non-linear models.

J.L. Marins [39] studied an extended Kalman Filter for Quaternion-Based Orientation Estimation Using MARG Sensors of a rigid body motion in real-time. According to the highly nonlinear functions of the process state variables from 3 sensors (gyroscope, accelerometer, and magnetometer), the partial derivatives needed for Kalman Filtering were very complicated and not applicable for real-time applications. To be able to use Kalman Filter in real-time, Marins needed to apply the Gauss-Newton iteration algorithm in his method. The complexity of using Kalman Filtering correction in real-time process is also shown in Peppoloni’s study [40] as it highlights the difficulty in correctly setting up and updating the critical Kalman Filter parameters.

Some commercial MARG modules used to offer an internally implemented (Extended) Kalman Filter. For example, the 3-Space sensor Micro USB model from Yostlab [41] provides the Kalman Filter as a default filter mode. In it, statistical techniques were used to combine normalized sensor data and reference vector data into a final orientation which be directly read from the module (This has been done in this research to have a basis for comparison). However, Yostlab has more recently developed an alternative proprietary filter algorithm called “QGRAD2” [42]. They claim that this new sensor fusion algorithm is more efficient and has 3 times faster update rate than their original Kalman Filter. Another potential drawback on many Kalman-based approaches is that the levels of “trustworthiness” of the several sources of information (gyroscope-based model and accelerometer- and magnetometer-based
measurements) are represented by constant covariance matrices, which is not an exact match with the reality that governs the appropriateness of the accelerometer- or magnetometer-based correction estimates. As Mahoney et al. pointed out [36] “Traditional linear Kalman Filter techniques including EKF techniques have proved extremely difficult to apply robustly to applications with low quality sensors systems.”

More recently, some groups have attempted to consider dynamic assessments of the varying levels of “trustworthiness” that must be assigned in making accelerometer- and magnetometer-based orientation corrections. Madgwick [35], indicates that his algorithm “contains 1 (IMU) or 2 (MARG) adjustable parameters defined by observable system characteristics.” For the case of an IMU (i.e., no magnetometer) he indicates “The filter gain, $\beta$, represents all mean zero gyroscope measurement errors, expressed as the magnitude of a quaternion derivative” However, Madgwick adjusted these parameters just once for each of his implementations (“The proposed filter's gain $\beta$ was set to 0.033 for the IMU implementation and 0.041 for the MARG implementation.”) These parameters are not, therefore truly dynamic in terms of being re-assigned as frequently as every iteration according to the instantaneous circumstances in which the system is performing.

There have also been studies that have attempted to assess and account for the reduced level of “trustworthiness” associated to the magnetometer-based corrections. Roetenberg et al. [43] showed that the Kalman Filter’s performance would deteriorate when there is a disturbance of the magnetic field. They attempt the identification of a possible magnetic disturbance on the basis of the magnitude of the total magnetic flux recorded by the magnetometer and the detection of change in the magnetic inclination (“magnetic dip angle”) derived from the magnetometer measurements and the current orientation estimate of the MARG module. Then in 2015, Daponte et al. [44] proposed
a new way to estimate the orientation of MARG units with the capability of compensating short-duration magnetic disturbances using the Gradient Descent Algorithm. He continued his research for long-duration disturbances from the magnetometer embedded on a smartphone in 2017 [45] by modeling the disturbances from both hard iron and soft iron. Jin Wu [46] proposed a novel nonlinear optimization approach to address magnetometer disturbances in real-time, using the interior-point method. He proposed that his unique solution could correct the issue efficiently and with a fast response, but this proposed method only deals with soft-iron disturbances that are not very strong.

In recent years, Valenti et al. [47] and the group at the FIU DSP laboratory have independently proposed an alternative way to execute the “correction” phase of the sensor fusion algorithm for MARG orientation estimation. Based on the work of Shoemake [48], Valenti proposed the use of linear quaternion interpolation (LERP), for small corrections and spherical linear quaternion interpolation (SLERP) only for larger corrections, calculating the correction quaternion (i.e., $\Delta q_{\text{acc}}$) and executing the correction in the global (inertial) frame of reference. Based on the report from Dam et al. [49] the FIU group proposed the use of SLERP for all corrections, calculating the quaternion correction (i.e., $\Delta q_A$) and executing it in the MARG’s body frame, in an algorithm, called GMV-S [50]. Valenti et al. control the amount of interpolation through a parameter $0 < \alpha < 1$, which characterizes the cutoff frequency of their complementary filter, as described by Franceschi and Zardi [51]. In GMV-S the interpolation control variable is derived from the “confidence” parameter representing how close the MARG is to being static [41]. In their report, Valenti et al., found that their method achieved better results in all three angles (roll, pitch, and yaw) in
comparison to the Madgwick filter and the Extended Kalman Filter for situations with and without magnetic disturbances.

In 2019, Meyer developed an approach similar to Valenti’s but with a more cautious incorporation of the accelerometer measurement \[52\]. He added layers for checking the accelerometer magnitude error, rate of change of the magnetic field vector and a model of the gyro bias before he fed the data into the estimation correction system. He performed a simulation test with modeled Gaussian noise added to the three sensors. In results from 50 test runs, his model was successful in distinguishing slow dynamic maneuvers from biases and could eliminate the problem of pseudo steady states. However, he indicated that it was too soon to be sure that the filter would be suitable for extended periods of dynamic motion and the filter’s robustness to magnetic distortion would need to be investigated more extensively.
CHAPTER 2 - TRANSFORMATIONS IN THREE-DIMENSIONAL SPACE

This chapter will briefly explain the equations and models available for the transformations in three-dimensional space that apply to this research. The linear three-dimensional transformations consist of four types: Translation, Rotation, Scaling, Shearing, and Reflection. This research focuses only on rotation. Two main systems that are used in this research to model rotations are Euler angles and Quaternions.

2.1 Three-Dimensional Coordinate Systems

The locations of points or geometric elements can be represented in coordinate systems, which may be orthogonal or non-orthogonal. Non-orthogonal coordinate systems are rarely used for the study of the topics relevant to this dissertation, while the orthogonal are most commonly used. With the information from the coordinate system, measuring distance, area, volume, and direction can be obtained using an algebraic system. Examples of orthogonal coordinate systems are the Cartesian or Rectangular, the Circular Cylindrical, the Spherical, the Elliptic Cylindrical, the Parabolic Cylindrical, the Conical, the Prolate Spheroidal, the Oblate Spheroidal, and the Ellipsoidal. This research will primarily use the Rectangular or Cartesian system. The Cartesian coordinate system in three-dimensional space consists of three axes (x, y, z), perpendicular to each other. In a two-dimensional space, the x-axis is the horizontal axis, where the numbers increase from left to right. The y-axis is the vertical axis, where the numbers increase in an upward direction. These two axes form a plane where the y-axis is 90 degrees counterclockwise apart from the x-axis. In three-dimensional space, the z-axis is added to the x-y plane. The direction assigned to the z-axis depends on the use of “left-hand rule” or the “right-hand rule”. The left-hand rule and the right-hand
rule are shown in Figure 2.1, where the thumb, index finger, and middle finger indicate the x-axis, y-axis, and z-axis, respectively. The curved arrows represent the direction of positive rotation about each axis according to the rule.

![Figure 2.1 The Cartesian Coordinate in Three-Dimensional Space](image)

The German geodesist F.R. Helmert [53] introduced a transformation method within three-dimensional space that changes coordinates from one reference frame to another. Defining the frames of reference and their axes is necessary to unambiguously specify the location, direction, and orientation of objects in a navigation system. In this research, there are three reference frames used to describe the three-dimensional rotation: the inertial frame, body frame, and earth-centered-earth-fixed frame. All the frames are orthogonal and right-handed Cartesian frames.

2.2.1 The Inertial Frame

A fundamental coordinate system, used in the study of Newton’s laws of motion, is the inertial frame. This frame of reference is commonly used when a body is at rest (or moving at a constant speed in a linear motion) [54]. The origin of this frame is located at the center of the Earth. The frame is not rotating under the gravitational influence of the stars.
2.2.2 The Body Frame

The body frame is the frame commonly used to represent an object under study. The origin of this frame is located at the center of gravity of the rotating object. The axes are determined along the forward direction as x-axis (roll), in the right direction as y-axis (pitch), and through-the-floor direction as z-axis (yaw). Figure 2.2 depicts the body frame of an aircraft, indicating the corresponding directions of rotation.

![Body frame of an aircraft](image)

*Figure 2.2 Body frame of an aircraft*

2.2.3 The Earth-Centered-Earth-Fixed Frame

The third frame of reference considered is fixed to the Earth, with its origin located at the center of the Earth's mass. The Earth frame is constantly rotating with the polar motion of the Earth about the international reference pole (IRP), which is defined as the z-axis. The point where the prime meridian in Greenwich (0° longitude) and the Earth’s equator (0° latitude) intersect, creates the fixed x-axis of the Earth’s frame. The x-axis and z-axis generate a plane, and the y-axis is a line perpendicular to this xz-plane. The graphic in Figure 2.3 shows the location of the axes of the Earth’s frame.
2.3 Transformations

A transformation is a process for re-defining the position and orientation of a rigid body from one frame to another with respect to known coordinates. It can be simply defined by a vector of coordinate differences which is equally applied to all the points in a frame. If the frame coordinates are orthogonal, the transformation will be linear. There are three approaches to implement the linear transformation: Direction Cosines, Euler Angles (or Rotation Angles), and Quaternions. In this research, Quaternions is the main approach applied in the proposed algorithm and will be described in detail in the next section.
2.3.1 Direction Cosines

The direction cosines are used to define the angles between the three coordinate axes and a vector, which characterize the direction of the vector. To find the relationship between two frames, “A” and “B,” let’s consider that each frame is represented by orthogonal unit vectors $\hat{x}, \hat{y}, \hat{z}$. Each vector in the frame is defined by components $x, y, z$. Therefore, the frames A and B are shown as the matrices in Equations 2.1 and 2.2.

$$A = [\hat{x}_A, \hat{y}_A, \hat{z}_A] = \begin{bmatrix}
X_{A1} & X_{A2} & X_{A3} \\
Y_{A1} & Y_{A2} & Y_{A3} \\
Z_{A1} & Z_{A2} & Z_{A3}
\end{bmatrix} \quad (2.1)$$

$$B = [\hat{x}_B, \hat{y}_B, \hat{z}_B] = \begin{bmatrix}
X_{B1} & X_{B2} & X_{B3} \\
Y_{B1} & Y_{B2} & Y_{B3} \\
Z_{B1} & Z_{B2} & Z_{B3}
\end{bmatrix} \quad (2.2)$$

Then the relationship between frame A and frame B is defined as $A^B_C$, where the original frame and the transformed frame are indicated in the lower index and upper index, respectively. $A^B_C$ can be calculated using the dot product, which equals the cosine of the angle between the two-unit vectors. The direction cosine matrix is derived as the following equations.

$$A = A^B_C \quad (2.3)$$

$$C^A_B = A B^{-1} = \begin{bmatrix}
\hat{x}_A \cdot \hat{x}_B & \hat{y}_A \cdot \hat{x}_B & \hat{z}_A \cdot \hat{x}_B \\
\hat{x}_A \cdot \hat{y}_B & \hat{y}_A \cdot \hat{y}_B & \hat{z}_A \cdot \hat{y}_B \\
\hat{x}_A \cdot \hat{z}_B & \hat{y}_A \cdot \hat{z}_B & \hat{z}_A \cdot \hat{z}_B
\end{bmatrix} \quad (2.4)$$

$$\begin{bmatrix}
\hat{x}_A \cdot \hat{x}_B & \hat{y}_A \cdot \hat{x}_B & \hat{z}_A \cdot \hat{x}_B \\
\hat{x}_A \cdot \hat{y}_B & \hat{y}_A \cdot \hat{y}_B & \hat{z}_A \cdot \hat{y}_B \\
\hat{x}_A \cdot \hat{z}_B & \hat{y}_A \cdot \hat{z}_B & \hat{z}_A \cdot \hat{z}_B
\end{bmatrix} = \begin{bmatrix}
\cos \theta_{\hat{x}_A, \hat{x}_B} & \cos \theta_{\hat{y}_A, \hat{x}_B} & \cos \theta_{\hat{z}_A, \hat{x}_B} \\
\cos \theta_{\hat{x}_A, \hat{y}_B} & \cos \theta_{\hat{y}_A, \hat{y}_B} & \cos \theta_{\hat{z}_A, \hat{y}_B} \\
\cos \theta_{\hat{x}_A, \hat{z}_B} & \cos \theta_{\hat{y}_A, \hat{z}_B} & \cos \theta_{\hat{z}_A, \hat{z}_B}
\end{bmatrix} \quad (2.5)$$

The direction cosine matrix has these properties:

$$C(C)^T = I \Rightarrow C = C^{-1} = (C)^T \quad (2.6)$$
The relative orientation is dominated by only three degrees of freedom due to the orthogonality assumption of the dependence among the nine elements in C. Now, the relationship between frames A and B can be derived as Equations (2.7) to (2.9), following the properties above.

\[ C_B^A = (C_B^A)^T \]  
\[ \begin{bmatrix} \bar{x}_B & \bar{y}_B & \bar{z}_B \end{bmatrix}^T = C_B^A \begin{bmatrix} \bar{x}_A & \bar{y}_A & \bar{z}_A \end{bmatrix}^T \]  
\[ \begin{bmatrix} \bar{x}_A & \bar{y}_A & \bar{z}_A \end{bmatrix}^T = C_B^A \begin{bmatrix} \bar{x}_B & \bar{y}_B & \bar{z}_B \end{bmatrix}^T \]  

2.3.2 Euler Angles and Euler Rotation

The Euler angles are a common system to describe the orientation relation of two frames in three-dimensional Euclidean space by a sequence of rotations. These coordinate frames are successfully transformed using the rotation matrices about specific axes. These rotations around axes follow the direction cosine matrix for a right-handed (counterclockwise) rotation, creating the rotational angles or Euler angles as \( \phi \), \( \theta \), and \( \psi \) for the x-axis, y-axis, and z-axis, respectively. Therefore, the rotation matrices of a coordinate frame are as follows:

\[ R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \]  
\[ R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \]  
\[ R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

A 3-by-3 rotational matrix ‘\( R \)’ is used to multiply a 3-by-1 coordinate vector in order to rotate a primary vector in three-dimensional space. The new vector can be obtained from Equation (2.13).
\[ \vec{V}_2 = R \vec{V}_1 \]

\[
\begin{bmatrix}
    v_{2x} \\
v_{2y} \\
v_{2z}
\end{bmatrix} =
\begin{bmatrix}
    r_{11} & r_{12} & r_{13} \\
r_{21} & r_{22} & r_{23} \\
r_{31} & r_{32} & r_{33}
\end{bmatrix}
\begin{bmatrix}
    v_{1x} \\
v_{1y} \\
v_{1z}
\end{bmatrix}
\]  

(2.13)

Euler’s Theorem states: “Any two independent orthogonal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis” [56]. There are twelve different independent rotation sequences in three Euclidean axes, which are:

- xyz, xzy, xzx, xzy
- yxy, yxz, yzx, yzy
- zxy, zxz, zyx, zyz

In the global coordinate frame, the rotation about the x-axis is called ‘roll’, the rotation about the y-axis is called ‘pitch’, and the rotation about the z-axis is called ‘yaw’. The Euler angle-axis sequence ZYX is commonly used to describe rotations in the aerospace field for tracking the aircraft’s heading and attitude. The global rotation matrix is shown in the following equations:

\[
R_{zyx} = R_x(\psi)R_y(\theta)R_x(\phi)
\]

\[
= \begin{bmatrix}
\cos(\psi) & -\sin(\psi) & 0 \\
\sin(\psi) & \cos(\psi) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos(\theta) & 0 & \sin(\theta) \\
0 & 1 & 0 \\
-\sin(\theta) & 0 & \cos(\theta)
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\phi) & \sin(\phi) \\
0 & -\sin(\phi) & \cos(\phi)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\cos(\psi)\cos(\theta) & \cos(\phi)\cos(\psi)\sin(\theta) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta) \\
\cos(\psi)\sin(\theta) & \cos(\phi)\cos(\psi) + \sin(\theta)\sin(\phi) & \cos(\phi)\sin(\psi) - \cos(\phi)\cos(\psi)\sin(\theta) \\
-\sin(\theta) & \cos(\phi)\sin(\phi) & \cos(\psi)\cos(\phi)
\end{bmatrix}
\]

This rotation matrix \( R_{abc} \) where \( a, b, \) and \( c \) can be any \( x, y, \) and \( z \) are an alternative transformation of the direction cosine matrix but expressed in the Euler angles form.

Even though the direct cosine matrix and Euler angles are very straightforward and clearly describe the rotation transformation in three-dimensional space, there is a weakness in this approach. When the pitch angle approaches +/- 90 degrees and causes
the loss of one degree of freedom in a three-dimensional space, the phenomenon called “Gimbal lock” occurs. Gimbal lock happens when the axis of the first rotation is parallel to the third rotation axis and becomes a system in two-dimensional space. The gimbal lock blocks the possibility of finding the individual angles for each of the axes. To avoid the gimbal lock disadvantage, quaternions were introduced to express rotations in three-dimensional space.

2.4 Quaternions

2.4.1 Quaternion Definition

Quaternions were first proposed by an Irish mathematician, William R. Hamilton, in 1843 [57]. A quaternion is a combination of a real number and three imaginary components which makes it a hyper-complex number. A quaternion ‘q’ can be denoted in different ways. For the orthogonal basis in R³, a quaternion is defined as the grouping of a vector $\vec{q}$ in three-dimensional space and a scalar $q_w$, as shown in Equation 2.15.

$$q = \vec{q} + q_w = q_x\hat{i} + q_y\hat{j} + q_z\hat{k} + q_w$$  \hspace{1cm} (2.15)

In the previous Equation, $q_x$, $q_y$, $q_z$, and $q_w$ are all real numbers. Therefore, the quaternion can turn to be the orthogonal basis of R⁴ simply defined using only those four scalar quantities, as shown in Equation 2.16. This definition and arrangement of quaternion will be used throughout this research.

$$q = [q_x, q_y, q_z, q_w]$$  \hspace{1cm} (2.16)

2.4.2 Basic Definitions and Properties of Quaternions

Consider, $\mathbf{a} = a_x\hat{i} + a_y\hat{j} + a_z\hat{k} + a_w$ and $\mathbf{b} = b_x\hat{i} + b_y\hat{j} + b_z\hat{k} + b_w$ as two quaternions.

**Addition**: The addition of two quaternions is a quaternion.
The sum of $a$ and $b$ is defined by Equation 2.17.

$$a + b = (a_x + b_x)i + (a_y + b_y)j + (a_z + b_z)k + (a_w + b_w)$$ (2.17)

**Multiplication**: The multiplication of a quaternion with a scalar is a quaternion. Let $c$ be a scalar. Equation 2.18 shows the multiplication of a quaternion and a scalar.

$$ca = (ca_x)i + (ca_y)j + (ca_z)k + (ca_w)$$ (2.18)

Also, the multiplication of two quaternions is a quaternion. However, the fundamental products of the elements $i, j, and k$ must be considered as in Equations 2.19 to 2.22.

$$i^2 = j^2 = k^2 = ijk = -1$$ (2.19)

$$ij = k = -ji$$ (2.20)

$$jk = i = -kj$$ (2.21)

$$ki = j = -ik$$ (2.22)

The product of two quaternions is indicated using the symbol ‘⊗’. Equation 2.23 defines the product of two quaternions.

$$a ⊗ b = (a_xi + a_yj + a_zk + a_w)(b_xi + b_yj + b_zk + b_w)$$ (2.23)

$$= a_xb_xi^2 + a_xb_yij + a_xb_zik + a_xb_wi$$

$$+ a_yb_xj^2 + a_yb_yj^2 + a_yb_zjk + a_yb_wj$$

$$+ a_zb_xk^2 + a_zb_yk^2 + a_zb_zk^2 + a_zb_wk$$

$$+ a_wb_xi + a_wb_yj + a_wb_zk + a_wb_w$$

Note that, the quaternion multiplication is non-commutative.

By applying the fundamental rules, the product of two quaternions can be simplified and rewritten as Equation 2.24.

$$a ⊗ b = a_wb_w - (a_xb_x + a_yb_y + a_zb_z) + a_w(b_xi + b_yj + b_zk) + b_w(a_xi + a_yj + a_zk) + (a_yb_z - a_zb_y)i + (a_xb_z - a_zb_x)j + (a_xb_y - a_yb_x)k$$ (2.24)
The quaternion multiplication can also be written in matrix form by grouping the coefficients for each component: \( \hat{i}, \hat{j}, \hat{k} \) and real part together using Equation 2.23.

\[
\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix}
    b_w & b_z & -b_y & b_x \\
    -b_z & b_w & b_x & -b_y \\
    b_y & -b_x & b_w & b_z \\
    -b_x & b_y & -b_z & b_w \\
\end{bmatrix}
\begin{bmatrix}
    a_x \\
    a_y \\
    a_z \\
    a_w \\
\end{bmatrix}
\]  

\( (2.25) \)

**Conjugation** : A quaternion \( a \) can define a conjugate \( a^* \) as in Equation 2.26.

\[
a^* = -a_x \hat{i} - a_y \hat{j} - a_z \hat{k} + a_w
\]

\( (2.26) \)

**Norm** : The norm of a quaternion is scalar, denoted by \(|a|\). It is simply described as the square root of the sum of each component squared, as shown in Equations 2.27 and 2.28.

\[
|a| = \sqrt{a^* \otimes a}
\]

\( (2.27) \)

\[
|a| = \sqrt{a_x^2 + a_y^2 + a_z^2 + a_w^2}
\]

\( (2.28) \)

**Unit quaternion** : If the quaternion \( a \) is a unit quaternion, denoted as \( u \), then the norm is equal to 1, \(|u| = 1\). Equation 2.29 defines a unit quaternion \( u \) which equals the multiplication of the inverse norm of quaternion \( a \), \( \left( \frac{1}{|a|} \right) \), and that quaternion \( a \).

\[
u = \frac{1}{|a|} \left( a_x \hat{i} + a_y \hat{j} + a_z \hat{k} + a_w \right)
\]

\( (2.29) \)

**Inverse** : The inverse of a quaternion \( a \) is defined as in Equation 2.30, where \( a^{-1} a = a a^{-1} \). If, \( a \) is a unit quaternion, its inverse is simply its conjugate \( a^* \).

\[
a^{-1} = \frac{a^*}{|a|^2}
\]

\( (2.30) \)

**Exponential** : The exponential of a quaternion is denoted by \( e^a \) as shown in Equation 2.31. The scalar quantity of the quaternion \( a^n \) can directly apply the exponential rule. However, the vector quantity of quaternion \( \hat{a} \) is derived through the Taylor series expansion for the exponential function. The final result is shown in Equation 2.32, from
O-larnnithipong [58], which applied Taylor series expansion formulas for sine and cosine.

\[
    e^a = e^{(\bar{a} + a_w)} = e^{\bar{a}} e^{a_w} \tag{2.31}
\]

\[
    e^a = e^{a_w} \left( \frac{\bar{a}}{\|\bar{a}\|} \sin(\|\bar{a}\|) + \cos(\|\bar{a}\|) \right) \tag{2.32}
\]

### 2.4.3 Quaternion Transformations

The previous section has explained most of the essential basics of quaternion algebra. Now we will apply those basic equations to perform a three-dimensional rotation using quaternions. The simple rotation of a point, or a vector within one coordinate frame, can be obtained as in Equation 2.33 where the \( \vec{v} \) is a vector, in three-dimension space and \( \vec{w} \) is a rotated vector with angle of \( \theta \) around an axis in the same reference frame.

\[
    \vec{w} = q \otimes \vec{v} \otimes q^* \tag{2.33}
\]

Where, \( q \) is a unit quaternion (i.e., norm \( ||q|| \) equal to 1).

Previously, the rotation of Frame A with respect to Frame B, was denoted as \( R_B^A \). This is equivalent to \( \tilde{q}_B^A \) in terms of quaternions. One property of rotating a reference coordinate frame with respect to another coordinate frame is that the rotation of frame “A” with respect to frame “B” can be paired with the rotation of frame “B” with respect to frame “A” depending on the point of view. Therefore, the quaternion \( \tilde{q}_B^A \) is equal to the quaternion \( \tilde{q}_A^B \) in counter-rotation, as shown in Equation 2.34. Then Equations 2.35 and 2.36 describe the relationship of vector \( \vec{v}_A \) and \( \vec{v}_B \) referenced in frames “A” and “B”, respectively.

\[
    \tilde{q}_B^A = \tilde{q}_B^A^* \quad \text{or} \quad \tilde{q}_B^A = \tilde{q}_A^B^* \tag{2.34}
\]

\[
    \vec{v}_B = \tilde{q}_A^B \otimes \vec{v}_A \otimes \tilde{q}_A^B^* \tag{2.35}
\]

\[
    \vec{v}_B = \tilde{q}_B^A \otimes \vec{v}_A \otimes \tilde{q}_B^A \tag{2.36}
\]
According to the multiplication rule for quaternions, the $R^3$ vectors $\mathbf{v}_A$ and $\mathbf{v}_B$ must be transformed into quaternion space $R^4$. First, the three components of the $R^3$ vector become the three imaginary elements, in the $i, j, and k$ directions. Then a zero real part is added. Equation 2.37 represents any vector $\mathbf{\tilde{v}}$ in the form of quaternion space.

$$\mathbf{\tilde{v}} \in R^4 = \mathbf{\tilde{v}} + 0 = v_xi + v_yj + v_zk + 0 = [v_x, v_y, v_z, 0]$$

(2.37)

This type of quaternion, with zero real part is called a ‘pure quaternion’.

Equation 2.38 shows the unit quaternion ‘$\mathbf{\mathbf{q}}$’ as a rotation operator which is written in trigonometrical form. The symbol $\theta$ represents a half of the rotating angle about the unit vector $\mathbf{\mathbf{u}}$ (rotational axis).

$$\mathbf{q} = \mathbf{\mathbf{\tilde{q}}} + q_w = \mathbf{\mathbf{\tilde{u}}} \sin(\theta) + \cos(\theta)$$

(2.38)

Knowing the desired axis of rotation and the angle to be rotated, the process for representing the desired rotation with quaternions is better explained considering the initial and final positions of the rotated vector as existing in a common plane. The angular relationship involved in a rotation represented by a quaternion product can be found recalling the dot product and cross product of two vectors as shown in the Equations 2.39 and 2.40, respectively, where $\mathbf{a}$ and $\mathbf{\tilde{b}}$ are the two vectors in three-dimensional space and $2\theta$ is the angle between them.

$$\mathbf{a} \times \mathbf{\tilde{b}} = ||\mathbf{a}||||\mathbf{\tilde{b}}|| \sin(2\theta) \mathbf{n}$$

(2.39)

$$\mathbf{a} \cdot \mathbf{\tilde{b}} = ||\mathbf{a}||||\mathbf{\tilde{b}}|| \cos(2\theta)$$

(2.40)

Let $\mathbf{n}$ be a unit quaternion where its norm squared equals to 1. After solving algebraic equations, the unit quaternion ‘$\mathbf{\mathbf{q'}}$’ in Equation 2.41 then represents the rotation of angle $2\theta$ about $\mathbf{n}$ and describes the angular difference between two vectors $\mathbf{a}$ and $\mathbf{\tilde{b}}$.

$$\mathbf{q'} = \mathbf{a} \times \mathbf{\tilde{b}} + (\mathbf{a} \cdot \mathbf{\tilde{b}} + ||\mathbf{a}||||\mathbf{\tilde{b}}||)$$

(2.41)
The $\vec{a} \times \vec{b}$ portion contains the vector part $\vec{q}$ of the quaternion where $(\vec{a} \cdot \vec{b} + \|\vec{a}\| \|\vec{b}\|)$ portion is the scalar part $q_w$. Then we can rewrite the equation as in Equation 2.42.

$$q' = \Delta q = H(\vec{a}, q_w) \quad (2.42)$$

### 2.4.4 Conversion between Euler Angles and Quaternion.

To better describe 3-D angles and perform rotations, it is necessary to be able to convert back and forth between Euler Angles and Quaternions.

Firstly, the angles Phi ($\phi$), Theta ($\theta$) and Psi ($\psi$) represent the value of the angles rotated about axis x, y and z respectively in the Euler Angles method. Therefore, Equations 2.43 to 2.45 show how to convert angles for each orthogonal axis into quaternion form.

$$q^l_{\phi} = \cos \left( \frac{\phi}{2} \right) + i \sin \left( \frac{\phi}{2} \right) \quad (2.43)$$

$$q^l_{\theta} = \cos \left( \frac{\theta}{2} \right) + j \sin \left( \frac{\theta}{2} \right) \quad (2.44)$$

$$q^l_{\psi} = \cos \left( \frac{\psi}{2} \right) + k \sin \left( \frac{\psi}{2} \right) \quad (2.45)$$

The common Euler angle-axis sequence used to describe a rotation in the aerospace field is ZYX. Then, Equations 2.46 and 2.47 construct the quaternion rotation of ‘$q$’ according to that sequence.

$$q = q^l_{\psi} \otimes q^l_{\theta} \otimes q^l_{\phi} \quad (2.46)$$

$$q = q_x i + q_y j + q_z k + q_w \quad (2.47)$$

Given that,

$$q_x = \cos \left( \frac{\psi}{2} \right) \cos \left( \frac{\theta}{2} \right) \sin \left( \frac{\phi}{2} \right) - \sin \left( \frac{\psi}{2} \right) \sin \left( \frac{\theta}{2} \right) \sin \left( \frac{\phi}{2} \right)$$

$$q_y = \cos \left( \frac{\psi}{2} \right) \sin \left( \frac{\theta}{2} \right) \cos \left( \frac{\phi}{2} \right) + \sin \left( \frac{\psi}{2} \right) \cos \left( \frac{\theta}{2} \right) \sin \left( \frac{\phi}{2} \right)$$
\[ q_x = \sin \left( \frac{\psi}{2} \right) \cos \left( \frac{\theta}{2} \right) \cos \left( \frac{\phi}{2} \right) - \cos \left( \frac{\psi}{2} \right) \sin \left( \frac{\theta}{2} \right) \sin \left( \frac{\phi}{2} \right) \]

\[ q_w = \cos \left( \frac{\psi}{2} \right) \cos \left( \frac{\theta}{2} \right) \cos \left( \frac{\phi}{2} \right) + \sin \left( \frac{\psi}{2} \right) \sin \left( \frac{\theta}{2} \right) \sin \left( \frac{\phi}{2} \right) \]

After the quaternion is calculated and used to rotate the vector, Equations 2.48 to 2.50 can be used to convert a unit quaternion ‘q’ back into the Euler angles.

Angle rotated about x-axis; \[ \psi = tan^{-1} \left[ \frac{2(q_w q_x + q_x q_y)}{1 - 2(q_y^2 + q_z^2)} \right] \] (2.48)

Angle rotated about y-axis; \[ \theta = sin^{-1} \left[ 2(q_w q_y + q_x q_z) \right] \] (2.49)

Angle rotated about z-axis; \[ \phi = tan^{-1} \left[ \frac{2(q_w q_z + q_y q_x)}{1 - 2(q_x^2 + q_y^2)} \right] \] (2.50)
CHAPTER 3 – MEMS MAGNETIC, ANGULAR-RATE, GRAVITY (MARG) SENSORS

Micro-electromechanical Systems or MEMS were first introduced in the 1950’s and further developed to be widely used in commercial products in the mid-1990’s [59][60]. A MEMS system is a small integrated device that consists of mechanical and electrical components. It is mostly made from the silicon material which is the same type of an electrical processor found in computers. MEMS can be constructed to use in various systems such as temperature control, air pressure control, force control, etc. MEMS have significant advantages, such as their small sizes, low power consumption and low cost of manufacturing. Nevertheless, the data obtained from MEMS have errors and noises that require calibrations and filtering processes. One type of MEMS systems that are commonly used in tracking motion or for navigation are called MEMS inertial sensors or inertial measurement units (IMUs).

An inertial measurement unit is a measurement electronic device composed of an accelerometer and a gyroscope sensor. This device may be used to capture the motion of the body. Some IMUs may include magnetometers for better measuring results. The combination of three sensors creates a 9-degree of freedom system. IMUs are commonly used in the attitude and heading reference system for ships and aircraft. With the current technological developments, new IMU modules, which are smaller in size and cheaper, are available in the market. This increases the number of applications using IMUs, such as in the mobile telephony field.

3.1 Gyroscopes

A gyroscope is a device used to measure angular velocity. It is originally a spinning disc able to spin rapidly about an axis which is itself free to alter its direction.
With the conservation of angular momentum, the orientation of the axis is not affected by tilting of the mounting and it is able to maintain its spinning axis. While the wheel is spinning along the spin axis, if there is a rotational force applied to the input axis, the rotational force will show about the output axis, which is perpendicular to the plane of the spinning and input axes according to the right-hand rule. This occurrence is known as “Gyroscopic Effect”.

MEMS gyroscopic sensors are not composed by spinning discs or gimbals. Instead, they use the concept of the Foucault pendulum, the Coriolis effect [61] and apply it to a vibrating mechanism to detect the changes in orientation. When a mass is driven constantly at velocity \( \vec{v} \), and an external angular rate \( \vec{\Omega} \) is applied, the flexible part of the proof mass would vibrate out of the plane and create a perpendicular displacement caused by the Coriolis forces \( \vec{F}_{\text{Coriolis}} \), sensed capacitively with a specific CMOS ASIC. This type of MEMS gyroscopes are called “Tuning Fork Gyroscopes” where the comb-type structures drive the tuning fork into resonance. Equation 3.1 represents the relationship between the driven velocity and the angular velocity.

\[
\vec{F}_{\text{Coriolis}} = -2m\vec{\Omega} \times \vec{v} \tag{3.1}
\]

![Figure 3.1A classic gyroscope (left). A structure of MEMS gyroscope from Howtomechatronics.com [62] (right).](image)
3.2 Accelerometers

An accelerometer is a device that is used to measure the rate of change of velocity, known as acceleration, that is applied to a body. The amount of acceleration is the combination from two types of acceleration forces: dynamic forces (external forces act to produce linear motion) and static forces (the Earth’s gravity). Some examples of MEMS accelerometers are piezo-electric, piezo-resistance and capacitive. For consumer grade applications (low cost), MEMS capacitive accelerometers are commonly preferred over other types [63]. Figure 3.2 shows the microstructure of MEMS capacitive accelerometers. They consist of two main parts, which are static and dynamic. The static part is constituted by fixed electrodes and the dynamic part has a flexible spring leg attached with a movable proof mass. In the at-rest position (no applied acceleration), the gap between the left side of a fixed plate and the movable part ($d_1$) equals the gap on the right side ($d_2$). When the acceleration is applied, the distance between both gaps changes in the direction of the force applied. Therefore, the output signal of MEMS capacitive accelerometer is generated from the change of capacitance between the fixed and the moving mass electrodes. With today’s technology, a MEMS accelerometer consists of 3 axes placed orthogonally.

![Micro-structure of MEMS accelerometer, at rest position (left) and applied acceleration (right).](image)

*Figure 3.2 Micro-structure of MEMS accelerometer, at rest position (left) and applied acceleration (right).*
3.3 Magnetometers

A magnetometer can be used to measure the direction and strength or the change in a magnetic field. A compass is a classic analog device that provides an output indicating the direction of the ambient Earth’s magnetic field. A German mathematician and physicist, Johann Carl Friedrich Gauss, introduced the magnetometer in 1833 [64]. Then in the 19th century, magnetometers were develop using the Hall effect, which is the most common method until today.

The Hall effect or Magneto Resistive Effect uses a thin metallic plate with electric current flowing through it. When a strong magnetic field passes through the plate, in a perpendicular direction, a voltage called ‘Hall voltage’, is generated, as shown in Figure 3.3. The intensity and direction of the magnetic field will disturb the flow of electrons to one side of the plate, which results in the Hall voltage.

![Figure 3.3 The Hall Effect principle.](image-url)
3.4 Errors and Limitations in MEMS Inertial Sensors

Figure 3.4 Types of Inertial sensors based on their bias Stability and Scale Factor stability. [65]

Inertial sensors have different types according to their performance and costs. MEMS Inertial Sensors are commonly used for the commercial applications due to their low cost and small size. However, they generate significant amounts of errors in their measurement. Figure 3.4 shows a graph mapping different types of Inertial Sensors with respect to their errors. The errors in commercial grade MEMS have 2 main categories, which are Systematic errors (deterministic) and Stochastic errors (random). To improve the performance of low-cost MEMS, any error from the sensor that affects the accuracy of the measurement needs to be addressed.

3.4.1 Gyroscopes Errors

The raw drift on the gyroscope of the MEMS module used in this research is rated by the manufacturer as 11º/hour, but the actual error measured from a module at the FIU DSP Lab was 55.2º/hour [66].

The drift error phenomenon involves both types of errors, Systematic and Stochastic. The Systematic errors are the module defects created in the manufacturing process and can be predicted, for example Bias offset error, Thermal, Repeatability,
Scale factor, and non-orthogonality. These types of errors can be calibrated from the raw gyroscope data using pre-assigned mathematical models.

The bias offset error is specifically targeted by the proposed algorithm. The inertial sensors should, ideally, give zero output when they are in their static states and no input forces are applied. However, even in a static state, the low-cost MEMS provide some negative and positive values that deviate from zero, called “offset”, as shown in Figure 3.5 (top). Once the raw gyroscope data is integrated to obtain the angle of the sensor shown in Figure 3.5 (bottom), the angle output from the offset value shows a pattern that can be fixed using mathematical models. An angle error ($\theta_e$) from the bias offset of gyroscope is proportional to the time (t) as shown in Equation 3.1, where $b_w$ is the bias offset error magnitude.

$$\theta_e = \int (b_w)dt = b_w t$$  \hspace{1cm} (3.1)

![Figure 3.5](image)

Figure 3.5(Top) Raw gyroscope with Bias offset error. (Bottom) Integrated result in angle with drift.

In addition to the bias offset error, Stochastic errors or Random errors are the major cause of drift in the output of the sensor. The Stochastic errors are randomly
created over time and cannot be predicted or determined by fixed models. There are some types of random processes that can be used to describe the random errors of the sensors.

The Gaussian Random process is a common process used to define a common type of random error. It is described by the normal distribution of $x(t)$ for any time (t) using its probability density function, as shown in Equation 3.2.

$$f[x(t)] = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x-\mu}{\sigma} \right)^2}$$  \hspace{1cm} (3.2)

The random walk process is another random process that may be applied to model the inertial sensor’s errors. Equation 3.3 describes the relationship of a random variable $\dot{b}(t)$ and a white noise process $w(t)$ integrated.

$$\dot{b}(t) = w(t)$$  \hspace{1cm} (3.3)

This output white noise from the MEMS will be integrated into velocity and angle errors known as “velocity random walk (VRW)” and “angular random walk (ARW)”.

To obtain more accurate outputs, those random errors needed to be addressed. The research reported in this dissertation takes advantage of the diversity of sensors in the MEMS module, using two other types of sensors included in the module, accelerometers, and magnetometers, to compensate errors. Signals from the accelerometers and magnetometers can be manipulated to also yield outputs in angular formats. Therefore, the gyroscope drift from random error can be detected and corrected by comparison to the angular results of the other sensors. By correcting the error directly to its output, all noises and errors involved in the MEMS sensor will be comprehensively addressed.
3.4.2 Limitations of Accelerometers

The accelerometers measure the rate of change of velocity produced by both dynamic forces (external forces that act to produce linear motion) and static forces (the Earth’s gravity), as described in Section 3.2. The proposed algorithm uses the exclusive acceleration due to the gravity while the module is static, or near static, to predict and compensate the output angle of the MEMS module. The vector measured from gravity always points vertically and in a downward direction, to the center of the Earth, in the Earth frame. It is necessary to restrict the use of accelerometer correction only to time interval when the MEMS module is static, and no force is applied to the sensor.

3.4.3 Limitations of Magnetometers

To correct the gyroscope drift while the sensor is moving, the use of magnetometers is an alternative option. The algorithm uses the magnetic North vector measured by the magnetometers in the MEMS module as the reference to correct the estimated output angle of the sensor. However, the magnetometers measure the direction and strength or the change in a magnetic field. Therefore, the proximity of large ferromagnetic objects, such as metal desk, electronic devices, metal shelves, etc., in the working area can cause distortions in the local magnetic field.

A study of the distortion of the magnetic field in the proposed experimental area was conducted, in order to assess the level of trustworthiness of using magnetometers as a means for gyroscope drift correction. The experiment was set up as shown in Figure 3.6(a). The non-ferromagnetic frame was built to create a 5’x5’x5’ grid with 1’ spacing in all 3 directions. Figure 3.6(b) shows the result of mapping the magnetic North vector within the grid. The level of consistency of those distortions through time was also captured during this experiment, as shown in Figure 3.7.
The results show that ferromagnetic objects do affect the magnetic field and cause the magnetic north vector to be distorted. In this working space, the metal desk is the key object causing the most significant distortion of the magnetic field.

In the space regions where the magnetic field is distorted, the correction of the module orientation errors cannot be applied directly [67].

Figure 3.6 (a) Office environment space where the experiment took place. (b) 3D plot of Magnetic Vector recordings in the 5x5x5 Dense Grid (separation between nodes in 1' in all directions)
Figure 3.7 The bar chart of mean error at the most critical point (1,1,5) compared with the least critical point (3,5,1) from day 1 to day 30.
4.1 Gravity-Magnetic Vector Compensation (GMV)

The Gravity-Magnetic Vector Compensation (GMV) approach was introduced by O-larnithipong [50]. He created an algorithm that used the benefit of sensor fusion to solve the drift problem commonly present in orientation estimates obtained from low-cost gyroscope sensors. His algorithm has the flow as described in Figure 4.1.

![Flow Chart](image)

*Figure 4.1 The flow chart explained Gravity-Magnetic Vector Compensation (GMV) Algorithm.*

Firstly, the Bias Offset was estimated. This includes the bias error generated from manufacturing defects. In this algorithm, the bias offset error is assessed for every sampling data to prepare for calculation of an unbiased angular velocity ($\tilde{\omega}_B$), which will be used in the algorithm. A simple linear regression model [68] was applied to determine the bias offset error and Equations (4.1) – (4.4) were used to obtain the unbiased angular velocity ($\tilde{\omega}_B$) by subtracting the calculated bias offset error from the raw gyroscope data.

$$\hat{b} = \beta_0 + \beta_1 t \quad (4.1)$$
\[ \beta_0 = \bar{b} - \beta_1 \bar{t} \]  
(4.2)
\[ \beta_1 = \frac{\sum_{i=1}^{n}(t_i - \bar{t})(b_i - \bar{b})}{\sum_{i=1}^{n}(t_i - \bar{t})^2} \]  
(4.3)

Then;
\[ \tilde{\omega}_0 = \tilde{\omega}_0 - \bar{b} \]  
(4.4)

Where, \( \tilde{\omega}_0 \) is the raw gyroscope reading from the sensor.

This algorithm mainly used quaternion notation to represent the rotation, which avoids ambiguities created by the gimbal lock problem. The unbiased angular velocity (\( \tilde{\omega}_B \)) in R3 is transformed into the quaternion space, R4, by augmenting a zero as the fourth component of the quaternion. The quaternion rate (\( \dot{q} \)) is calculated using the zero degree or initial state of a quaternion as \([0,0,0,1]\) and the value of unbiased angular velocity (\( \tilde{\omega}_B \)) in quaternion form as shown in Equation (4.5). This Equation can also be written in a matrix form, Equation (4.6).

\[ \dot{q} = \frac{1}{2} q_0 \otimes \tilde{\omega}_B \]  
(4.5)
\[ \dot{q} = \frac{1}{2} \begin{bmatrix} 0 & \omega_{Bz} & -\omega_{By} & \omega_{Bx} \\ -\omega_{Bz} & 0 & \omega_{Bx} & -\omega_{By} \\ \omega_{By} & -\omega_{Bx} & 0 & \omega_{Bz} \\ -\omega_{Bx} & \omega_{By} & -\omega_{Bz} & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_{0x} \\ \dot{q}_{0y} \\ \dot{q}_{0z} \end{bmatrix} \]  
(4.6)

The estimated orientation (\( q_G \)) is calculated from Equation (4.7) using the integration method, where the rate (\( \Delta t \)) is from the sampling interval used by the IMU. By integrating the quaternion rate, the orientation of the sensor module was obtained from its initial position.

\[ q_G = e^{((\Delta t)\hat{\omega} \otimes \hat{q}_0) \otimes \hat{q}_o} \]  
(4.7)

This proposed method's main goal is to correct the drift error from the gyroscope and its impact on the estimated orientation (\( q_G \)). Two additional sensing units, which are accelerometers and magnetometers, are used for that purpose. The output from the accelerometer ideally results from the acceleration due to the gravity, called "gravity
vector." Simultaneously, the magnetometer measuring the "magnetic North vector" represents the direction of the Earth's magnetic field. The concept of comparing between the sensor's body frame and the Earth's frame was applied to calculate the module's orientation using these two sensing units, as indicated by Equation (4.8). The vector in the sensor's body frame can be expressed as $B\hat{v}$, which uses the 3-factor product quaternion operator shown in Equation (4.9). By referring to Equation (4.8), the $B\hat{v}$ can be calculated using only a unit quaternion representing the sensor's body frame orientation, with respect to the earth’s frame, created in Equation (4.10).

$$\frac{B}{E}q = \frac{E}{B}q^*$$ (4.8)

$$B\hat{v} = \frac{B}{E}q \otimes E\hat{v} \otimes \frac{B}{E}q^*$$ (4.9)

$$B\hat{v} = \frac{E}{B}q^* \otimes E\hat{v} \otimes \frac{E}{B}q$$ (4.10)

The measurement from the accelerometer (gravity vector) is assumed to be pointing towards the center of the Earth, while the magnetometer results are expected to be pointing to the North magnetic pole. Both sensing units were used to follow similar approaches to the quaternion correction challenge, as described in the following equations.

**Enhancement of the estimated quaternion with gravity vector correction ($q_{GA}$):**

$$\hat{q}_{GA} = q_G \otimes \Delta q_A$$ (4.11)

where:

$$\Delta q_A = H(\tilde{q}_{Av}, q_{Aw})$$ (4.12)

$$\tilde{q}_{Av} = \tilde{a}_0 \times \tilde{a}(q_G)$$ (4.13)

and

$$q_{Aw} = ||\tilde{a}_0|||\tilde{a}(q_G)|| + \tilde{a}_0 \cdot \tilde{a}(q_G)$$ (4.14)

$$\tilde{a}(q_G) = q_G^* \otimes A_{int} \otimes q_G$$ (4.15)

In these equations $\tilde{a}_0$ is the measured gravity vector from the accelerometer, $\tilde{a}(q_G)$ is the calculated gravity vector referenced in the sensor's body frame and $A_{int}$ is the
initial gravity vector that is assumed to be pointing towards the Earth's center (in the Earth frame).

**Enhancement of the estimated quaternion with magnetic north vector correction** ($\hat{q}_{GM}$):

\[
\hat{q}_{GM} = q_G \otimes \Delta q_M \tag{4.16}
\]

where;

\[
\Delta q_M = H(\hat{q}_{Mv}, q_{MW}) \tag{4.17}
\]

\[
\hat{q}_{Mv} = \vec{m}_0 \times \vec{m}(q_G) \tag{4.18}
\]

and

\[
q_{MW} = ||\vec{m}_0|| ||\vec{m}(q_G)|| + \vec{m}_0 \cdot \vec{m}(q_G) \tag{4.19}
\]

\[
\vec{m}(q_G) = q_G^* \otimes M_{int} \otimes q_G \tag{4.20}
\]

In these equations $\vec{m}_0$ is the measured magnetic North vector from the magnetometer, $\vec{m}(q_G)$ is the calculated magnetic North vector referenced in the sensor's body frame and $M_{int}$ is the initial magnetic North vector that is assumed to be pointing to the Earth's North pole (in the Earth frame).

Both $\hat{q}_{GA}$ and $\hat{q}_{GM}$ are required to be normalized before using them as a rotation operator in the next step. As described in the discussion of the limitations of each sensor (Chapter 3), the compensation using $\hat{q}_{GA}$ could be directly and exclusively applied if the sensor module is static or moving without change of speed. Similarly, the $\hat{q}_{GM}$ represents the corrected estimated orientation that uses the measurement from the magnetometer, under the assumption that the Earth’s magnetic field is constant in orientation. This assumption may be disrupted in areas where there is significant magnetic field distortion. To define the final estimated orientation, the quaternion interpolation uses a control parameter, which assigns complementary weights to the contributions of the calculated $\hat{q}_{GA}$ and $\hat{q}_{GM}$. 
4.1.1 Spherical Linear Quaternion Interpolation

Spherical Linear Quaternion Interpolation (SLERP) or great arc interpolation is an approach that defines a final orientation estimate given two initial rotation estimates in quaternion form by using a parametric weight. Erik Dam [49] has proved the equivalence of the expressions for SLERP and introduced the SLERP without exponentiation as shown in Equations (4.21) and (4.22).

\[
\cos(\Omega) = q_0 \cdot q_1 \tag{4.21}
\]

\[
SLERP(q_0, q_1, h) = \frac{q_0\sin((1-h)\Omega) + q_1\sin(h\Omega)}{\sin(\Omega)} \tag{4.22}
\]

A parameter \( \Omega \) is obtained from the dot product of the two initial quaternions \( (q_0 \text{ and } q_1) \). The control parameter \( (h) \) varies between 0 and 1 to specify the interpolated orientation. Figure 4.2 shows a step in the interpolation, where the value of \( h \in [0,1] \), makes the resulting orientation, indicated by the endpoint (O), move from \( q_0 \) to \( q_1 \).

---

*Figure 4.2 The output (o) from the interpolation between two quaternions \((q_0 \text{ and } q_1)\) with the control parameter \((h)\)*
This SLERP method yields the optimal interpolation curve between two rotations with the preservation of the magnitude of a unit quaternion on the same great arc.

The original GMV method [50] always defines the final orientation quaternion from SLERP interpolation between \( \hat{q}_{GA} \) and \( \hat{q}_{GM} \), and it uses only one control parameter for the SLERP operation, which is \( h = \alpha \), the “stillness” parameter, using it as representative of how close the state of the module is to being static (Alpha is explained further in the following sections). If this parameter, alpha, is close to 1, the assumption that is necessary for exact validity of gravity vector correction is essentially being fulfilled and \( \hat{q}_{GA} \) is favored over \( \hat{q}_{GM} \) in the SLERP interpolation. The magnetic North vector correction contribution plays a “passive” role in the weighting of the 2 contributions and is only allowed to play a significant role in the definition of the final orientation estimate as an “else” condition, when it is known that the gravity vector correction alone will not be fully adequate.

4.2 Gravity-Magnetic Vector Compensation with Double SLERP (GMV-D)

This dissertation proposes a new multisensory orientation estimation algorithm which combines the possible ways of correcting the initial orientation estimation from gyroscope signals, \( \hat{q}_G \), in a manner that takes full advantage of all the information available to an orientation and position tracking system. In this new approach, position estimates are utilized in an interactive way to assign a value (in a 0-to-1 scale) to the trustworthiness (\( \mu \)) of a tentative magnetic North vector correction, as a counterpart to the alpha parameter (“stillness”), which quantizes the adequacy of the gravity vector correction.
Moreover, in this new approach, if the merging of both the gravity vector correction and the magnetic North vector correction is appropriate, it will be performed by an algorithm that is controlled by two parameters: $\alpha$ and $\mu$, allowing both contributions to play an “active” role in the weighing of the partial corrections. This new method of merging the partial corrections has been called “Double SLERP”

4.2.1 Double SLERP

Considering the limitations in using both accelerometers and magnetometers for orientation correction, the double application of SLERP was proposed to estimate the final orientation estimate ($\hat{q}_{out}$) in the new algorithm. The double SLERP process uses two control parameters, Stillness ($\alpha$) and Magnetic Trustworthiness ($\mu$) to define the output (final) orientation, as shown in the Equation (4.23). There are three tentative estimates of orientation available, which are the estimated quaternion ($\hat{q}_G$), the estimated quaternion with gravity vector correction ($\hat{q}_{GA}$), and the estimated quaternion with magnetic North vector correction ($\hat{q}_{GM}$). The consideration of both $\alpha$ and $\mu$ yields four possible stages as described in the Table 4.1, which also indicates the correction source that will be given preference in each of the 4 stages.

Let's $[q_0 \cdot q_1(h)] = SLERP(q_0, q_1, h) = \frac{q_0 sin((1-h)\Omega)+q_1 sin(h\Omega)}{sin(\Omega)}$

$\hat{q}_{out} = [(\hat{q}_G \cdot \hat{q}_{GM}(\mu)) \cdot [\hat{q}_G \cdot \hat{q}_{GA}(\alpha)](\alpha)]$  

(4.23)

Table 4.1 Four possibilities of the final estimate orientation from the Double SLERP method.

<table>
<thead>
<tr>
<th>STAGE</th>
<th>CONTROL PARAMETER</th>
<th>The final estimated orientation ($\hat{q}_{out}$) tends to use the compensation from:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\alpha$ close to: 0</td>
<td>$\hat{q}_G$: correction did not apply</td>
</tr>
<tr>
<td></td>
<td>$\mu$ close to: 0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$\alpha$ close to: 1</td>
<td>$\hat{q}_{GA}$: correction using gravity vector</td>
</tr>
<tr>
<td></td>
<td>$\mu$ close to: 0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\alpha$ close to: 0</td>
<td>$\hat{q}_{GM}$: correction using magnetic North vector</td>
</tr>
<tr>
<td></td>
<td>$\mu$ close to: 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$\alpha$ close to: 1</td>
<td>$\hat{q}_{GA}$: correction using gravity vector</td>
</tr>
<tr>
<td></td>
<td>$\mu$ close to: 1</td>
<td></td>
</tr>
</tbody>
</table>
4.2.2 The Control Parameters ($\alpha$ and $\mu$)

4.2.2.1 Sensor's Stillness ($\alpha$)

The stillness of the sensor indicates the confidence that exists to apply the orientation correction with the gravity vector. Section 3.4.2 described the limitations of accelerometers-based correction in different circumstances. Therefore, the control parameter of quaternion interpolation for gravity vector ($\alpha$) is calculated from the stillness data given out as the Yost Labs 3-Space Sensor's “confidence readable parameter”.

Prior to being used to determine the value of the control parameter for quaternion interpolation by gravity vector ($\alpha$), the average confidence value (stillness), read from the IMU module, was processed by a first-order Gamma memory filter to smoothen the signal. The Gamma memory filter uses a weight parameter ($W_a$) that ranges from 0 to 1, to control the filtering characteristics of the lowpass filter it implements on the signal. The first-order Gamma memory filter has the transfer function in z-domain derived in Equations (4.24) – (4.27).

$$H(z) = \frac{Y(z)}{X(z)} = \frac{w}{z^{-1}(1-w)}$$  \hspace{1cm} (4.24)

$$Y(z) = (W)z^{-1}X(z) + (1-W)z^{-1}Y(z)$$ \hspace{1cm} (4.25)

$$y[n] = (W)x[n-1] + (1-W)y[n-1]$$ \hspace{1cm} (4.26)

$$\alpha_g = W_a (Stillness(n-1)^2) + (1-W_a)\alpha_g(n-1)$$ \hspace{1cm} (4.27)

By experimentation with the IMU module, a weight parameter ($W_a$) value of 0.25 was found to be adequate to smooth the signal, while avoiding the introduction of a large amount of delay. Once the stillness signal is smoothened, a linear equation was applied to accelerate the drop of the $\alpha$ value when the sensor begins departing a static status, driving the value of the stillness parameter down. This equation is characterized by the
The higher the value of this slope, the quicker the graph will drop below 1, as shown in Figure 4.3. The final value of the control parameter for quaternion interpolation of gravity vector correction ($\alpha$) is calculated from Equations 4.28 and 4.29.

\[
\alpha' = m_a \alpha_g + (1 - m_a) \tag{4.28}
\]

\[
\alpha = \frac{\alpha' + |\alpha'|}{2} \tag{4.29}
\]

Figure 4.3 Calculated alpha with Four different slopes ($m_a$), smallest value of slope = Blue, Largest value of slope = Violet.

4.2.2 Magnetic Correction Trustworthiness ($\mu$)

From the studies described in Section 3.4.3, the distortion of the magnetic field is usually only significant in the areas around ferromagnetic items that contain soft iron. Therefore, it is possible to interactively develop a position-dependent assessment of the significance of the magnetic distortion within the overall environment in which is the IMU module is to be used. It was decided to design this position-dependent parameter to take on values between $\mu=1$ (which will indicate negligible distortion) to $\mu=0$ (which will indicate very strong distortion of the magnetic North vector’s direction). To define
this Magnetic Correction Trustworthiness parameter ($\mu$), data from both the IR cameras included in the system setup and the MARG sensors are utilized. The cameras locate the position in the working area to map the amount of magnetic distortion locally present, which is calculated using a comparison between data from the magnetometer and the accelerometer in the MARG sensor. The Magnetic Correction Trustworthiness was initialized as 'zero' for the complete working space. The calculation of a new $\mu$ value for reassignment to a specific position is performed only when the MARG sensor adopts a static state at that position (when the gamma-filtered $\alpha$ values surpass an $\alpha_{\text{Threshold}}$.) The estimated quaternion successfully corrected using the gravity vector correction ($\hat{q}_{GA}$) is now stored in a temporary variable called $\hat{q}_{Gpost}$. The concept of mapping a three-dimensional vector between two different frames was applied, as previously described in Equation 4.10. Therefore, the initial (and supposedly also current) orientation of the magnetic North vector is mapped to the module’s body frame using ($\hat{q}_{Gpost}$), as shown in Equation 4.30, as $\tilde{\mu}(q_{Gpost})$. Next, the cosine of the angle, $\gamma$, between the mapped North magnetic vector and the components of the North magnetic vector sensed by the magnetometer in the IMU ($\vec{m}_0$), is determined, as shown in Equation 4.31. The angle $\gamma$ represents the angle between the current magnetic North vector and the corrected quaternion. If this angle were 0, it would mean that the corrected estimation by means of the gravity vector and the correction by means of the magnetic North vector coincide exactly. As this whole process is undertaken only when the stillness parameter indicates that the gravity vector correction is highly justified, $\mu=0$ would, in turn, imply that the magnetic North vector correction, at this particular position, is fully trustworthy. Larger values of $\gamma$ would indicate that larger levels of distortion in the Earth’s magnetic field can be suspected at the position being
considered. The magnetic correction trustworthiness parameter ($\mu$) is calculated in Equations 4.32 and 4.33 to capture in a single number the level of adequacy of a prospective magnetic correction on the basis of $\gamma$. Table 4.2 illustrates the relation of angle $\gamma$ and the value of Magnetic Correction Trustworthiness ($\mu$). A linear equation with negative slope ($m_m$) is also applied to accelerate the change of the $\mu$ parameter, as the sensor is placed at locations where the distortion of the Earth’s magnetic fields is more and more intense. The higher of the value of the slope ($m_m$), the quicker the resulting $\mu$ value will be driven down, as exemplified in Figure 4.4.

$$\mu'(q_{\text{Gpost}}) = q_{\text{Gpost}}^* \otimes M_{\text{int}} \otimes q_{\text{Gpost}}$$  \hspace{1cm} (4.30)

$$\cos(\gamma) = \frac{\bar{m}_o\mu(q_{\text{Gpost}})}{|\bar{m}_o||\mu(q_{\text{Gpost}})|}$$  \hspace{1cm} (4.31)

$$\mu' = -m_m(acos(cos(\gamma))) + 1$$  \hspace{1cm} (4.32)

$$\mu = \frac{(1 + \mu)}{2}$$  \hspace{1cm} (4.33)

(if $\mu < 0$, $\mu$ is overwritten as 0)
Table 4.2 Relations between the current magnetic North vector ($\mathbf{m}_0$) and the corrected quaternion, $\tilde{\mu}(q_{\text{Gpost}})$, given the Magnetic Correction Trustworthiness ($\mu$) value.

<table>
<thead>
<tr>
<th>Vectors</th>
<th>$\gamma$</th>
<th>$\cos(\gamma)$</th>
<th>$\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sim 0^\circ$</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$0^\circ \leq \gamma \leq 90^\circ$</td>
<td>$0 \leq \cos(\gamma) \leq 1$</td>
<td>$0.5 \leq \mu \leq 1$</td>
<td></td>
</tr>
<tr>
<td>$\gamma = 90^\circ$</td>
<td>0</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>$90^\circ \leq \gamma \leq 180^\circ$</td>
<td>$-1 \leq \cos(\gamma) \leq 0$</td>
<td>$0 \leq \mu \leq 0.5$</td>
<td></td>
</tr>
<tr>
<td>$\gamma = 180^\circ$</td>
<td>-1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.4 Calculated MU with Four different slopes \((m_\alpha)\), smallest value of slope = Blue, Largest value of slope = Violet.

4.2.2.3 Studies of the relationship between settings for both parameters

The previous sections described the definition and impact of each of the two control parameters for the double SLERP function. Linear equations were involved in both calculations to accelerate the decrease of the parameter values when the limitations for each of the two sensors becomes increasingly significant. Small variations in the values of the slopes of those linear functions affect the overall output of the algorithm. This section shows the study of the behavior of the algorithm as these two slopes, \(m_\alpha\) and \(m_m\), are varied in the processing of recordings that include four possible scenarios for the sensor module: rotation in an area without magnetic distortion, rotation in an area with magnetic distortion, translation in an area without magnetic distortion, and translation in an area with magnetic distortion. The experiment investigated three levels for both slopes: low \((m_\alpha, m_m = 1)\), mid \((m_\alpha, m_m = 5)\), and high \((m_\alpha, m_m = 10)\), yielding nine cases for each scenario, as shown in Figure 4.5(a-d). The green boxes indicate the settings that shows acceptable outputs of the Double SLERP algorithm for each case.
Figure 4.5 All the panels display the evolution of the output quaternion, $q_{OUT}$, of the Single SLERP vs Double SLERP with varying Slope $m_a$ and $m_\mu$ at 4 Scenarios (a) Rotation without Magnetic Distortion, (b) Rotation with Magnetic Distortion, (c) Translation without Magnetic Distortion, and (d) Translation with Magnetic Distortion.

By analyzing all possible outcomes with different slope settings, the slope for alpha ($m_a$) was set as 1, while the slope for mu ($m_\mu$) was varied between 1 and 5. Therefore, another experiment was performed to test with a fine-tuning value of slope ($m_\mu$)
running from 1 to 5 with 1 as increment. The set of slope values for both $\alpha$ and $\mu$ were then set finally as $m_a=1$ and $m_m=2$, which yields adequate results in all four scenarios.

4.2.2.4 Voxel partition of the working space

Section 4.2.2.3 mentioned that the IR cameras were used to detect the MARG sensor's locations, which were then mapped within the working space of the system, for storing the values of the calculated $\mu$ parameter. However, attempting to deal with the position coordinates as continuous variables in this process is unnecessary and would slow the performance of the algorithm. Instead, the level of magnetic distortion in different regions of the working space for the system can be evaluated in small discrete sub-regions. This section outlines a method to divide the working space into a 3-dimensional array of sub-regions, which creates an appropriate framework, easily mapped to an adequate data structure, where the position-dependent values of $\mu$ can be stored.

First, the units of the position coordinates output by the camera system must be defined and converted into the units used to map the working space. This research has used the metric system units to split the working space into a cubic grid (unit=cm). Each box in the cubic grid is called a "voxel," which has a specific size in each of its 3 dimensions (width, height, depth); a smaller size would allow a higher resolution mapping, which is desirable. However, the trade-off between resolution and processing time must be taken into account. This research has set the voxel size to be 1 cm. Equation 4.34 shows how a voxel index is assigned from the current coordinate of the marker identified by the camera system, in any of the 3 spatial directions. For example, if the marker position is indicated by a coordinates value between 0 to 1 cm, the marker
is now located at voxel #1 (in that particular dimension), and a coordinate value running between 1 to 2 cm will yield a voxel index of 2.

\[
\text{Location of marker (index)} = \text{floor}\left(\frac{\text{Current Coordinate}}{\text{Voxel Size}}\right) + 1 \quad (4.34)
\]

All voxels are initialized with \( \mu = \text{zero} \). The system will gradually store actual values of \( \mu \) in each voxel only when that specific region of the working space is visited by the system and if the accelerometer-corrections were found to be valid (the sensor is static). In time, more and more voxels will be assigned measured non-zero \( \mu \) values, and a considerable proportion of magnetometer-corrections will also be enabled, as shown in Figure 4.6. Numerically, the system keeps the \( \mu \) values as a 3-D array, \( \mu(x,y,z) \), for all the voxels in the space where the sensor will operate.

\[\text{Figure 4.6 The cubic grid with voxels in the working space. Gray voxels indicate } \mu = \text{zero. White voxels indicate non-zero } \mu, \text{ i.e., no significant magnetic distortion}\]
CHAPTER 5 – SETUP AND IMPLEMENTATION OF THE HAND MOTION TRACKING SYSTEM

In this chapter, the implementation of the hand motion tracking system is described. Two devices were used to set up the system: the infrared cameras and a Magnetic, Angular-rate, Gravity (MARG) module. The infrared cameras determined the position of the hand in three-dimensional space while the MARG was used to detect the hand orientation. The Cartesian coordinates of an infrared-reflective marker attached to the wooden block used for testing were obtained from the infrared cameras. The orientation of the block is captured from the MARG unit attached to it. A MARG module consists of three types of sensors: gyroscope, accelerometer, and magnetometer, each providing tri-axial data. The hand motion tracking system with the proposed algorithm uses the combined data from both infrared cameras and the MARG module. Two specific devices were utilized in this research: for tracking position the OptiTrack V120: Trio infrared camera system was used, and for orientation measurement the Yost Labs 3-Space sensor was used.

5.1 Equipment

5.1.1 Use of OptiTrack V120: Trio for Position Tracking

The infrared camera model V120: Trio is manufactured by the OptiTrack company. Three infrared sensing units are horizontally aligned in a single bar with the size of 23” W x 1.6” H x 2” D, shown in Figure 5.1, which allows for tracking up to 6 degrees of freedom objects. Each image sensor has an image resolution of 640x480 with 120 FPS frame rate surrounded by 26 adjustable brightness infrared LEDs. The OptiTrack V120: Trio operates using only two ports: the power supply of 12 VDC 3A through a power adapter and data transfer via USB 2.0 port. The three cameras
contained in the module are self-contained and pre-calibrated by the manufacturer, and ready to use. Camera lenses capture a field of view of 47 degrees horizontally and 43 degrees vertically with a 3.5mm focal length, giving the visible operation distance of 2 to 17 feet away from the device. The operation volume size of the OptiTrack V120:Trio is displayed in Figure 5.2.

![Figure 5.1 The infrared camera model V120: Trio. From Optrictrack, NaturalPoint, Inc., March 2021 [69].](image)

![Figure 5.2 The operation volume size of the OptiTrack V120: Trio from side view (left) and top view (right). [70]](image)

OptiTrack provides an engineering-grade software called “Motive: Tracker.” This software modifies the camera setting such as broadcasting channel, assign markers or rigid body, adjust the exposure of LEDs, etc. Adjusting the exposure value in the Motive setting helps eliminate an unwanted reflection that appears on the image. OptiTrack also developed a plug-in that allows the linkage of the position of designated
markers obtained from the cameras with a virtual 3D object in Unity. The Motive application establishes the local connection to a NatNet server and streams the tracking coordinate data using the NatNet protocol.

Three IR-reflective dot markers are attached at each side of the holding test box, shown in Figure 5.3. One dot marker is visible to the infrared cameras and appears as a single point in the three-dimensional space, representing the sensor module's position. The LEDs from cameras emit infrared light to reflect on the marker and capture the dot marker using the image-sensing units. Then, the Motive Tracker program computes three-dimensional Cartesian coordinates (x, y, and z) in real-time, as the position of the dot marker. The coordinates x, y, and z will be used to represent the position of the holding test box in a 3D environment and used to calculate the location of the voxels to determine the magnetic distortion area.

![Figure 5.3A wooden box with Infrared-reflective markers.](image)

5.1.2 Use of Yost Lab 3-Space Sensors for Orientation Tracking

The commercial-grade MEMS IMU “3-space Micro USB” from the Yost Labs was chosen for detecting the orientation, (Figure 5.4). This MEMS IMU consists of a
gyroscope, accelerometer, and magnetometer, capable of measuring the module’s rotational motion, module’s acceleration, and Earth’s magnetic field in three-dimensional space, respectively. The Yost Labs 3-space™ sensor Micro USB model is an ultra-miniature, low-cost inertial measurement unit with high precision and high reliability. The sensor costs $65 per unit with 23mm x 23mm x 2.2 mm in size and weight of only 1.3 grams. The sensor was attached to the holding test box to capture the movement. The Yost Labs 3-space™ sensor consumes a current of 45mA at 5V and transfers data using a single port of mini USB2.0 (Asynchronous Serial) connection. This sensor module provides orientation estimates in several data formats, such as Quaternion, Euler angles, Axis angle rotation matrix, and the direct data such as normalized sensor data and calibrated sensor data. The specification of Yost Labs 3-space™ sensor Micro USB is shown in Table 5.1.

![Yost Lab 3-spaceTM sensor Micro USB with dimension](image)

Figure 5.4 Yost Lab 3-spaceTM sensor Micro USB with dimension

Yost labs provided software called “3-space Software Suite” to calibrate the sensor before implementing the proposed algorithm. The sensor is connected to the host PC using a micro-USB to type-A USB cable. To activate the sensor port and obtain the data, specific commands, expressed as strings of ASCII characters [41] have to be sent to the sensor.
A C# script was written in Unity to receive the streamed measurement data via the communication ports, which are the timestamp (4 bytes), sensor’s confidence value (4 bytes), accelerometer data (12 bytes, 4 bytes for each x, y and z axes), angular velocity (12 bytes), and magnetometer data (12 bytes). The rotational movement is obtained from the parsed data of angular velocity streaming and compensated by the proposed correction algorithm using the streaming data from the accelerometer and the magnetometer.

Table 5.1 Specifications of Yost Lab 3-space™ sensor Micro USB [71]

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orientation range</td>
<td>360° about all axes</td>
</tr>
<tr>
<td>Orientation accuracy</td>
<td>±1° for dynamic conditions &amp; all orientations</td>
</tr>
<tr>
<td>Orientation resolution</td>
<td>&lt;0.08°</td>
</tr>
<tr>
<td>Orientation repeatability</td>
<td>0.085° for all orientations</td>
</tr>
<tr>
<td>Accelerometer scale</td>
<td>±2g / ±4g / ±8g selectable for standard models</td>
</tr>
<tr>
<td></td>
<td>±6g / ±12g / ±24g selectable for HH models</td>
</tr>
<tr>
<td></td>
<td>±100g / ±200g / ±400g selectable for H3 models</td>
</tr>
<tr>
<td>Accelerometer resolution</td>
<td>14 bit, 12 bit(HH), 12 bit(H3)</td>
</tr>
<tr>
<td>Accelerometer noise density</td>
<td>99μg/VHz, 650μg/VHz(HH), 15mg/VHz(H3)</td>
</tr>
<tr>
<td>Accelerometer sensitivity</td>
<td>0.00024g/digit-0.00096g/digit</td>
</tr>
<tr>
<td></td>
<td>0.003g/digit-0.012g/digit(HH)</td>
</tr>
<tr>
<td></td>
<td>0.049g/digit-0.195g/digit(H3)</td>
</tr>
<tr>
<td>Accelerometer temperature sensitivity</td>
<td>±0.008%/°C, ±0.01%/°C(HH, H3)</td>
</tr>
<tr>
<td>Gyro scale</td>
<td>±250°/±500°/±1000°/±2000°/sec selectable</td>
</tr>
<tr>
<td>Gyro resolution</td>
<td>16 bit</td>
</tr>
<tr>
<td>Gyro noise density</td>
<td>0.009°/sec/VHz</td>
</tr>
<tr>
<td>Gyro bias stability @ 25°C</td>
<td>2.5°/hr average for all axes</td>
</tr>
<tr>
<td>Gyro sensitivity</td>
<td>0.00833°/sec/digit for ±250°/sec</td>
</tr>
<tr>
<td></td>
<td>0.06667°/sec/digit for ±2000°/sec</td>
</tr>
<tr>
<td>Gyro non-linearity</td>
<td>0.2% full-scale</td>
</tr>
<tr>
<td>Gyro temperature sensitivity</td>
<td>±0.03%/°C</td>
</tr>
<tr>
<td>Compass scale</td>
<td>±0.88 Ga to ±8.1 Ga selectable (±1.3 Ga default)</td>
</tr>
<tr>
<td>Compass resolution</td>
<td>12 bit</td>
</tr>
<tr>
<td>Compass sensitivity</td>
<td>0.73 mGa/digit</td>
</tr>
<tr>
<td>Compass non-linearity</td>
<td>0.1% full-scale</td>
</tr>
</tbody>
</table>
5.1.3 Wooden test box to hold the sensor.

A non-ferromagnetic (wooden) box was created to hold the Yost Labs 3-space™ sensor. The box has dimensions of the 8cm x 4cm x 1cm, and MARG module was attached to the holding box, which also has a handling stick at one end, as shown in Figure 5.5. Each side of the box is perpendicular, which allows its reliable positioning at orientations that differ by 90-degree from each other. The flat surfaces of the box can be placed flush against a level surface with a calibrated horizontal position. Attaching the sensor to the holding box helps to eliminate relative orientation errors that could be introduced by the irregular shape of the hand. If the sensor were attached to glove to be worn by each experimental subject, offset gaps between the hand and the glove and between the fabric of the glove and the sensor could exist, as illustrated in Figure 5.6. Using the box increases the consistency in positioning from one subject to another (the size of the subject’s hand will not introduce an additional factor) and reduces and controls the error that might occur. Therefore, the output from the experiment will vary mainly according to the performance of the different algorithms.

*Figure 5.5 The non-ferromagnetic box with Yost Lab 3-space™ sensor attached.*
5.2 Implementation

In order to evaluate the performance of the motion tracking system, based on the MARG sensor, an experiment with human subjects were conducted (the experiment received FIU Internal Review Board authorization #IRB-19-0110). Thirty human subjects were asked to participate in the experiment. Each of the subjects came to the testing area and performed the movement tasks. During the performance, the locations of the holding box were recorded using the marker coordinates from the OptiTrack V120: Trio system, and the data from a Magnetic, Angular-rate, Gravity (MARG) module attached to the holding box was recorded. The proposed algorithm was implemented to calculate the estimated orientation. The calculated results obtained from the Gravity Vector and Magnetometer with Double SLERP (GMV-D) were compared with the orientation estimates obtained from the Gravity Vector and Magnetometer with Single SLERP (GMV-S), Fixed Bias offset, and the Kalman-based filter quaternion output that was provided directly from the Yost Labs 3-Space sensor module.

5.2.1 Hardware and Environment setup

The experimental space was set up with a wooden frame to prevent the presence of uncontrolled magnetic distortion in the area. On the opposite side, an iron bar (0.5cm...
x 3.8cm x 37.5cm) was placed on a wooden stool on the same level as the non-magnetic distortion area to create a magnetically distort area. There was a docking position “home position” where the holding box would rest before and after each trial. The docking position controlled the initial data that will be used in the algorithm. The OptiTrack V120: Trio camera was placed in front of the experimental area with a PC monitor next to it. Participants followed animated on-screen instructions created in Unity software. The testing area is shown in Figure 5.7. Figure 5.8 compares the real position of the areas with/without magnetic distortion and the voxel path of µ plot in a 3D map, where blue indicates the µ ≥ 0.9 and red represents µ < 0.9.

Figure 5.7 The subject performs the experiment at the testing area.
5.2.2 Software setting

The quaternion-based correction using the gravity vector and magnetic North vector with double SLERP was implemented as depicted in the block diagram shown in Figure 5.9. The Gravity Magnetic Vector using double SLERP (GMV-D) C# script was created to implement the orientation correction, using the Unity plugin package from OptiTrack to determine the position of the sensor. A flowchart showing the algorithm is displayed in Figure 5.10. A virtual 3D environment was also created in the same project with Unity. A 3D rectangle shape was created and indicated the wooden holding box. The C# script was written to pre-defined the movements of the sensor box, which consists of an initial position at the dock area, nine specific orientations (or “poses”) at the non-magnetic distorted area, nine poses at the magnetic distorted area, and the ending position at the dock area. The 3D box model was displayed as the movement guide for the subjects to follow and perform the experiment. The nine poses at both the non-magnetically distorted area and the magnetically distorted area are identical, as shown in Figure 5.11.
Figure 5.9 Block diagram of the orientation correction algorithm using the gravity vector and magnetic North vector with Double SLERP.
Begin

Calculate Average value in three axis of 3-sample window of gyroscope data (gyroAvg)

gyroAvg < gyroThreshold

trigCount=trigCount+1

trigCount==5

Calculate predicted bias offset error ($\hat{h}$) and unbiased angular rate ($\omega_p$) from previous predicted bias offset error

Calculate quaternion ($q_G$)

Calculate Gravity vector $\hat{a}(q_G) = q_G^* \otimes A_{int} \otimes q_G$

Calculate difference in quaternion ($\Delta q_A$)

Calculate magnetic North vector $\hat{m}(q_G) = q_G^* \otimes M_{int} \otimes q_G$

Calculate difference in quaternion ($\Delta q_M$)

A
Figure 5.10 Flowchart showing the implementation of Gravity Magnetic Vector using Double SLERP corrections algorithm for one iteration, where pos.X, pos.Y and pos.Z obtained from OptiTrack’s Unity Plugin.
Figure 5.11 The movement guide animation in Non-magnetic distorted and Magnetic distorted area showing identical rotation.
5.2.3 Experiment Procedure (from the instruction to the subject approved by FIU IRB)

1. Firstly, the subject was asked to stand in the testing area, facing toward the IR camera and a desktop monitor with the holding box placed in front of him/her. Then, the experimenter started the program.

2. The experimenter clicked on the button “Mark this position and orientation” to record the initial position and orientation of the sensor.

3. The experimenter clicked on the button “Show next movement” to show the animation of the 3D box model. The 3D box model rotated or translated to the next state (pose) of the box movement sequence.

4. The subject grasped the box and moved it to match the position and orientation as shown by the movement guide on the screen.

5. The experimenter clicked on the button “Mark this position and orientation” to record the current position and orientation of the held box.

6. Steps 3 to 5 were repeated until the expected 20 states or poses of the movement of the 3D box model had been performed by the subject.

7. The subject was asked to repeat the experiment two times. Then, the subject was asked to answer the simple questionnaire about age, gender, and his/her dominant hand.

Before the experiment began, all participants were asked to read and signed the agreement of the consent to participate in a research study form. This protocol received approval number “IRB-19-0110” on March 29, 2019, from the FIU Internal Review Board.
CHAPTER 6 - EXPERIMENTAL RESULTS

In this chapter, the experiment results are analyzed and described in two ways. Firstly, four types of methods are compared in the time domain by showing the four components of quaternion resulting in graphs and by visualizing the poses of a 3D hand model. Statistical evaluation is explained in the second section where the Fixed Bias method (which only counters the gyroscope bias by always subtracting the same offset as found during initialization) was disregarded due to its particularly poor results, observed in Section 6.1.

6.1 Comparison of quaternion components

To verify the results of the proposed GMV-D algorithm, the corrected orientation was expressed as a quaternion ($\tilde{q}_{out}$ in Figure 5.9) for the GMV-D method and compared with three other methods (Fixed Bias, Kalman Filtering, and GMV-S). Three of the numbers contained in a quaternion (in our case, $\tilde{q}_x$, $\tilde{q}_y$ and $\tilde{q}_z$) represent an axis vector in 3-dimensional space, and the 4th component ($\tilde{q}_w$ in our case), indicates the amount of rotation to be performed [6]. Therefore, the 4 components of the output orientation quaternion were displayed through time in order to analyze the results (Figure 6.1). The panels in Figure 6.1 show the evolution of the 4 quaternion components from the 4 methods, for the complete duration of the experiment (The horizontal axes are labeled in samples, where the sampling interval was 10ms). The first half of the record (left of the vertical green dividing line) corresponds to the rotations that took place in the area that was not magnetically distorted. The second part of the record (to the right of the green line) represents the same sequence of rotations, now performed in the magnetically distorted area. The 4 quaternion components in the top panel (“Fixed Bias”) display drift that grows gradually throughout the experiment,
with and without magnetic distortion. All other methods ("Kalman Filter", "GMV-S", and "GMV-D") performed very similarly when magnetic distortion was not present. However, both the Kalman Filter and GMV-S show erroneous results in the magnetically distorted area. The bottom panel of Figure 6.1 shows the corresponding output quaternion components obtained from the newly proposed GMV-D algorithm, which displays performance that was much less deteriorated when the rotations took place in the magnetically distorted area.

6.2 Orientation Visualizations

The black numerals in Figure 6.1 (1 to 10) help identified short segments during the experiment when the subject was instructed to sustain specific hand orientations or “poses”. The corresponding instructed poses are visualized in 3-D, in Figure 6.2 and Figure 6.3, under “Sequence Reference” (leftmost column). In each pose, the 4 visualizations that are shown to the right of the instructed pose are defined (in Unity) by the quaternion results obtained from “Fixed Bias”, “Kalman Filter”, “GMV-S”, and “GMV-D”, from left to right, respectively. This additional form of visualization shows that the orientations obtained from GMV-D were much closer to the orientations that the subject was instructed to temporarily maintain, particularly when the rotations took place in the magnetically distorted zone (Poses 6 through 10). Figure 6.1, Figure 6.2 and Figure 6.3 confirm that the added processing integrated with GMV-D has made it more resilient to the potential distortion of the magnetic field that may exist in some regions of the working space for the MARG module.
Figure 6.1 The evolution of the 4 quaternion components from 4 orientation estimation methods for a complete duration of experiment with interval sampling rate = 10ms.
<table>
<thead>
<tr>
<th>Sequence Reference</th>
<th>FixBias</th>
<th>Kalman Filter</th>
<th>GMV with Single SLERP</th>
<th>GMV with Double SLERP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
</tr>
<tr>
<td>5</td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
</tbody>
</table>

*Figure 6.2 3D hand rendering oriented according to the orientation estimation output from 4 different methods while in the non-magnetically distorted area (Poses 1-5)*
6.2 Statistical Evaluations

To quantify the performance of the proposed algorithm, thirty human subjects (22 males and 8 females, all right-handed) participated in an experiment. Their ages ranged from 18 to 60 years old. None of the subjects reported any motion impediments who could affect the performance of the evaluation task. The orientations estimated by the methods implemented while the subjects were sustaining specific instructed poses were analyzed statistically. In the area without magnetic distortion, each subject held 9 poses (First, pose 1, then poses 2, 3, 4 and 5, with a return to Pose 1 after each). Similarly, each subject held 9 poses in the magnetically distorted area (Pose 6 and then
7, 8, 9 and 10 with returns to 6). Since the Fixed Bias method was seen to perform very poorly (Figures 6-1, 6-2 and 6-3), the statistical analysis is concentrated only on 3 algorithms: Kalman Filter (KF), GMV-S, and GMV-D. The total of 1620 rows of data (30 subjects x 18 orientations x 3 algorithms) were recorded and statistically analyzed using the SPSS statistical package. The recorded data represent the difference between the reference orientation (instructed to the subjects) and the output from the algorithms, measured in terms of the corresponding Euler Angles (Phi, Theta and Psi, which represent the value of the angles rotated about the x, y and z axes). If the orientation algorithm worked correctly, these differences were expected to be ‘zero’. The estimated means and standard deviation of the orientation errors in all Euler Angles in both areas (with and without magnetic distortion) are shown in the Table 6.1. It is clearly seen that the means and standard deviation of the orientation errors for GMV-D are much less than that of other two methods (GMV-S, KF) in every Euler Angle. Figure 6.4 to Figure 6.6 show the estimated marginal means of the orientation errors for Phi, Theta and Psi from all methods, respectively.

Table 6.1 Estimated means and Standard Deviation of the orientation output. (In degree)

<table>
<thead>
<tr>
<th>Dependent Variable</th>
<th>Algorithm</th>
<th>Phi</th>
<th>Theta</th>
<th>Psi</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMV-D</td>
<td>Mean</td>
<td>1.858</td>
<td>8.231</td>
<td>3.992</td>
</tr>
<tr>
<td></td>
<td>Std. Deviation</td>
<td>2.658</td>
<td>10.818</td>
<td>8.554</td>
</tr>
<tr>
<td>GMV-S</td>
<td>Mean</td>
<td>5.053</td>
<td>38.318</td>
<td>16.104</td>
</tr>
<tr>
<td></td>
<td>Std. Deviation</td>
<td>14.769</td>
<td>52.557</td>
<td>37.527</td>
</tr>
<tr>
<td>KF</td>
<td>Mean</td>
<td>11.065</td>
<td>53.452</td>
<td>17.887</td>
</tr>
<tr>
<td></td>
<td>Std. Deviation</td>
<td>17.071</td>
<td>65.382</td>
<td>40.489</td>
</tr>
<tr>
<td>Total</td>
<td>Mean</td>
<td>5.992</td>
<td>33.334</td>
<td>12.661</td>
</tr>
<tr>
<td></td>
<td>Std. Deviation</td>
<td>13.659</td>
<td>52.302</td>
<td>32.821</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>1800</td>
<td>1800</td>
<td>1800</td>
</tr>
</tbody>
</table>
Figure 6.4 Estimated Marginal means of the orientation errors for Phi (In degrees)

Figure 6.5 Estimated Marginal means of the orientation errors for Theta (In degrees)
The multivariate analysis of variance (MANOVA) was originally chosen to test for the effects of the three algorithms on the orientation output errors. However, the appropriate application of MANOVA analysis requires verification of two key assumptions in the data, which are normality of the error and equal variances across treatments. The null hypothesis for normality test is that the data are normally distributed within each treatment group. The results from the tests of normality (Kolmogorov-Smirnov and Shapiro-Wilk) are shown in Table 6.2 having the p-values of 0.000 for all angles and all methods. These results provided strong evidence that the orientation output errors are not normally distributed, which can be confirmed graphically in the Normal Q-Q plots in Figure 6.7, Figure 6.8, and Figure 6.9, for angles Phi, Theta, and Psi, respectively.
Table 6.2 Tests of Normality: Kolmogorov-Smirnov and Shapiro-Wilk

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Kolmogorov-Smirnov&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Shapiro-Wilk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Statistic</td>
<td>df</td>
</tr>
<tr>
<td>Phi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMV-D</td>
<td>.243</td>
<td>600</td>
</tr>
<tr>
<td>GMV-S</td>
<td>.424</td>
<td>600</td>
</tr>
<tr>
<td>KF</td>
<td>.319</td>
<td>600</td>
</tr>
<tr>
<td>Theta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMV-D</td>
<td>.224</td>
<td>600</td>
</tr>
<tr>
<td>GMV-S</td>
<td>.348</td>
<td>600</td>
</tr>
<tr>
<td>KF</td>
<td>.302</td>
<td>600</td>
</tr>
<tr>
<td>Psi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMV-D</td>
<td>.321</td>
<td>600</td>
</tr>
<tr>
<td>GMV-S</td>
<td>.412</td>
<td>600</td>
</tr>
<tr>
<td>KF</td>
<td>.377</td>
<td>600</td>
</tr>
</tbody>
</table>

<sup>a</sup> Lilliefors Significance Correction

Figure 6.7 The Normal Q-Q plot of Phi
Figure 6.8 The Normal Q-Q plot of Theta

Figure 6.9 The Normal Q-Q plot of Psi
Next, the homogeneity of variances was tested with the null hypothesis that the error variance of the dependent variable is equal across treatment groups, yielding the results shown in the Table 6.3. The p-values = 0.000 from all three dependent variables of the test of homogeneity of variances, provide strong evidence that the error variances are not equal among three treatment groups (GMV-D, GMV-S and KF) which, therefore, results in rejection of the null hypothesis.

Table 6.3: Levene’s Test of Equality of Error Variances.

<table>
<thead>
<tr>
<th>Dependent Variable</th>
<th>Levene Statistic</th>
<th>df1</th>
<th>df2</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phi</td>
<td>Based on Mean</td>
<td>24.062</td>
<td>59</td>
<td>1740</td>
</tr>
<tr>
<td></td>
<td>Based on Median</td>
<td>21.110</td>
<td>59</td>
<td>1740</td>
</tr>
<tr>
<td></td>
<td>Based on Median and with adjusted df</td>
<td>21.110</td>
<td>59</td>
<td>131.746</td>
</tr>
<tr>
<td></td>
<td>Based on trimmed mean</td>
<td>22.299</td>
<td>59</td>
<td>1740</td>
</tr>
<tr>
<td>Theta</td>
<td>Based on Mean</td>
<td>61.970</td>
<td>59</td>
<td>1740</td>
</tr>
<tr>
<td></td>
<td>Based on Median</td>
<td>37.068</td>
<td>59</td>
<td>1740</td>
</tr>
<tr>
<td></td>
<td>Based on Median and with adjusted df</td>
<td>37.068</td>
<td>59</td>
<td>193.641</td>
</tr>
<tr>
<td></td>
<td>Based on trimmed mean</td>
<td>58.456</td>
<td>59</td>
<td>1740</td>
</tr>
<tr>
<td>Psi</td>
<td>Based on Mean</td>
<td>105.301</td>
<td>59</td>
<td>1740</td>
</tr>
<tr>
<td></td>
<td>Based on Median</td>
<td>64.094</td>
<td>59</td>
<td>1740</td>
</tr>
<tr>
<td></td>
<td>Based on Median and with adjusted df</td>
<td>64.094</td>
<td>59</td>
<td>96.072</td>
</tr>
<tr>
<td></td>
<td>Based on trimmed mean</td>
<td>103.664</td>
<td>59</td>
<td>1740</td>
</tr>
</tbody>
</table>

As the MANOVA analysis was found not to be appropriate for the data, the nonparametric Kruskal-Wallis H test was chosen to perform the analysis, since it does not require the assumptions of variance homogeneity and normality of errors [72]. The Kruskal-Wallis is a rank-based nonparametric test, commonly used for determining the statistical significance of the differences of a dependent variable across two or more treatment groups [73]. Each dependent variable (Phi, Theta and Psi) was tested with the Kruskal-Wallis approach at a level of significance of 0.05 to determine if there are
differences in means across the three algorithms (treatments). The analysis was performed separately for the poses held in the area that was not magnetically disturbed and the area that was magnetically disturbed.

For the area which was not magnetically distorted, the summary of results in Table 6.4 shows that the distributions of both Phi and Theta were not significantly different across the three algorithms, as shown in more detail by Figure 6.11 and Figure 6.11, with H=5.478, p=0.065 and H=2.439, p=0.295, respectively. The null hypothesis that the distribution of orientation errors is the same across algorithms was rejected only for the Psi angle (Figure 6.12), for which H(2)=14.586, p=0.001, with a mean rank of 381.70 for GMV-S, 384.93 for GMV-D and 449.86 for KF. Through pairwise comparisons, the results indicate that there are no statistically significant differences of the orientation errors in Psi between GMV-S and GMV-D (p=1.000) while KF shows statistically significant differences of the orientation errors with GMV-S (p=0.002) and GMV-D (p=0.004).
Table 6.4 Summary results of Kruskal-Wallis Test for the orientation errors in all three angles (Phi, Theta, Psi), across three different methods in the non-magnetically distorted area.

### Hypothesis Test Summary

<table>
<thead>
<tr>
<th>Null Hypothesis</th>
<th>Test</th>
<th>Sig.</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The distribution of Phi is the same across categories of Algorithm.</td>
<td>Independent-Samples Kruskal-Wallis Test</td>
<td>.065</td>
<td>Retain the null hypothesis.</td>
</tr>
<tr>
<td>2 The distribution of Theta is the same across categories of Algorithm.</td>
<td>Independent-Samples Kruskal-Wallis Test</td>
<td>.295</td>
<td>Retain the null hypothesis.</td>
</tr>
<tr>
<td>3 The distribution of Psi is the same across categories of Algorithm.</td>
<td>Independent-Samples Kruskal-Wallis Test</td>
<td>.001</td>
<td>Reject the null hypothesis.</td>
</tr>
</tbody>
</table>

Asymptotic significances are displayed. The significance level is .05.
Figure 6.10 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle Phi, across three different methods in the non-magnetically distorted area.

1. The test statistic is adjusted for ties.
2. Multiple comparisons are not performed because the overall test does not show significant differences across samples.
Figure 6.11 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle Theta, across three different methods in the non-magnetically distorted area.
Figure 6.12 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle $\Psi$, across three different methods in the non-magnetically distorted area.
For the area affected by magnetic distortion, Table 6.5 shows the test summary, which leads to rejection of the null hypothesis that the distribution of orientation errors for all the angles (Phi, Theta and Psi) is the same across algorithms.

Figure 6.13 shows the test statistics for the orientation output errors in the Euler Angle Phi, across the three algorithms. In this test the null hypothesis is that the distribution of orientation errors is the same across the algorithms. The results from the test indicate that there is a statistically significant difference between the orientation errors in Phi produced by different algorithms \( (H(2) = 365.929, p = 0.000) \), with a mean rank of 252.20 for GMV-D, 342.62 for GMV-S and 621.69 for KF. Through pairwise comparisons among the three algorithms, it was found that there are statistically significant differences of the orientation errors in Phi between GMV-D and GMV-S \( (p = 0.000) \), between GMV-D and KF \( (p = 0.000) \), and between GMV-S and KF \( (p = 0.000) \). GMV-D shows the best performance, which is 9.539 less than GMV-S and 16.631 less than KF, in standard test statistic value.

Figure 6.14 shows the test statistics for the orientation output errors in the Euler Angle Theta, across the three algorithms. In this test the null hypothesis is that the distribution of orientation errors is the same across the algorithms. The results from the test indicate that there is a statistically significant difference between the orientation errors in Theta produced by the different algorithms \( (H(2) = 377.616, p = 0.000) \), with a mean rank of 197.76 for GMV-D, 432.47 for GMV-S and 586.27 for KF. Through pairwise comparisons among the three algorithms, it was found that there are statistically significant differences of the orientation errors in Theta between GMV-D and GMV-S \( (p = 0.000) \), between GMV-D and KF \( (p = 0.000) \), between GMV-S and KF \( (p = 0.000) \). GMV-D shows the best performance, which is 11.656 less than GMV-S and 19.293 less than KF in standard test statistic value.
Figure 6.15 shows the test statistics for the orientation output errors in the Euler Angle Psi, across the three algorithms. In this test the null hypothesis is that the distribution of orientation errors is the same across the algorithms. The results from the test indicate that there is a statistically significant difference between the orientation errors in Psi for different algorithms ($H(2) = 278.583, p = 0.000$), with a mean rank of 229.84 for GMV-D, 421.93 for GMV-S and 564.73 for KF. Through pairwise comparisons among the three algorithms, it was found that there are statistically significant differences of the orientation errors in Psi between GMV-D and GMV-S ($p = 0.000$), between GMV-D and KF ($p = 0.000$), and between GMV-S and KF ($p = 0.000$). GMV-D shows the best performance, which is 9.539 less than GMV-S and 16.631 less than KF in standard test statistic value.

Table 6.5 Summary results of Kruskal-Wallis Test for the orientation errors in all three angles (Phi, Theta, Psi), across three different methods at the magnetic distortion area

<table>
<thead>
<tr>
<th>Null Hypothesis</th>
<th>Test</th>
<th>Sig</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The distribution of Phi is the same across categories of Algorithm.</td>
<td>Independent-Samples Kruskal-Wallis Test</td>
<td>.000</td>
<td>Reject the null hypothesis.</td>
</tr>
<tr>
<td>2 The distribution of Theta is the same across categories of Algorithm.</td>
<td>Independent-Samples Kruskal-Wallis Test</td>
<td>.000</td>
<td>Reject the null hypothesis.</td>
</tr>
<tr>
<td>3 The distribution of Psi is the same across categories of Algorithm.</td>
<td>Independent-Samples Kruskal-Wallis Test</td>
<td>.000</td>
<td>Reject the null hypothesis.</td>
</tr>
</tbody>
</table>

Asymptotic significances are displayed. The significance level is .05.
Figure 6.13 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle $\Phi$, across three different methods in the magnetically distorted area.
Figure 6.14 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle Theta, across three different methods in the magnetically distorted area.
Figure 6.15 Kruskal-Wallis test statistics results for the orientation errors in the Euler angle $\Psi$, across three different methods in the magnetically distorted area.
CHAPTER 7 - DISCUSSION

The preceding sections, overall, have shown results that match the key intent of the development of the newly proposed algorithm, GMV-D, which was to make the orientation estimation derived from signals of the MARG module more robust in circumstances where distortion of the geomagnetic field exist.

The evaluation procedure was defined in such a way that its first half would take place within a region of space where the geomagnetic field was not distorted (non-magnetically distorted area), whereas the second half took place in the immediate neighborhood of the iron bar, which was known to introduce significant distortion of the magnetic field (magnetically distorted area).

The initial assessment, comprising the plotting of the 4 components of the orientation quaternions found by the 4 methods compared (Fixed Bias, Kalman Filter, GMV-S and the newly proposed GMV-D), was performed for the complete duration of the experimental task, as a whole. Its results are shown in Figure 6.1. However, with the exception of the Fixed Bias approach, which shows progressive degradation in performance that starts immediately after the beginning of the experiment and is found in both the areas (with and without magnetic distortion), the remaining 3 algorithms display very different behaviors for the first and second halves of the experiment. Before the green vertical dividing line Figure 6.1, KF, GMV-S and GMV-D perform in almost identical fashion, which seems to reflect correct orientation estimates for the multiple poses instructed to the subjects in this first half. In contrast, while the poses held after the subject translated the MARG module to the magnetically distorted zone (i.e., after the green dividing line in Figure 6.1) were exactly the same poses as previously executed in the non-magnetically distorted area, the quaternion outputs of KF and GMV-S do not appear similar to their outputs for the first half. This indicates
that significant orientation estimation errors were introduced in the results from KF and GMV-S while at the magnetically distorted area. Only the bottom traces in Figure 6.1, corresponding to the approach proposed in this dissertation (GMV-D) displays traces that are essentially the same in the first and the second half of the experiment, as they were expected to be. It is, however, not easy to perceive the importance of the orientation errors suspected in the outputs from KF and GMV-S through the second half of the experiment, from the plots in Figure 6.1.

The effective importance of the orientation errors is better appreciated when the output quaternions from the 4 methods are used to determine the attitude of a 3D virtual hand in the Unity software package and renderings of that 3D hand are obtained at the times when the subjects were instructed to hold specific poses (Poses 1-5 in the non-magnetically distorted area and Poses 6-10 in the magnetically distorted area).

Figure 6.2 compares the renderings of the 3D hand with its orientation driven by the quaternion results from Fixed Bias, KF, GMV-S and GMV-D, in columns 2, 3, 4, and 5, respectively, at the poses identified with numbers 1-5, in the non-magnetically distorted area. For reference, the first column shows renderings of the 3D hand taking on the reference orientation that the subjects were instructed to execute. It can be seen that, in the non-magnetically distorted area, other than the Fixed Bias method, all remaining orientation estimators did not introduce visually significant errors.

In contrast, Figure 6.3, which displays the 3D hand for Poses 6 to 10, executed in the magnetically distorted area, shows that only the proposed GMV-D method (rightmost column) resulted in 3D hand orientations that are visually perceived as similar to the reference orientations (leftmost column). This confirms the suspected additional robustness to magnetic distortion that was achieved by the GMV-D method.
Figure 6.1, Figure 6.2, and Figure 6.3 were created from orientation results from the completion of one representative experiment, performed by one of the volunteer subjects. However, it was also necessary to evaluate the results accounting for the diversity of trajectories, movement speed, etc. that various human subjects would use in completing the experimental task, as the system is meant to be part of a human-computer interface. To that end, 30 volunteer subjects were asked to perform the experimental protocol and the orientation results generated for all the poses by them were recorded. To investigate the deviations of those recorded orientations from the “instructed orientations” (“ground truth”) and in order to report the results in a more intuitive way, the orientation errors were expressed as Euler Angles, which are the errors around the 3 orthogonal axes of the “body frame” of the MARG module. These angles (Phi, Theta and Psi) are the angles rotated about the x, y, and z axes, and can, therefore, be more readily interpreted than the 4 numerical components of a quaternion. Since the analysis was performed on angular errors, a lower mean value found for a given method than for another implies that the former performed better than the latter.

Proceeding on these bases, the statistical analysis of error for the 3 Euler Angles was performed separately for the poses in the first half of the experimental procedure (in the non-magnetically distorted area) and for the poses in the second half of the experimental procedure (in the magnetically distorted area.) Additionally, only the 3 orientation estimation methods that were seen to perform reasonably well (KF, GMV-S and GMV-D) were included in this second level of analysis.

For the non-magnetically distorted area, Table 6.4 shows that, as expected, the performance of KF, GMV-S and GMV-D without magnetic distortions was very similar. Only the distribution of errors in Psi was found to be significantly different across the 3 methods. A more detailed analysis, summarized in Figure 6.12 indicated
that GMV-S and GMV-D still yielded similar values of their mean ranks (381.70 and 384.93, respectively), while KF had a more different mean rank at 449.86. This is reinforced by the results of pairwise comparisons, which only found significant differences between KF and GMV-S and between KF and GMV-D, but not between GMV-S and GMV-D.

The results from the poses held in the magnetically distorted area, summarized in Table 6.5, are very different. The Kruskal-Wallis test indicated that the null hypothesis of similar level of errors across orientation estimation methods must be rejected, for all the Euler Angles. That means, in brief, that the levels of orientation error, as represented by the errors in the Phi, Theta and Psi Euler Angles was definitely not the same across orientation estimation methods when they were employed in the magnetically distorted area.

Looking into the performance for each specific angle a consistent pattern was found, in which the newly proposed GMV-D algorithm reported the lowest mean rank from the three algorithms, followed by GMV-S and KF recorded the highest mean ranks, with respect to all 3 of the Euler Angles. For all 3 of the Euler Angles, pairwise comparisons confirmed that there were significant differences in error between every possible pair of these 3 methods (KF vs. GMV-S, KF vs. GMV-D and GMV-S vs. GMV-D).

Overall, the statistical results for the errors that were recorded confirm the intuitive perception that is derived from observation of the plots representing the marginal means of orientation errors for Phi, Theta and Psi, in Figure 6.4, Figure 6.5, and Figure 6.6, respectively. Focusing on the second (right) half of these plots, which corresponds to poses held while in the magnetically distorted area, the errors for GMV-
D are typically lowest, with errors from GMV-S at an intermediate level and errors from KF typically being the highest.

Then, it can be summarized that the provisions included in the proposed GMV-D orientation estimation algorithm, for progressive assessment of the level of distortion of the geomagnetic filed in specific regions of the operating space of the MARG module, have resulted in a meaningful performance improvement. The new formulation of GMV-D has made it possible to avoid the introduction of strong orientation errors in its final estimate that judiciously fuses the original orientation assessment based on integration of gyroscope signals with weighted corrections from the accelerometer and magnetometer signals. The addition of magnetic distortion capability, the new definition of a magnetic correction trustworthiness parameter (µ) and the use of two tiers of SLERP correction of the initial dead reckoning orientation estimate appear to have, indeed, enhanced the performance in magnetically disturbed areas beyond that of the previous GMV-S algorithm.

It can be speculated that, while all 3 of the methods compared in detail: KF, GMV-S and GMV-D derived their orientation estimates from all 3 of the sensor modalities (gyroscope, accelerometer and magnetometer), it is the higher level of adaptability in GMV-D that has yielded the best performance. The Kalman Filter internally implemented by the 3-Space Embedded MARG used for the experiments does not have any specific provisions to modify the items in the Kalman Filtering algorithm that weigh the value of the accelerometer or magnetometer corrections (the covariance matrices corresponding to these measurements). Therefore, these covariance matrices were kept constant throughout the experimental process. The GMV-S algorithm includes the assessment of the acceleration correction trustworthiness parameter, α, on a sample-by-sample basis, but it did not include a
similar parameter to represent the magnetic correction trustworthiness, and, therefore, adapts the weighted mixture of accelerometer and magnetometer corrections on the exclusive knowledge of the accelerometer correction trustworthiness. This may allow inappropriately large influence of the magnetometer correction when the accelerometer correction is not highly reliable, even if the MARG module is located in a magnetically disturbed area where that strong involvement of the magnetic correction could lead to orientation errors. Only the GMV-D approach, proposed in this dissertation, performs a concurrent and independent assessment of the trustworthiness of both the accelerometer and the magnetometer corrections and defines its final orientation estimate by fusion of both corrections using trustworthiness measures that are updated on a sample-by-sample basis.

Lastly, it should also be noted that the performance of parallel and separate processing of correction quaternions (Δq_A and Δq_M), which are kept independent and individually accessible until the last steps of each iteration of the GMV-D algorithm, opens up possibilities for manipulations that may extract valuable information from these items towards future enhancements of the algorithm. This is in marked contrast with the ways in which corrective measurement information is blended into the global results in early stages of other information fusion approaches for orientation estimation, such as the Kalman Filter.
CHAPTER 8 – ADAPTING GMV-D FOR NEW MARG MODULES

Today, microelectromechanical system (MEMS) sensors are used in many electronic devices such as medical instruments, smartphones, smartwatches, IoT modules, autonomous vehicles, etc. Allied Market Research reported that in 2018 the global MEMS sensor market was valued at $25.7 million and expected to grow and reach $60.6 million by 2026 [74]. This highly competitive market causes the decline of the selling price for these devices, and puts pressure on manufacturers to enhance their performance, for instance, by offering improvements in key characteristics, such as: accuracy, power consumption, sizes, functions, etc.

Yost Labs is one of the commercial MEMS sensor providers. They name their products in this category as “3-Space Sensors”, offering different members of this line with prices that range from as low as $12.50 (the IC alone) to $205.00 and continue to develop and introduce more products to serve the sensor market. The section below will describe a few 3-Space sensors that may be of interest to continue developing the research presented in this dissertation.

8.1 Specification

In this research, the specific MARG model used for implementation was the 3-Space™ Micro USB. Table 8.1 shows a comparison between the 3-Space™ Micro USB and other models recently introduced by the Yost Labs company, which may be used for future development of this research.
Table 8.1 Three models of 3-Space sensors provided by Yost Labs

<table>
<thead>
<tr>
<th>Model</th>
<th>Micro USB</th>
<th>LX Embedded</th>
<th>Nano IC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photo</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>Dimension</td>
<td>23x23x2.2 mm</td>
<td>16x15x1.7 mm</td>
<td>3.8x5.3x1.1 mm</td>
</tr>
<tr>
<td>Weight</td>
<td>1.3 g</td>
<td>0.9 g</td>
<td>0.01 g</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>45mA @ 5v</td>
<td>22mA @ 3.3V</td>
<td>20mA @3.3V</td>
</tr>
<tr>
<td>Orientation Accuracy</td>
<td>+/- 1 Deg</td>
<td>+/- 1.5 Deg</td>
<td>+/- 1.5 Deg</td>
</tr>
<tr>
<td>Gyro Bias Stability</td>
<td>2.5 Deg/hr</td>
<td>11 Deg/hr</td>
<td>11 Deg/hr</td>
</tr>
<tr>
<td>Filter</td>
<td>Kalman, QCOMP</td>
<td>QGRAD2</td>
<td>QGRAD2</td>
</tr>
<tr>
<td>Confidence value (Stillness): 0x2D</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Price</td>
<td>$65</td>
<td>$37.50-$75</td>
<td>$12.50-$25</td>
</tr>
</tbody>
</table>

8.2 Motionlessness Algorithm

From Table 8.1, it is clear that the new generations of MARG modules, such as Yost Labs’ LX Embedded and Nano IC, provide advantages in terms of smaller sizes, lower weight and lower power consumption, which will make them even more appropriate for their use in an instrumented glove. However, Table 8.1 also indicates that an explicit, user-readable parameter of “Confidence” (which indicates, in a range from 0 to 1, how much the MARG module is currently moving) may not be available in the newer modules from Yost Labs. Similarly, this type of parameter may not be available from MARG modules from other manufacturers.

As the orientation estimation algorithm proposed in this dissertation currently uses the “Confidence” parameter provided by the 3-Space Micro USB module, this section proposes a mechanism to define an equivalent floating-point variable, which will be called “Motionlessness (MTNLNS)”, directly from the 3 types of measurements carried out by all MARG modules.
The confidence factor provided by the Micro USB module has a range of values from 0 to 1, where 1 indicates that the sensor is static [41]. In the proposed orientation estimation algorithm, that confidence factor is used to calculate the value of “alpha”, which defines the weight given to the accelerometer correction ($q_{GA}$).

The “Motionlessness” variable was created to substitute the Confidence Factor as defined below. The new MTNLNS parameter was defined by processing the signals from both the gyroscope axes and the accelerometer axes. Two separate versions, $Gyro_{MTNLNS}(i)$ and $Accel_{MTNLNS}(i)$, were calculated, and each of them was filtered to yield a contribution towards a new formulation of the alpha parameter used in the GMV-D algorithm.

The version of MTNLNS derived from accelerometer measurements is calculated according to Equations 7.1 and 7.2, where $Th_{MTNLNS}$ is the sensitivity control of MTNLNS (default $Th_{MTNLNS} = 0.5$).

$$\Delta Gyro(x, y, z) = |Gyro_x(i) - Gyro_x(i-1)|, |Gyro_y(i) - Gyro_y(i-1)|, |Gyro_z(i) - Gyro_z(i-1)| \quad (7.1)$$

If $\max(\Delta Gyro(i)) \leq Th_{MTNLNS}$:

$$Gyro_{MTNLNS}(i) = 1 - \frac{\max(\Delta Gyro(i))}{Th_{MTNLNS}} \quad (7.2)$$

Otherwise:

$$Gyro_{MTNLNS}(i) = 0$$

Once the gyroscope version of motionlessness has been calculated, then the Gamma filter is applied to smoothen the signal and obtain the alpha value from Gyroscope ($\alpha_{Gyro}$).

Similarly, to retrieve the alpha value from accelerometer ($\alpha_{Acc}$), the same process is applied to the data from accelerometer (x, y, z). That is $Accel_{MTNLNS}(i)$ is calculated by equations like 7.1 and 7.2, but using accelerometer readings, instead. $Accel_{MTNLNS}(i)$ is also processed through the Gamma filter to yield $\alpha_{Acc}$. 

101
After the Alpha value from both accelerometer and gyroscope were obtained, as shown in Figure 8.1, the possibilities of fusing them as an average or by multiplying their values, i.e., finding the product of their two values at every sampling instant were considered. For the product method, it was found that the resulting overall alpha value would more closely approach the alpha derived from the “confidence” (stillness) internally calculated by the Micro USB module if an offset of 0.1 is applied to the product of the individual alphas. Figure 8.2 shows that the alpha obtained by the product method with 0.1 offset displays very similar trends as the alpha calculated from the “stillness” (“confidence”) parameter offered by the Micro USB module. In fact, the alpha from the product method tends to be more “conservative” in indicating a static condition for the MARG module, which will probably result in increased reliability of the accelerometer corrections, when they are fully enabled by large values of alpha.
Figure 8.1: Alpha value obtained from Gyroscope data (top), and Accelerometer data (bottom).
Figure 8.2 Comparison between Stillness vs Motionlessness with Average method (top) and after applied offset product (bottom)
As a further means to compare the alpha values calculated from the “confidence” or “stillness” parameter provided by the Micro USB MARG module and the alpha that can be calculated as average or product of alphas derived from the Gyro\textsubscript{MTNLNS} and the Accel\textsubscript{MTNLNS}, Figure 8.3 shows the intervals at which the 3 versions of alpha surpass 0.9, in reaction to the gyroscope and accelerometer signals displayed in Figure 8.1. For the alpha derived from the “stillness” parameter the intervals where its value surpassed 0.9 are drawn at a height of 1 according to the scale in the right margin of the figure. For the alpha variables defined as average and product of $\alpha_{\text{Gyro}}$ and $\alpha_{\text{Acc}}$, the intervals above the 0.9 threshold have been drawn at a height of 1 but following the (larger) scale that appears on the left margin of the figure (This was intended to make the traces more easily discernable). This figure confirms that the average combination of alphas and (particularly) the product combination of alphas is actually slightly more conservative than the alpha from the “stillness parameter.
Figure 8.3: Square wave pulse showing on/off Accelerometer Correction periods
CHAPTER 9 - CONCLUSION AND FUTURE WORK

9.1 Conclusion

This research aimed to create a robust system that can determine, in real-time, the correct orientation of a MEMS MARG module so that multiple MARG modules can be embedded in an instrumented glove that will report the real-time position, orientation, and configuration of the hand of a computer user, enabling new avenues for hand-gesture-based human-computer interaction.

When miniature MEMS accelerometers and gyroscopes were first introduced in the late 1990s and early 2000s, there were hopes that these devices could be used for determining orientation and position in ways similar to the use of their large-scale counterparts for aircraft navigation. However, it was soon found that the much poorer performance characteristics of the MEMS sensors prevented the direct use of the same processing approaches as used for larger devices. In particular, MEMS gyroscopes have been found to have significant and varying levels of “offset,” i.e., non-zero output that is generated when the sensor is actually not turning. As the output of the gyroscopes must be integrated to calculate orientation, even small levels of gyroscope offset will cause high levels of “drift” error in the orientation estimation. Further, this error will tend to rapidly and linearly grow with respect to time. Designers have sought to apply frequent corrections to the orientation calculated from the signals of the gyroscope using information from the accelerometer that usually accompanied the MEMS gyroscopes and even from MEMS magnetometers which began to be included in the sensor packages which were then called Magnetic, Angular-Rate, Gravity (MARG) Sensors. However, accelerometer corrections should only be applied when the module is static so that the accelerometer reports only the acceleration of gravity, and
magnetometer corrections should only be applied if the local geomagnetic field measured by the device is not distorted due to the presence of a nearby medium or large ferromagnetic objects.

This dissertation pursued the definition and evaluation of a novel processing approach that could achieve real-time robust orientation estimation for a typical MARG module in the context of human-computer interaction. This context implies that a 3-camera IR-video system can be used to determine the approximate position of the MARG module, which allowed the novel idea of mapping the level of magnetic trustworthiness (encoded in a parameter $0 < \mu < 1$) of small regions of the working space of the device. This is used to reduce the weight given to the magnetic correction component where the magnetic field is distorted, enhancing the robustness of the system.

At the beginning of the work reported in this dissertation, the first research question asked if the gyroscope drift artifact could be controlled by performing appropriate corrections using information from the accelerometer and magnetometer. The results in Chapter 6 indicate that the proposed algorithm, GMV-D, was successful in avoiding orientation errors to a large degree (more than two alternative methods).

The second research question for this dissertation asked whether it would be possible to spatially detect (“map”) the spatial regions where the geomagnetic field might be distorted to properly decrease the level of involvement of the magnetic correction in the definition of the final orientation estimate. Chapter 4 in this dissertation details the process that has been proposed to map the magnetic trustworthiness parameter, $\mu$, in the regions of space visited by the MARG module. It also explains how the knowledge of that mapping is used to adjust the two-tier SLERP interpolation that defines the final orientation estimate of the GMV-D algorithm.
The last research question in this dissertation asked if a system could be developed that would bring together information from all the sensor modalities in the MARG module to result in a robust estimate of its orientation, even in areas where the geomagnetic field might be distorted. This dissertation has proposed the GMV-D algorithm, explained in Chapter 4, and it has benchmarked its performance against its precursor approach, GMV-S, and a commonly used solution, the Kalman Filter. The results shown in Chapter 6 indicate that, indeed, the GMV-D method was more successful than the other two methods in providing robust orientation estimates, even in the region of space in which a significant magnetic field distortion was present.

The improved real-time orientation estimation achieved by GMV-D may facilitate the development of alternative input methods for human-computer interaction, towards the development of more flexible input devices, and also for the tracking of hand movements in 3D virtual environments, which are becoming more and more popular. The development of hand motion tracking benefits the users as it provides a more natural way to interact with computers.

Two distinct advantages of the GMV-D approach are, first that it does not require the user to set any initialization parameters (as other approaches, such as the Kalman Filter require), and second, that it implements processing pipelines where the information derived from the three sensing modalities available in the MARG module (gyroscope, accelerometer, and magnetometer) are kept independent and separately accessible throughout most of each algorithm iteration. This is in contrast to other approaches which might “fuse” or “mix” the several sources of information early-on in the algorithm. This parallel management of the information from the sensors allows a more explicit comparison between them or even a retrospective analysis for each of
them, such that additional mechanisms for suppressing the influence of a sensing modality that is displaying incongruent behavior may be implemented.

In summary, it is likely that the contributions of the GMV-D method may help in making the MARG orientation estimation process more robust by fully taking advantage of the MARG operating conditions for a typical human-computer interaction application and comprehensively utilizing all the sensing modalities available in the MARG module.

9.2 Future Work

The parallel processing pipelines that keep the information from the accelerometer and magnetometer independent throughout each iteration of the GMV-D algorithm may provide an alternative mechanism to detect when the local magnetic field is likely distorted, based exclusively on the internal signals from the MARG module. In the current version of GMV-D, the local magnetic trustworthiness parameter, $\mu$, is defined by “mapping” the regions of space that have been previously visited by the MARG module. The “mapping” process makes use of the position estimates provided by the IR-camera system. This is completely acceptable for the intended use of the MARG module considered in this dissertation (human-computer interaction). However, finding an alternative mechanism for the detection of magnetic field alterations, utilizing only the signals generated by the MARG module, would extend the scope of use of the algorithm. This would be very significant, for example, if the GMV-D orientation estimation method could also be used in ambulatory applications (such as gait analysis) where position estimates may not be readily available.
Similarly, the accuracy level reached by the GMV-D method may be increased if a higher-level management of the trustworthiness parameters, \( \alpha \) and \( \mu \), is implemented dynamically. For example, the current system updates the \( \mu \) value for any visited voxel by merely replacing the value with the newest calculation of \( \mu \). This was implemented so that, even if the ferromagnetic objects in the working space of the system were to move slowly, the system would keep an updated \( \mu \) map that can reflect those slow variations. To enhance the accuracy of the system, however, it may be beneficial to keep track of the last few (e.g., 3 or 5) values of \( \mu \) calculated for a given voxel, assigning the mean of those few values in the update of the voxel.

The overall accuracy level of the GMV-D method might be increased if a progressive, nonlinear “alpha degradation” process is in place to restrict the accelerometer corrections to only those cases with a high level of acceleration trustworthiness, \( \alpha \), in the later intervals of operation of the system. Initially, all the voxels are initialized with \( \mu = 0 \), and, therefore, no significant corrections are made on the basis of the magnetometer measurements. This means that, in the initial intervals of operation of the system, the only significant corrections will be made on the basis of the accelerometer measurements, in proportion to \( \alpha \). Therefore, it is important that the “gaps” between those meaningful corrections be kept small (to prevent the uncontrolled growth of the gyroscopic drift error). In contrast, during the later phases of operation of the system, the commonly visited voxels around the user will be mapped, and for many of them, the \( \mu \) value assigned will be high. Therefore, significant corrections will be performed more frequently based on both accelerometer and magnetometer readings. In these later stages of operation, a mechanism could be put in place to “degrade” to very low levels (close to 0) any values of \( \alpha \) that are not above a
dynamically changing threshold ($\alpha_{\text{DEGRADE}}$), so that, for these late iterations in the operation of the system significant accelerometer corrections will only be applied for cases where it is highly likely that the module is really very close to static, enhancing the accuracy of those corrections. Initially ($\alpha_{\text{DEGRADE}}$) can be set at a low value so that, essentially, $\alpha$ is never “degraded,” and GMV-D will perform as described in this dissertation. When the system has been operating for a while, and more frequent correct magnetic corrections are already taking place, a larger value of $\alpha_{\text{DEGRADE}}$ will make it possible to only apply significant accelerometer corrections when the system is highly confident that the assumption of a static module is closely fulfilled. This is a luxury that may not be afforded at the beginning of the operation of the system since significant magnetic corrections will likely be infrequent right after the initialization of the system.
BIBLIOGRAPHY


[58] O-larnnithipong, N. *Hand motion tracking system using inertial measurement units and infrared cameras* Retrieved from https://digitalcommons.fiu.edu/etd/3905


Appendix A

Source Codes of The Implementation of Orientation Correction Algorithm using Gravity Vector and Magnetic North Vector with Double SLERP
using UnityEngine;
using System;
using System.Collections;
using System.IO.Ports;
using UnityEngine-editor;
using System.Text;
using System.Collections.Generic;

public class StreamRaw1MovementGuide : MonoBehaviour
{
    private IEnumerator coroutine;
    // Connect to the serial port the 3-Space Sensor is connected to
    public static SerialPort sp0 = new SerialPort(\\.\\COM3", 115200,
    Parity.None, 8, StopBits.One);

    // Print-to-file variables
    public string filenameNo = "";
    private static string FileStr0;
    private static string FileStr1;
    private static string FileStrMark;
    private int rotCount = 0;
    private int markCount = 0;
    private bool handRotated = true;
    private float k = 0.0f;
    private GUIStyle guiStyle = new GUIStyle();
    List<Quaternion> Qref = new List<Quaternion>();
    List<Vector3> uPref = new List<Vector3>();
    List<Vector3> Pref = new List<Vector3>();

    // Command packet for getting the filtered tared orientation as a quaternion
    // {header byte, command byte, [data bytes], checksum byte}
    // checksum = (command byte + data bytes) % 256

    public static byte TSS_START_BYTE = 0xF7;
    public static byte TSS_SET_STREAMING_SLOTS = 0x50;
    public static byte TSS_SET_STREAMING_TIMING = 0x52;
    public static byte TSS_START_STREAMING = 0x55;
    public static byte TSS_STOP_STREAMING = 0x56;
    public static byte TSS_GET_SENSOR_MOTION = 0x2D;
    public static byte TSS_GET_RAD_PER_SEC_GYROSCOPE = 0x26;
    public static byte TSS_GET_CORRECTED_LINEAR_ACC_AND_GRAVITY = 0x27;
    public static byte TSS_GET_CORRECTED_COMPASS = 0x28;
    public static byte TSS_GET_TARED_ORIENTATION_AS_QUAT = 0x00;
    public static byte TSS_NULL = 0xFF;
    public static byte TSS_TARE_CURRENT_ORIENTATION = 0x60;
    public static byte CHECK_SUM = (byte)((TSS_SET_STREAMING_SLOTS +
    TSS_GET_SENSOR_MOTION + TSS_GET_RAD_PER_SEC_GYROSCOPE +
    TSS_GET_CORRECTED_LINEAR_ACC_AND_GRAVITY +
    TSS_GET_CORRECTED_COMPASS + TSS_GET_TARED_ORIENTATION_AS_QUAT + TSS_NULL
    + TSS_NULL + TSS_NULL) % 256);

    public static byte[] stream_slots_bytes = {TSS_START_BYTE,
    TSS_SET_STREAMING_SLOTS,
    TSS_GET_SENSOR_MOTION, // Slot0 - 4
    TSS_GET_RAD_PER_SEC_GYROSCOPE, // Slot1 - 12
    TSS_GET_CORRECTED_LINEAR_ACC_AND_GRAVITY, // Slot2 - 12
    TSS_GET_CORRECTED_COMPASS, // Slot3 - 12

    TSS_NULL, TSS_NULL, TSS_NULL, TSS_NULL, TSS_NULL, TSS_NULL,
    TSS_NULL, TSS_NULL, TSS_NULL, TSS_NULL, TSS_NULL, TSS_NULL};
private double CurrentTime = 0.00;
private double PreviousTime = 0.00;
private float SamplingTime;
private int N = 0;
private static int BuffSize = 3;
private float B_SCALING = 1.0f;
private float alpWeight = 0.25f;
private float muWeight = 0.25f;

private float Stillness0;
private Vector3 Gyro0, Accelero0, Magneto0;
private Quaternion IMUQuat0;
private Vector3[] GyroBuff0 = new Vector3[BuffSize];
private Vector3[] AcceleroBuff0 = new Vector3[BuffSize];
private Vector3[] MagnetoBuff0 = new Vector3[BuffSize];
private float[] StillnessBuff0 = new float[BuffSize];
private Vector3 GyroAvg0 = new Vector3();
private Vector3 AcceleroAvg0 = new Vector3();
private Vector3 MagnetoAvg0 = new Vector3();
private float StillnessAvg0 = new float();
private float alpha0 = new float();
private float prevAlphaX0 = 1.0f;
private float thisAlphaX0 = new float();
private float prevAlphaY0 = 1.0f;
private float thisAlphaY0 = new float();
private Vector3 Bias0 = new Vector3(0, 0, 0);
private Vector3 fixedBias0 = new Vector3(-0.008081f, -0.002751f, -0.008184f);

private Vector3 BiasBuff0 = new Vector3(0, 0, 0);
private int trigcount0 = 0;
private Vector3 UnbiasedGyro0 = new Vector3(0, 0, 0);
private Vector3 UnbiasedGyroWithfixedBias0 = new Vector3(0, 0, 0);
private Quaternion w0; // Pure Quaternion
private Quaternion wfixed0; // Pure Quaternion
private Quaternion dqG0;
private Quaternion dqGfixed0;
private Quaternion qG0 = new Quaternion(0, 0, 0, 1);
private Quaternion qGfixed0 = new Quaternion(0, 0, 0, 1);
private Quaternion dqGA0;
private Quaternion qGA0 = new Quaternion(0, 0, 0, 1);
private Quaternion dqGM0;
private Quaternion qGM0 = new Quaternion(0, 0, 0, 1);
private Quaternion A_int0 = new Quaternion();
private Quaternion M_int0 = new Quaternion();
private Quaternion a40 = new Quaternion(0, 0, 0, 1);
private Vector3 a30 = new Vector3();
private Quaternion m40 = new Quaternion(0, 0, 0, 1);
private Quaternion m30 = new Vector3();
public static Quaternion qOUT0 = new Quaternion(0, 0, 0, 1);
public static Quaternion qOUT1 = new Quaternion(0, 0, 0, 1);

private Vector3 EulerFixed0;
private Vector3 EulerKalman0;
private Vector3 EulerGMV1;
private Vector3 EulerGMV0;

private Quaternion qGfixed0e;
private Quaternion IMUQuat0e;
public static Quaternion qOUT0e;
public static Quaternion qOUT1e;

private Vector3 EulerFixed0e;
private Vector3 EulerKalman0e;
private Vector3 EulerGMV0e;
private Vector3 EulerGMV1e;

cpyGfixed0e

private Vector3 PosE;

private static int VoxNO = 200;
private static float[,] MU = new float[VoxNO, VoxNO, VoxNO];

float TA = 0.80f; //Alpha Threshold
float TM = 0.80f; //MU Threshold

Quaternion qGpost, mA4;
Vector3 mA3;
float cosGamma, norm_MagAvg, norm_mA3, temp_mut, gg, gl;
static float thisMut = 0.0f, prevMut = 0.0f, thisTempMut = 0.0f, prevTempMut = 0.0f, mut0 = 0.0f;
int Stage;
int ma = 1;
int mt = 2;

static int LocateVoxX = 0;
static int LocateVoxY = 0;
static int LocateVoxZ = 0;

float prevMutX = 0.0f;
float prevMutY = 0.0f;

string[] step = { "Start at initial position", "Move to position 1", "Move to position 2", "Move to initial position" };
void Start()
{
    /*
    private static string FileParameter;
    private static string FileEUL;
    */
    FileStr0 = "Assets/Raw1Recordings/rec" + filenameNo + "GMV0.txt";
    FileStr1 = "Assets/Raw1Recordings/rec" + filenameNo + "GMV1.txt";
    FileStrMark = "Assets/Raw1Marks/mark" + filenameNo + ".txt";

    // Start box at dock.
Qref.Add(Quaternion.Euler(0, 0, 0));  // Move hand to Pos1
// Non mag distortion
Qref.Add(Quaternion.Euler(0, 0, 0));
Qref.Add(Quaternion.Euler(0, 0, 90));
Qref.Add(Quaternion.Euler(0, 0, 0));
Qref.Add(Quaternion.Euler(90, 0, 0));
Qref.Add(Quaternion.Euler(0, 0, 0));
Qref.Add(Quaternion.Euler(0, 90, 0));  // Check y
Qref.Add(Quaternion.Euler(0, 0, 0));
Qref.Add(Quaternion.Euler(90, 0, 45)); // Pos2
// mag distortion
Qref.Add(Quaternion.Euler(0, 0, 0));
Qref.Add(Quaternion.Euler(0, 0, 90));
Qref.Add(Quaternion.Euler(0, 0, 0));
Qref.Add(Quaternion.Euler(90, 0, 0));
Qref.Add(Quaternion.Euler(0, 0, 0));
Qref.Add(Quaternion.Euler(0, 90, 0));  // Check y
Qref.Add(Quaternion.Euler(0, 0, 0));
Qref.Add(Quaternion.Euler(90, 0, 45)); // Dock
Qref.Add(Quaternion.Euler(0, 0, 0));

// Set the read/write timeouts
sp0.WriteTimeout = 500;
sp0.ReadTimeout = 500;

if (!sp0.IsOpen)
{
    try
    {
        sp0.Open();
        print("Serial Port 0 is open (COM4)");
    }
    catch (TimeoutException)
    {
    }
}
else
{
    Debug.LogError("All Serial Ports are already open.");
}

sp0.Write(stream_slots_bytes, 0, stream_slots_bytes.Length);
Array.Reverse(interval);
Array.Reverse(delay);
Array.Reverse(duration);

stream_timing_bytes[0] = TSS_START_BYTE;
stream_timing_bytes[1] = TSS_SET_STREAMING_TIMING;
interval.CopyTo(stream_timing_bytes, 2);
delay.CopyTo(stream_timing_bytes, 6);
duration.CopyTo(stream_timing_bytes, 10);
stream_timing_bytes[14] = (byte)((stream_timing_bytes[1] +
stream_timing_bytes[12] + stream_timing_bytes[13]) % 256);

sp0.Write(stream_timing_bytes, 0, stream_timing_bytes.Length);
tareSensor();
start_stream_bytes[0] = TSS_START_BYTE;
start_stream_bytes[1] = TSS_START_STREAMING;
start_stream_bytes[2] = TSS_START_STREAMING;
sp0.Write(start_stream_bytes, 0, start_stream_bytes.Length);

Update()
{
    if (!EndOfSimulation)
    {
        // A quaternion consists of 4 floats which is 16 bytes
        byte[] read_bytes0 = new byte[56]; // <---------------- No. of Bytes
        // Mono, for some reason, seems to randomly fail on the first read after a
        // write so we must loop
        // through to make sure the bytes are read and Mono also seems not to always
        // read the amount asked
        // so we must also read one byte at a time
        int read_counter = 100;
        int byte_idx0 = 0;

        PreviousTime = CurrentTime;
        CurrentTime += 1 * Time.deltaTime;
        SamplingTime = (float)(CurrentTime - PreviousTime);

        while (read_counter > 0)
        {
            try
            {
                byte_idx0 += sp0.Read(read_bytes0, byte_idx0, 1);
            }
            catch
            {
                // Failed to read from serial port
            }
            if (byte_idx0 == 56)
            {
                // <----------- No. of Bytes
                break;
            }
            if (read_counter <= 0)
            {
                throw new System.Exception("Failed to read quaternion from
port too many times." +
                " This could mean the port is not open or the Mono
serial read is not responding.");
            }
        }
    }
}
--read_counter;

// Convert bytes to floats
Stillness0 = bytesToFloat(read_bytes0, 0);
Gyro0.x = bytesToFloat(read_bytes0, 4);
Gyro0.y = bytesToFloat(read_bytes0, 8);
Gyro0.z = bytesToFloat(read_bytes0, 12);
Accelero0.x = bytesToFloat(read_bytes0, 16);
Accelero0.y = bytesToFloat(read_bytes0, 20);
Accelero0.z = bytesToFloat(read_bytes0, 24);
Magneto0.x = bytesToFloat(read_bytes0, 28);
Magneto0.y = bytesToFloat(read_bytes0, 32);
Magneto0.z = bytesToFloat(read_bytes0, 36);
IMUQuat0.x = bytesToFloat(read_bytes0, 40);
IMUQuat0.y = bytesToFloat(read_bytes0, 44);
IMUQuat0.z = bytesToFloat(read_bytes0, 48);
IMUQuat0.w = bytesToFloat(read_bytes0, 52);

// Orientation Correction Algorithm using gravity vector and magnetic North vector correction.
// --- Code starts here ---!
GMVD();

if (Input.GetKey("escape"))
{
    sp0.Close();
    print("All ports are closed!");
    EndOfSimulation = true;
}

if (rotCount != 0)
{
    HandOrientation = Quaternion.Slerp(Qref[rotCount - 1], Qref[rotCount], k);
    // HandPosition = Vector3.Lerp(uPref[rotCount - 1], uPref[rotCount], k);
    if (k < 1)
    {
        k += 0.05f; // Set speed rotation animation
    }
} else
{
    HandOrientation = Qref[rotCount];
    // HandPosition = uPref[rotCount];
}

//print ("x=" + HandOrientation.x + " y=" + HandOrientation.y + " z=" + HandOrientation.z + " w=" + HandOrientation.w);
this.transform.rotation = HandOrientation;
// this.transform.position = HandPosition;

// Helper function for taking the bytes read from the 3-Space Sensor and converting them into a float
float bytesToFloat(byte[] raw_bytes, int offset)
{
    byte[] big_bytes = new byte[4];
    big_bytes[0] = raw_bytes[offset + 3];
    big_bytes[1] = raw_bytes[offset + 2];
big_bytes[2] = raw_bytes[offset + 1];
big_bytes[3] = raw_bytes[offset + 0];
return BitConverter.ToSingle(big_bytes, 0);
}

void tareSensor()
{
    sp0.Write(tare_bytes, 0, 3);
}

Quaternion myQuatConj(Quaternion q)
{
    Quaternion q_result = new Quaternion(-1.0f * q.x, -1.0f * q.y, -1.0f * q.z, q.w);
    return q_result;
}

Quaternion myQuatIntegrate(Quaternion dq, Quaternion q, float dt)
{
    Quaternion omega = dq * myQuatConj(q);
    omega = new Quaternion(2.0f * omega.x, 2.0f * omega.y, 2.0f * omega.z, 2.0f * omega.w);
    omega = new Quaternion((omega.x * dt) / 2.0f, (omega.y * dt) / 2.0f, (omega.z * dt) / 2.0f, (omega.w * dt) / 2.0f);
    float omega_norm2 = Mathf.Sqrt(Mathf.Pow(omega.x, 2) + Mathf.Pow(omega.y, 2) + Mathf.Pow(omega.z, 2));
    Quaternion exp = new Quaternion();
    if (omega_norm2 != 0)
    {
        exp.x = Mathf.Exp(omega.w) * (Mathf.Sin(omega_norm2) / omega_norm2) * omega.x;
        exp.y = Mathf.Exp(omega.w) * (Mathf.Sin(omega_norm2) / omega_norm2) * omega.y;
        exp.z = Mathf.Exp(omega.w) * (Mathf.Sin(omega_norm2) / omega_norm2) * omega.z;
        exp.w = Mathf.Exp(omega.w) * Mathf.Cos(omega_norm2);
    }
    else
    {
        exp.x = Mathf.Exp(omega.w) * omega.x;
        exp.y = Mathf.Exp(omega.w) * omega.y;
        exp.z = Mathf.Exp(omega.w) * omega.z;
        exp.w = Mathf.Exp(omega.w) * Mathf.Cos(omega_norm2);
    }
    Quaternion q_result = exp * q;
    return q_result;
}

Quaternion myQuatNormalize(Quaternion q)
{
    float q_norm = Mathf.Sqrt((Mathf.Pow(q.x, 2) + Mathf.Pow(q.y, 2) + Mathf.Pow(q.z, 2) + Mathf.Pow(q.w, 2)));
}
Quaternion q_result = new Quaternion(q.x / q_norm, q.y / q_norm, q.z / q_norm, q.w / q_norm);

return q_result;
}

void OnGUI()
{

GUIStyle buttonGUIStyle = new GUIStyle("button");
buttonGUIStyle.normal.textColor = Color.green;
buttonGUIStyle.hover.textColor = Color.green;
buttonGUIStyle.fontStyle = FontStyle.Bold;

GUILayout.BeginArea(new Rect(10, Screen.height - 40, 200, 200));
GUILayout.BeginVertical("box");
if (GUILayout.Button("Close ALL Ports"))
{
  sp0.Close();
  print("COM4 is closed!");
  EndOfSimulation = true;
  UnityEditor.EditorApplication.isPlaying = false;
}
GUILayout.EndVertical();
GUILayout.EndArea();

GUILayout.BeginArea(new Rect(Screen.width / 2 - 100, 40, 300, 200));
if (!EndOfSimulation)
{
  // GUILayout.Label("Now Recording... Subject" + filenameNo + ": markCount.ToString() + "/" + Qref.Count.ToString() + " orientation(s) marked");
}
else
{
  // GUILayout.Label("Recording Completed for Subject" + filenameNo);
}
GUILayout.EndArea();
if (markCount < Qref.Count)
{
  if (handRotated)
  {
    GUILayout.BeginArea(new Rect(Screen.width - 210, Screen.height - 40, 200, 200));
    GUILayout.BeginVertical("box");
    if (GUILayout.Button("Mark this position & orientation"))
    {
      PosE.x = OptitrackStreamingClient.markerX;
      PosE.y = OptitrackStreamingClient.markerY;
      PosE.z = OptitrackStreamingClient.marker2;

      qGfixed0e = myQuatNormalize(myQuatConj(Qref[rotCount]) * qGfixed0);
      IMUQuat0e = myQuatNormalize(myQuatConj(Qref[rotCount]) * IMUQuat0);
      qOUT0e = myQuatNormalize(myQuatConj(Qref[rotCount]) * qOUT0);
      qOUT1e = myQuatNormalize(myQuatConj(Qref[rotCount]) * qOUT1);

      EulerFixed0e = qGfixed0e.eulerAngles;
      EulerKalman0e = IMUQuat0e.eulerAngles;
    }
    GUILayout.EndVertical();
    GUILayout.EndArea();
  }
}
EulerGMV0e = qOUT0e.eulerAngles;
EulerGMV1e = qOUT1e.eulerAngles;

if (EulerFixed0e.x > 180.0)
    EulerFixed0e.x = -(360.0f - EulerFixed0e.x);
if (EulerFixed0e.y > 180.0)
    EulerFixed0e.y = -(360.0f - EulerFixed0e.y);
if (EulerFixed0e.z > 180.0)
    EulerFixed0e.z = -(360.0f - EulerFixed0e.z);

if (EulerKalman0e.x > 180.0)
    EulerKalman0e.x = -(360.0f - EulerKalman0e.x);
if (EulerKalman0e.y > 180.0)
    EulerKalman0e.y = -(360.0f - EulerKalman0e.y);
if (EulerKalman0e.z > 180.0)
    EulerKalman0e.z = -(360.0f - EulerKalman0e.z);

if (EulerGMV0e.x > 180.0)
    EulerGMV0e.x = -(360.0f - EulerGMV0e.x);
if (EulerGMV0e.y > 180.0)
    EulerGMV0e.y = -(360.0f - EulerGMV0e.y);
if (EulerGMV0e.z > 180.0)
    EulerGMV0e.z = -(360.0f - EulerGMV0e.z);

if (EulerGMV1e.x > 180.0)
    EulerGMV1e.x = -(360.0f - EulerGMV1e.x);
if (EulerGMV1e.y > 180.0)
    EulerGMV1e.y = -(360.0f - EulerGMV1e.y);
if (EulerGMV1e.z > 180.0)
    EulerGMV1e.z = -(360.0f - EulerGMV1e.z);

System.IO.File.AppendAllText(FileStrMark,
    System.String.Format("T{0},", CurrentTime)); //write data to file

markCount++;
print("Orientation marked (" + markCount.ToString() + "/" +
    Qref.Count.ToString() + ")");
    handRotated = false;
}
GUILayout.EndVertical();
GUILayout.EndArea();

else
{
    GUILayout.BeginArea(new Rect(Screen.width - 210, Screen.height - 40, 200, 200));
    GUILayout.BeginVertical("box");
    if (GUILayout.Button("Show Hand Sequence", buttonGUIStyle))
    {
        print("Hand Model is rotating");
        rotCount++;
        k = 0.0f;
        handRotated = true;
    }
    GUILayout.EndVertical();
    GUILayout.EndArea();
}
guiStyle = GUI.skin.GetStyle("Label");
guiStyle.fontSize = 30; // change the font size
guiStyle.alignment = TextAnchor.UpperCenter;

float movePosSp = 0.05f; // Position moving speed.
if (rotCount == 0)
{
    GUI.Label(new Rect(Screen.width / 2 - 200, 40, 400, 100), step[0], guiStyle);
} else if (rotCount == 1)
{
    Vector3 newPosition = transform.position; // We store the current position
    if((int)transform.position.z != -3)
    {
        newPosition.z = newPosition.z - movePosSp;
        transform.position = newPosition; // We pass it back
    }
    GUI.Label(new Rect(Screen.width / 2 - 200, 40, 400, 100), step[1], guiStyle);
} else if (rotCount == 10)
{
    Vector3 newPosition = transform.position; // We store the current position
    if((int)transform.position.x != -3){
        newPosition.x = newPosition.x - movePosSp;
        transform.position = newPosition; // We pass it back
    }
    GUI.Label(new Rect(Screen.width / 2 - 200, 40, 400, 100), step[2], guiStyle);
} else if (rotCount == 19)
{
    Vector3 newPosition = transform.position; // We store the current position
    if((int)transform.position.x != 3 &&
    (int)transform.position.z != 2){
        newPosition.x = newPosition.x + movePosSp;
        newPosition.z = newPosition.z + movePosSp;
        transform.position = newPosition; // We pass it back
    }
    GUI.Label(new Rect(Screen.width / 2 - 200, 40, 400, 100), step[3], guiStyle);
} else
{
    GUI.Label(new Rect(Screen.width / 2 - 200, 40, 400, 100), "Performing", guiStyle);
}

if (rotCount == 2 || rotCount == 11)
{
    GetComponent<Renderer>().material.color = Color.green;
} else if (rotCount == 4 || rotCount == 13)
{
    GetComponent<Renderer>().material.color = Color.blue;
} else if (rotCount == 6 || rotCount == 15)


```csharp
{
    GetComponent<Renderer>().material.color = Color.yellow;
}
else if (rotCount == 8 || rotCount == 17)
{
    GetComponent<Renderer>().material.color = Color.red;
}
else
{
    GetComponent<Renderer>().material.color = Color.grey;
}
}

void GMVD()
{
    for (int i = BuffSize - 1; i >= 1; i--)
    {
        GyroBuff0[i] = GyroBuff0[i - 1];
        AcceleroBuff0[i] = AcceleroBuff0[i - 1];
        MagnetoBuff0[i] = MagnetoBuff0[i - 1];
        StillnessBuff0[i] = StillnessBuff0[i - 1];
    }
    GyroBuff0[0] = Gyro0;
    AcceleroBuff0[0] = Accelero0;
    MagnetoBuff0[0] = Magneto0;
    StillnessBuff0[0] = Stillness0;

    N += 1;
    int ClampBuffSize = Mathf.Clamp(N, 0, BuffSize);

    GyroAvg0 = new Vector3(0, 0, 0);
    AcceleroAvg0 = new Vector3(0, 0, 0);
    MagnetoAvg0 = new Vector3(0, 0, 0);
    StillnessAvg0 = new float();

    for (int i = 0; i < ClampBuffSize; i++)
    {
        GyroAvg0 += GyroBuff0[i];
        AcceleroAvg0 += AcceleroBuff0[i];
        MagnetoAvg0 += MagnetoBuff0[i];
        StillnessAvg0 += StillnessBuff0[i];
    }
    GyroAvg0 /= ClampBuffSize;
    AcceleroAvg0 /= ClampBuffSize;
    MagnetoAvg0 /= ClampBuffSize;
    thisAlphaX0 = Mathf.Pow((StillnessAvg0 / ClampBuffSize), 2.0f);
    thisAlphaY0 = alpWeight * (prevAlphaX0) + (1.0f - alpWeight) * (prevAlphaY0);
    prevAlphaX0 = thisAlphaX0;
    prevAlphaY0 = thisAlphaY0;

    // Calculate new Bias offset Errors when the sensor is NOT rotating
    if (Mathf.Abs(Gyro0.x) < 0.03 && Mathf.Abs(Gyro0.y) < 0.03 &&
        Mathf.Abs(Gyro0.z) < 0.03)
    {
        BiasBuff0.x += Gyro0.x;
    }
```
BiasBuff0.y += Gyro0.y;
BiasBuff0.z += Gyro0.z;
trigcount0 += 1;
}
else
{
    trigcount0 = 0;
    BiasBuff0.x = 0;
    BiasBuff0.y = 0;
    BiasBuff0.z = 0;
}
if (trigcount0 == 5)
{
    Bias0 = new Vector3(BiasBuff0.x / 5.0f, BiasBuff0.y / 5.0f,
    BiasBuff0.z / 5.0f);
    trigcount0 = 0;
    BiasBuff0.x = 0;
    BiasBuff0.y = 0;
    BiasBuff0.z = 0;
} // Remove Gyroscope Bias
    UnbiasedGyro0 = Gyro0 - (B_SCALING * Bias0);
// Compute Quaternions
if (N == 1)
{
    A_int0 = new Quaternion(Accelero0.x, Accelero0.y, Accelero0.z,
    0.0f);
    M_int0 = new Quaternion(Magneto0.x, Magneto0.y, Magneto0.z, 0.0f);
}
w0 = new Quaternion(UnbiasedGyro0.x, UnbiasedGyro0.y, UnbiasedGyro0.z,
0.0f);
qG0 = qOUT1;
qGA0 = qOUT1;
qGM0 = qOUT1;

dqG0 = (qG0 * w0);
    dqG0 = new Quaternion(0.5f * dqG0.x, 0.5f * dqG0.y, 0.5f * dqG0.z, 0.5f
    * dqG0.w);
qG0 = myQuatIntegrate(dqG0, qG0, SamplingTime);
qG0 = myQuatNormalize(qG0);

dqGA0 = (qGA0 * w0);
    dqGA0 = new Quaternion(0.5f * dqGA0.x, 0.5f * dqGA0.y, 0.5f * dqGA0.z,
    0.5f * dqGA0.w);
qGA0 = myQuatIntegrate(dqGA0, qGA0, SamplingTime);
qGA0 = myQuatNormalize(qGA0);

dqGM0 = (qGM0 * w0);
    dqGM0 = new Quaternion(0.5f * dqGM0.x, 0.5f * dqGM0.y, 0.5f * dqGM0.z,
    0.5f * dqGM0.w);
qGM0 = myQuatIntegrate(dqGM0, qGM0, SamplingTime);
qGM0 = myQuatNormalize(qGM0);

// Compute Gravity and Magnetic North Vectors
a40 = myQuatConj(qGA0) * (A_int0 * qGA0);
\[ a_{30} = \text{new Vector3}(a_{40}.x, a_{40}.y, a_{40}.z); \]
\[ m_{40} = \text{myQuatConj}(q_{GM0}) * (M_{int0} * q_{GM0}); \]
\[ m_{30} = \text{new Vector3}(m_{40}.x, m_{40}.y, m_{40}.z); \]

// Compute Differences between measured and calculated Gravity Vector described in Quaternion domain

\[ \text{Vector3 } q_{A0} = \text{Vector3.Cross}(\text{AcceleroAvg0}, a_{30}); \]
\[ \text{float } q_{A0} = \text{Vector3.Magnitude}(\text{AcceleroAvg0}) * \text{Vector3.Magnitude}(a_{30}) + \]
\[ \text{Vector3.Dot}(\text{AcceleroAvg0}, a_{30}); \]
\[ \text{Quaternion } \delta q_{A0} = \text{myQuatNormalize}(\text{new Quaternion}(q_{A0}.x, q_{A0}.y, q_{A0}.z, q_{A0}.w)); \]
\[ q_{GA0} = \text{myQuatNormalize}(q_{GA0} * \delta q_{A0}); \]

// Compute Differences between measured and calculated Magnetic North Vector described in Quaternion domain

\[ \text{Vector3 } q_{M0} = \text{Vector3.Cross}(\text{Magneto0}, m_{30}); \]
\[ \text{float } q_{M0} = \text{Vector3.Magnitude}(\text{Magneto0}) * \text{Vector3.Magnitude}(m_{30}) + \]
\[ \text{Vector3.Dot}(\text{Magneto0}, m_{30}); \]
\[ \text{Quaternion } \delta q_{M0} = \text{myQuatNormalize}(\text{new Quaternion}(q_{M0}.x, q_{M0}.y, q_{M0}.z, q_{M0}.w)); \]
\[ q_{GM0} = \text{myQuatNormalize}(q_{GM0} * \delta q_{M0}); \]

// Quaternion Interpolation

\[ \alpha_0 = \text{thisAlphaY0}; \]

PUBLIC //GENERATE (alpha)

\[ \alpha_0 = (ma \times \alpha_0) + (1 - ma); //\text{Linear Equation} \]
\[ \alpha_0 = (\alpha_0 + (\text{Math.Abs}(\alpha_0))) / 2; \]

PUBLIC //GENERATE (MU)

\[ q_{Gpost} = q_{GA0}; \]
\[ m_{GA4} = \text{myQuatConj}(q_{Gpost}) * (M_{int0} * q_{Gpost}); \]
\[ m_{G3} = \text{new Vector3}(m_{GA4}.x, m_{GA4}.y, m_{GA4}.z); \]
\[ \text{norm}_{MagAvg} = (\text{float})\sqrt{(\text{Magneto0}.x \times \text{Magneto0}.x) + (\text{Magneto0}.y \times \text{Magneto0}.y) + (\text{Magneto0}.z \times \text{Magneto0}.z)}; \]
\[ \text{norm}_{mG3} = (\text{float})\sqrt{(m_{GA3}.x \times m_{GA3}.x) + (m_{GA3}.y \times m_{GA3}.y) + (m_{GA3}.z \times m_{GA3}.z)}; \]
\[ \cosGamma = \text{Vector3.Dot}(\text{MagnetoAvg0}, m_{GA3}); \]
\[ \cosGamma = \cosGamma / (\text{norm}_{MagAvg} \times \text{norm}_{mG3}); \]
\[ \text{if } (\cosGamma > 1.0f) \]
\[ \cosGamma = 1.0f; \]
\[ \text{if } (\cosGamma < -1.0f) \]
\[ \cosGamma = -1.0f; \]

//\text{cosGamma} = \text{MagnetoAvg0.dot}(m_{GA3});
\[ gg = (\text{float})\text{Math.Acos}((\text{double})\cosGamma); \]

//\text{gg} = Gamma;
\[ gl = -mt \times gg + 1.0f; //\text{Linear Equation} \]
\[ \text{temp}_mu = (1.0f + gl) / 2.0f; \]
\[ \text{temp}_mu = gg; \]
\[ \text{if } (\text{temp}_mu < 0) \]
\[ \{ \]
\[ \text{thisMu} = 0; \]

134
else {
  thisMu = temp_mu;
}

float thisMuY = 0.0f;
float thisMuX = thisMu * thisMu;

thisMuY = (muWeight* prevMuX) + (1.0f-muWeight) * prevMuY;
prevMuX = thisMuX;
prevMuY = thisMuY;
LocateVOX(OptitrackStreamingClient.markerX, OptitrackStreamingClient.markerY, OptitrackStreamingClient.markerZ);

if (MU[LocateVoxX, LocateVoxY, LocateVoxZ] < TM && alpha0 > TA) {
  MU[LocateVoxX, LocateVoxY, LocateVoxZ] = thisMuY;
}

goUT1 = Quaternion.Slerp((Quaternion.Slerp(qG0, qGM0, MU[LocateVoxX, LocateVoxY, LocateVoxZ])), (Quaternion.Slerp(qG0, qGA0, alpha0)), alpha0);

// Print the data to a file
System.IO.File.AppendAllText(FileStr1, System.String.Format("T{0},{1},{2},{3},{4},{5},{6},{7},{8},{9},{10},{11},{12},{13},{14},{15},{16},{17},{18},{19}n", CurrentTime, StillnessAvg0, Gyro0.x, Gyro0.y, Gyro0.z, Accelero0.x, Accelero0.y, Accelero0.z, Magnet0.x, Magnet0.y, Magnet0.z, alpha0, qOUT1.x, qOUT1.y, qOUT1.z, qOUT1.w, LocateVoxX, LocateVoxY, LocateVoxZ, MU[LocateVoxX, LocateVoxY, LocateVoxZ]));
long milliseconds = DateTimeOffset.Now.ToUnixTimeMilliseconds();
}

// Voxel Calculation
private static void LocateVOX(float x, float y, float z) {
  float OffsetNewOriginX = (float)-50.0; // Set New origin at x: -200
  float OffsetNewOriginY = (float)-50.0; // Set New origin at y: -200

  float MPx = (x);
  float MPy = (y);
  float MPz = (z);

  float nMPx = MPx - OffsetNewOriginX;
  float nMPy = MPy - OffsetNewOriginY;
  float nMPz = MPz;

  int VoxelSize = 1;

  LocateVoxX = (int)(nMPx / VoxelSize) + 1;
  LocateVoxY = (int)(nMPy / VoxelSize) + 1;
  LocateVoxZ = (int)(nMPz / VoxelSize) + 1;
}
Appendix B

The Health Sciences Institutional Review Board (IRB) of Florida International University Protocol Approval
MEMORANDUM

To: Dr. Armando Barreto
CC: Neeranut Ratchatanantakit
From: Elizabeth Juhasz, Ph.D., IRB Coordinator
Date: April 3, 2019

Protocol Title: "Digital Signal Processing for Human-Computer Interactions from Inertial Measurement Units"

The Health Sciences Institutional Review Board of Florida International University has approved your study for the use of human subjects via the Expedited Review process. Your study was found to be in compliance with this institution’s Federal Wide Assurance (00000060).

IRB Protocol Approval #: IRB-19-0110
IRB Approval Date: 03/29/19
TOPAZ Reference #: 107758
IRB Expiration Date: 03/29/22

As a requirement of IRB Approval you are required to:

1) Submit an IRB Amendment Form for all proposed additions or changes in the procedures involving human subjects. All additions and changes must be reviewed and approved by the IRB prior to implementation.
2) Promptly submit an IRB Event Report Form for every serious or unusual or unanticipated adverse event, problems with the rights or welfare of the human subjects, and/or deviations from the approved protocol.
3) Utilize copies of the date stamped consent document(s) for obtaining consent from subjects (unless waived by the IRB). Signed consent documents must be retained for at least three years after the completion of the study.
4) Receive annual review and re-approval of your study prior to your IRB expiration date. Submit the IRB Renewal Form at least 30 days in advance of the study’s expiration date.
5) Submit an IRB Project Completion Report Form when the study is finished or discontinued.

HIPAA Privacy Rule: N/A
Special Conditions: N/A

For further information, you may visit the IRB website at http://research.fiu.edu/irb.
Appendix C

Adult Consent to Participate in a Research Study
SUMMARY INFORMATION

Things you should know about this study:

- **Purpose**: The purpose of the study is to develop a system capable of determining the movement of the human hand in real-time using Inertial Measurement Units (IMUs).
- **Procedures**: If you choose to participate, you will be asked to wearing a glove and follow the tasks.
- **Duration**: This will take about 60 minutes.
- **Risks**: The main risk or discomfort from this research is no different than normally working with a computer.
- **Benefits**: There is no direct benefit to the subject.
- **Alternatives**: There are no known alternatives available to you other than not taking part in this study.
- **Participation**: Taking part in this research project is voluntary.

Please carefully read the entire document before agreeing to participate.

PURPOSE OF THE STUDY

The purpose of this study is to develop a system capable of determining the movement of the human hand in real-time by combining two different sources of information: orientation tracking using Inertial Measurement Units (IMUs) and position tracking using infrared cameras.

NUMBER OF STUDY PARTICIPANTS

If you decide to be in this study, you will be one of 60 people in this research study.

DURATION OF THE STUDY

Your participation will involve 60 minutes of time.

PROCEDURES

If you agree to be in the study, we will ask you to do the following things:

1. You will be asked to sit down in front of a desktop monitor and wear a glove on your left hand. The glove used in this experiment is the same as a regular fabric-material work glove used in household

2. Then, you will be asked to perform a sequence of simple hand movement tasks by rotating and/or translating your hand. The hand movement will be numerically recorded and visually display on the computer screen while you are performing the task.
3. You will be asked to repeat performing a sequence of simple hand movement tasks again for different signal processing algorithm.
4. You will take off the glove after finishing the experiment.
5. You will be asked to fill out the questionnaire regarding the experience of using hand motion tracking system.

RISKS AND/OR DISCOMFORTS
The minimal-risk is no different than working with a computer at work or home and the data and the experiment uses non-invasive sensors for data collecting process.

BENEFITS
There is no direct benefit to the subject, other than contributing the knowledge of human-computer interaction to the development of more natural user interface.

ALTERNATIVES
There are no known alternatives available to you other than not taking part in this study. However, any significant new findings developed during the course of the research which may relate to your willingness to continue participation will be provided to you.

CONFIDENTIALITY
The records of this study will be kept private and will be protected to the fullest extent provided by law. In any sort of report, we might publish, we will not include any information that will make it possible to identify you. Research records will be stored securely, and only the researcher team will have access to the records. However, your records may be inspected by authorized University or other agents who will also keep the information confidential.

COMPENSATION & COSTS
You will not be provided any compensation for your participation. There are no costs to you for participating in this study.

RIGHT TO DECLINE OR WITHDRAW
Your participation in this study is voluntary. You are free to participate in the study or withdraw your consent at any time during the study. You will not lose any benefits if you decide not to participate or if you quit the study early. The investigator reserves the right to remove you without your consent at such time that he/she feels it is in the best interest.

RESEARCHER CONTACT INFORMATION
If you have any questions about the purpose, procedures, or any other issues relating to this research study you may contact Neeranut Ratchatanantakit at EC 3970, Tel. (305) 348-6072, Email Address: nratc001@fiu.edu

IRB CONTACT INFORMATION
If you would like to talk with someone about your rights of being a subject in this research study or about ethical issues with this research study, you may contact the FIU Office of Research Integrity by phone at 305-348-2494 or by email at ori@fiu.edu.

PARTICIPANT AGREEMENT
I have read the information in this consent form and agree to participate in this study. I have had a chance to ask any questions I have about this study, and they have been answered for me. I understand that I will be given a copy of this form for my records.

_______________________________
Signature of Participant

_______________________________
Signature of Person Obtaining Consent

_______________________________
Printed Name of Participant

_______________________________
Printed Name of Participant

141
VITA

NEERANUT RATCHATANANTAKIT

1989
Born, Bangkok, Thailand

2009-2013
B.Eng., Mechatronics Engineering
Assumption University
Samut Prakan, Thailand

2013
Automation & Robotics Lab Assistance
HEIG-VD (HES-SO)
Yverdon-les-Bains, Switzerland

2013 - 2015
Mechanical Design & Project Engineer
Seagate Technology (Thailand) Co.Ltd.
Samut Prakan, Thailand

2015 - 2016
PLC Trainer
Bosch Rexroth
Bangkok, Thailand

2016 - 2021
Ph.D., Electrical and Computer Engineering
Florida International University
Miami, Florida

Graduate Teaching Assistant
Florida International University
Miami, Florida

PUBLICATIONS AND PRESENTATIONS


Orientation Correction in a MARG Sensor. In 2021 IEEE SENSORS Virtual Conference. IEEE. (Approved)


