11-7-2018

Hand Motion Tracking System using Inertial Measurement Units and Infrared Cameras

Nonnarit O-larnnithipong
nolar002@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

HAND MOTION TRACKING SYSTEM USING INERTIAL MEASUREMENT UNITS AND INFRARED CAMERAS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Nonnarit O-larnnithipong

2018

To: Dean John Volakis
College of Engineering and Computing

This dissertation, written by Nonnarit O-larnnithipong, and entitled Hand Motion Tracking System Using Inertial Measurement Units and Infrared Cameras, 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.

Malek Adjouadi

Jean Andrian

Wei Zeng

Armando Barreto, Major Professor

Date of Defense: November 7, 2018

The dissertation of Nonnarit O-larnnithipong is approved.

Dean John 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, 2018

DEDICATION

To my parents, family and friends.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and deep appreciation to my major Professor, Dr. Armando Barreto, for his constant encouragement, valuable guidance and support in navigating the research process. I am grateful as the completion of this endeavor would not have been possible without his help, support and patience throughout the study period. He is the best teacher and advisor I have ever had in my life. I also would like to recognize my committee members Dr. Malek Adjouadi, Dr. Jean Andrian and Dr. Wei Zeng for their guidances and expertises.

Over the years of my PhD study, I have been motivated by Dr. Francisco Ortega and Dr. Fatemeh Abyarjoo who provide me the opportunities to extend my academic excellence and encourage me to produce quality research. I am fortunate to meet and be supported by several friends including Pasd Putthapipat, Sitthapon Pumpichet, Chayapol Chaiyanan, Peeraya Inyim, Praew Chantarasinlapin, Lukkamol Prapkree, Neeranut Ratchatanantakit, Panuwat Janwattanapong, Nalat Sornkhampan, Julian Gil, Steward Schwarz, Kelvin Gomez and Daniel Walls. Their friendship, encouragements and guidances facilitated challenges that I encountered during my PhD program. I would like to also extend my special thanks to the Department of Electrical and Computer Engineering staff, Pat Brammer, Layla El-Hilu and Luisa Ruiz for their administrative assistances.

I would like to express my love to my parents, family, and my friends in Thailand. Without their constant supports and encouragements, this accomplishment would never have been possible.

The research in this dissertation was supported by National Sciences Foundation grants HRD-0833093 and CNS-1532061, and the FIU Graduate School Dissertation Year Fellowship.

ABSTRACT OF THE DISSERTATION

HAND MOTION TRACKING SYSTEM USING INERTIAL MEASUREMENT UNITS AND INFRARED CAMERAS

by

Nonnarit O-larnnithipong

Florida International University, 2018

Miami, Florida

Professor Armando Barreto, Major Professor

This dissertation presents a novel approach to develop a system for real-time tracking of the position and orientation of the human hand in three-dimensional space, using MEMS inertial measurement units (IMUs) and infrared cameras. This research focuses on the study and implementation of an algorithm to correct the gyroscope drift, which is a major problem in orientation tracking using commercial-grade IMUs. An algorithm to improve the orientation estimation is proposed. It consists of: 1.) Prediction of the bias offset error while the sensor is static, 2.) Estimation of a quaternion orientation from the unbiased angular velocity, 3.) Correction of the orientation quaternion utilizing the gravity vector and the magnetic North vector, and 4.) Adaptive quaternion interpolation, which determines the final quaternion estimate based upon the current conditions of the sensor.

The results verified that the implementation of the orientation correction algorithm using the gravity vector and the magnetic North vector is able to reduce the amount of drift in orientation tracking and is compatible with position tracking using infrared cameras for real-time human hand motion tracking. Thirty human subjects participated in an experiment to validate the performance of the hand motion tracking system. The statistical analysis shows that the error of position tracking is, on average, 1.7 cm in the x-axis, 1.0 cm in the y-axis, and 3.5 cm in

the z-axis. The Kruskal-Wallis tests show that the orientation correction algorithm using gravity vector and magnetic North vector can significantly reduce the errors in orientation tracking in comparison to fixed offset compensation. Statistical analyses show that the orientation correction algorithm using gravity vector and magnetic North vector and the on-board Kalman-based orientation filtering produced orientation errors that were not significantly different in the Euler angles, Phi, Theta and Psi, with the p-values of 0.632, 0.262 and 0.728, respectively.

The proposed orientation correction algorithm represents a contribution to the emerging approaches to obtain reliable orientation estimates from MEMS IMUs. The development of a hand motion tracking system using IMUs and infrared cameras in this dissertation enables future improvements in natural human-computer interactions within a 3D virtual environment.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

The studies on human-computer interaction have been leading towards the development of systems in which humans would be able to interact with computers more naturally [1], [2], [3]. For example, there are several developments on integrating voice commands into the computer systems or personal mobile devices in order to request information or to command actions. There is also an increasing popularity for the uses of Virtual Reality (VR) and Augmented Reality (AR) in several applications so that the users can experience the interactions with computers or mobile devices more naturally. To interact with our environment, one of the common ways is to move our hands in order to grab, hold, move objects or express body language. Therefore, a computer system that could determine the position, orientation and track the users hand movement in real-time would greatly contribute to the development of natural human-computer interaction. The hand motion tracking system would introduce an alternative way to develop 3D User Interfaces using touchless gestures, to become more natural.

Furthermore, hand motion tracking can be adopted to use in several applications including robotics, gesture recognition, gaming, 3D user interfaces in AR and VR. For 3D User Interfaces applications in AR and VR, the users are subjected to interact with immersive environments. To achieve one of the key goals of AR and VR systems, the user must perceive him or herself as being in the provided AR or VR environment [4], [5], [6], [7]. This state of perception is called "presence". Hence, the AR and VR systems strive to provide the user with highly realistic 3D visual and binaural acoustic output. Nevertheless, in some modern AR and VR systems,

when the user is asked to provide an input to the AR or VR system by performing "unnatural" sequences of actions using an input device (e.g., mouse, keyboard or gamepad controller), the unnatural actions can interrupt the perception of presence. Thus, a full human hand motion tracking system would considerably overcome these current limitations.

1.2 Research Objective

The objective of this research is to develop a system that would be able to determine the position and orientation of the human hand and track the movement of the hand in real-time. The hand motion tracking system could simultaneously monitor two different sources of information: Inertial Measurement Units (IMUs) to determine the orientation of the hand, and infrared cameras to track the hand position. This research also studies and implements an algorithm to correct gyroscope drift within the inertial measurement unit and improve orientation tracking. The orientation correction algorithm is performed by determining the gyroscope bias offset error during the sensor's static periods in order to calculate an estimated quaternion. The orientation in a form of quaternion is then corrected using the gravity vector measured from the accelerometer and the magnetic North vector measured from magnetometer. The goal for this research is to monitor and visualize the movement of the human hand in real-time, including both translation and rotation in three-dimensional space.

1.3 Significance of this Research

An Inertial Measurement Unit (IMU) may have several names, depending on the field of its application. In Aerospace applications, the IMU is used to track an aircraft's position and orientation, and it is called the Attitude and Heading Reference System (AHRS). The IMUs normally consist of Microelectromechanical systems (MEMS) accelerometers and gyroscopes. In some models, the magnetometers are also integrated in the units. This MEMS type of inertial sensors are designed to replace conventional mechanical instruments. The gyroscopes in IMUs can provide the inertial measurements in a form of angular velocity. Hence, the mathematical integration is required to ideally calculate the orientation (angle) of a vehicle or a body in space. However, most commercial-grade MEMS gyroscopes may generate output signals that deviate from zero even when there is no rotational input applied on them. This type of error is called the bias offset error. This bias offset error produces a severe orientation tracking error called "drift", which grows proportional to time. The drift is a common phenomenon in orientation tracking, which causes several problems in navigation and other applications that utilize commercial-grade MEMS gyroscopes to determine the orientation of the vehicles, robot arms or any objects [8], [9].

Many studies have proposed algorithms and solutions to improve orientation tracking and eliminate gyroscope drift in inertial measurement units. Several studies [10], [11], [12] employ Kalman-based processes to correct the error of inertial measurements. However, this Kalman-based approach can be complex and difficult to implement [13], [14]. Some studies utilize the concept of sensor fusion, in which the information from two or more sensors are combined and used to estimate the orientation. The studies in [15] and [16] show examples of sensor fusion approaches

which combine the measurements of accelerometers, gyroscopes, and magnetometers to determine the orientation. It is reasonable to take advantage of the MEMS sensors contained in a single inertial measurement unit. IMUs then become a valid option in terms of cost, dimension and integrability to be used to determine the orientation in human-computer interaction applications.

For this research, we also employ the concept of sensor fusion for orientation tracking of the human hand. An algorithm to correct gyroscope drift and improve orientation tracking is proposed. It makes use of the measurements from the accelerometer, gyroscope, and magnetometer in the IMU, to determine the orientation. As mentioned earlier, we can ideally obtain the estimated orientation by integrating the angular velocity measured from the gyroscope. However, in order to aid the orientation estimation using the measurements from the accelerometer and magnetometer, we need to understand how to determine the orientation from acceleration and from Earth's magnetic field. The acceleration due to gravity is the vector quantity which can be measured by the accelerometer when the IMU is not in motion because it does not include the linear acceleration (i.e. the acceleration associated with the movement of the IMU). Even though the direction of acceleration due to gravity is always pointing towards the Earth's center, in the sensor's body frame the acceleration due to gravity is measured and decomposed into three components along orthogonal axes of the sensor when the sensor is in an oblique orientation. Therefore, the gravity vector measured from the accelerometer can represent the rotation of the sensor's body frame with respect to the Earth frame. The same idea is applied with the measurement of the direction of the magnetic North vector obtained from the magnetometer. This orientation correction algorithm is used to improve the estimated orientation of the human hand to which the IMUs are attached on.

1.4 Problem Statements and Hypotheses

Question 1: Can we build a system to track the hand motion using several input sources?

Hypothesis 1 (Objective 1): The proposed hand motion tracking system will provide the ability to efficiently track the human hand movement in real-time. The system will be capable of combining two different sources of data acquisition and provide accurate 3D visualization of the hand motion.

Question 2: Can the orientation estimate obtained from the IMU be corrected using the proposed algorithm?

Hypothesis 2 (Objective 2): The orientation estimate from the IMU will be corrected using the proposed algorithm, which involves determining the bias offset error and correcting the orientation estimate using the gravity vector from the accelerometer and the magnetic North vector from the magnetometer.

Question 3: Can the readings of the Earth's gravity vector and magnetic field be used to repeatedly fine-tune the orientation estimates obtained from a MEMS IMU?

Hypothesis 3 (Objective 3): The measurements of acceleration and magnetic field from the MEMS IMU can be used under specific circumstances, to fine-tune the orientation estimates obtained from a MEMS IMU.

Question 4: Can the proposed orientation correction algorithm perform at the same level as the internal Kalman filter within an IMU module?

Hypothesis 4: The accuracy of the orientation correction algorithm proposed in this dissertation will not be significantly different from that of the internal Kalman filter.

1.5 Literature Review on Human Hand Motion Tracking Technologies

There are several studies and developments on finger flexion monitoring and human hand motion tracking systems. Each system employs different types and numbers of sensing units to detect the hand motion and finger configurations. In this section, some examples of hand motion tracking systems are presented by categorizing them based on the types of the sensing units or technologies used in the system.

1.5.1 Using Accelerometers

Hernandez built a hand-shape recognition system that would be able to identify the different 26 hand shapes that represent the American Sign Language alphabet [17]. The system utilizes the tracking device called Accele Glove which consists of six 2-axis accelerometers. One accelerometer is attached on each finger and the thumb, and another one is attached on the back of the hand. The pattern recognition system optimizes the signals obtained from all six accelerometers in a form of gravitational vector which can be used to determine the fingers' orientations. The classifier is able to recognize 21 out of 26 letters with 100% accuracy. Another example of hand motion tracking system application for Sign Language has been found in Bai's work. Bai [18] proposed a Vietnmese Sign Language (VSL) recognition system by using six MEMS accelerometers in which five of them are attached to the fingers and the thumb and one accelerometer is attached on the back of the hand. The recognition utilizes a fuzzy rule-based model and a set of Vietnamese spelling rules to identify 23 different postures of the hand and fingers' configuration which represent the 23 Vietnamese letters with space and punctuation. The system can recognize the hand

and fingers posture with high accuracy. There is also a research trend which focuses on the feasibility of the wearable tracking device by developing the system in which the sensor information is transmitted wirelessly. A 3-D hand motion tracking and gesture recognition system developed by Kim [19] consists of one central controller attached on the wrist and tri-axial accelerometers attached to the thumb, middle finger and the back of the hand. The information gathered from all sensing units can be transmitted wirelessly through Bluetooth. The 3-D hand model is created to visualize the hand gestures using the kinematic chain theory. The system is able to recognize three simple hand gestures as scissor, rock, and paper.

1.5.2 Using Electromyogram (EMG) Sensors

A hand gesture recongition system which utilizes the forearm electromyography (EMG) signal was developed by Saponas [20]. The system is able to classify the finger gestures in real-time in situations when that the hands were both holding and not holding an object. The system also provides real-time visualization which can enable the applications in human-computer interaction using muscle-computer interfaces. In Zhang's work [21], not only EMG sensors are used, but the MEMS acclerometers are also utilized to aid in the tracking system. Zhang proposed the human hand gesture recognition system which is composed of a tri-axial MEMS accelerometer in combination with multi-channel surface EMG sensors attached on the wrist of the user. Multi-stream Hidden Markov models were used as decision method to recognize the hand gestures based on the streaming inertial measurement and electromyogram signals. Zhang's proposed method of human hand gesture recognition can promote more natural approaches to human-computer interaction.

1.5.3 Using Magnetic Sensors

uTrack, a technology created by Chen [22], is able to determine the movement of the thumb and fingers by using two magnetometers. The magnetometers are placed on a finger and a permanent magnet is affixed on the back of the thumb. The two magnetometers are separated with a constant distance to eliminate the ambiguity problem. The magnetic sensing data stream was used for 3D pointing. The system can provide the magnet's 3D position and tilt angle. The average tracking performance has the accuracy of 4.84 mm in three-dimensional space. Fahn also presented a system (namely data glove) that is capable of tracking the position of fingertips [23], using magnetic sensors attached on the fingertips. Generator coils are attached on the metacarpal areas in order to produce a magnetic field. The magnetic sensors detect the change in magnetic field due to the flexing of the fingers. The system can determine the fingers' bending angles using a method based on the anatomical shape of the human finger.

1.5.4 Using Resistive, Conductive and Capacitive Sensors

Tarchanidis [24] built a data glove in which force sensors are used to determine the flexing angles of the fingers and the thumb. The glove is made of rubber-coated cotton. The force sensors, utilizing commercial strain gauges, are attached on the back of the fingers and the thumb. Each force sensor outputs linear responses. The data glove prototype can be used in robotics or human-computer interaction applications. Saggio also presented a data glove using resistive bend sensors to detect the flexion of the fingers [25]. The research focused on transforming the output signals from the bend sensors for music composition using a virtual musical instrument interface. The system was able to map the signals from the sensors,

according to the hand movement to the parameters within a musical synthesizer using Neural Networks. Another novel data glove developed by Tognetti [26] is able to determine the hand kinematic configurations. This data glove is made of elastic fabric integrated with conductive elastomer sensor networks on the glove's surface. When the fingers are being flexed, the electrically conductive elastomer changes its resistance, resulting in changes of the electrical output signals. The device was validated to use for hand motion tracking purposes. Kurita [27] proposed a method for contactless hand motion tracking based on the measurement of the current changes due to the changes in capacitances between the moving hand and two electrodes. The changes in current measured are used to determine only the direction and velocity of the human hand movement.

1.5.5 Using Vision-based Technologies

There are several vision-based hand motion tracking technologies. The challenges in most studies are about developing the image processing and pattern recognition algorithms in order to determine hand gestures and track the human hand motion. Nolker proposed a hierarchical approach using neural networks to determine the positions of the fingertips in grayscale images of human hands [28]. The Gabor-Filters have been used to process human hand images. The system is capable of determining the positions of the tip of the fingers and the thumb. It can identify the pointing direction and detect the sequences of the hand movement even when the human hand images have low contrast. Rumyantsev [29] developed a method of detecting hand gestures by utilizing skin color images, a PCA-based detection algorithm of hand gestures and tracking for the hand centroids. The proposed image processing algorithm was implemented in a real-time application for human

hand motion and gesture recognition. Ong proposed an approach to train a robust detector in order to detect the presence of human hands in an image and be able to classify the shape of the hand [30]. The classification of the hand shapes uses a nearest k-neighbors clustering algorithm. The preliminary experimental results achieved a very high successful detection rate. Alsheakhali [31] also presented a new technique of classification for hand motion and trajectory tracking. The proposed technique is able to recognize 12 hand gestures with high accuracy. Park proposed a real-time 3D hand tracking system utilizing a 3D depth sensor [32]. The system tracked the hand location using a Kalman filtering approach. The study validated the proposed method comparing with a method that was exclusively vision-based. The results show that by including the 3D depth sensor, the system is able to track the hand motion more effectively than with the visual-based method. Elgendi [33] developed the human-computer interaction system using human hand motion. The study focuses on the training of a pattern recognition model to determine the human hand gesture speed in a noisy environment using multiple joint features. The proposed method resulted in a high detection accuracy rate.

CHAPTER 2

ROTATION IN THREE-DIMENSIONAL SPACE

In this chapter, we establish some notations, definitions and basic theories of rotation in three-dimensional space. There are several systems to describe the rotation in Euclidean space, two common systems used in this research, which are Euler angles and quaternions, will be presented.

2.1 Coordinate Systems

A coordinate system is a system that consists of numbers and rules to describe the values, positions of points or geometric elements in the Euclidean space. There are several coordinate systems such as Cartesian, Polar, Spherical and Cylindrical coordinate system. In this research, the Cartesian system system will be used. The Cartesian coordinate consists of axes which are perpendicular to each other. In two-dimensional space, the Cartesian coordinates are indicated in a horizontal x-axis, where the numbers on the axis increase from left to right, and vertical y-axis where the numbers on the axis increase in the upward direction. The y-axis is 90 degrees apart from the x-axis in counter clockwise direction. These two axes form a coordinate plane in two-dimensional space. For three-dimensional space, the Cartesian coordinate system consists of the original x- and y-axis in two-dimensional space with an addition of the third axis called z-axis which has its direction perpendicular to the xy-coordinate plane. There are two different coordinate systems for the three-dimensional Cartesian coordinate system depending on the direction of +z direction. Figure 2.1 shows two difference Cartesian coordinate systems: Left-hand and right-hand coordinate systems. The blue, green and red curved arrows indicate the direction of positive rotation about x-, y- and z-axis, respectively.

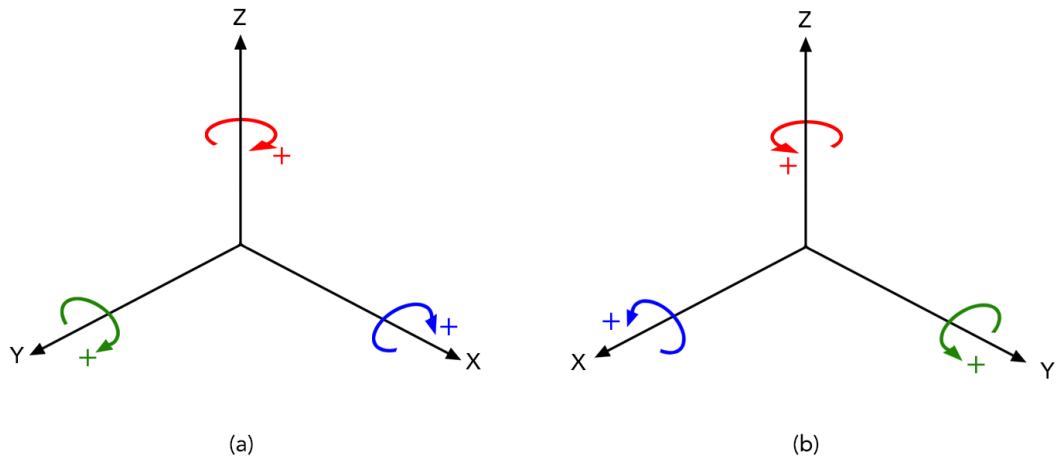


Figure 2.1: (a) Left-hand coordinate system and (b) right-hand coordinate system

2.2 Frames of Reference

In a navigation system, it is necessary to define the frames of reference and their axes in order to standardize and provide the correct measurement of the location and orientation of the vehicles, aircrafts or the moving objects. The reference frames in navigation systems are three-dimensional, orthogonal and right-handed Cartesian coordinate frames.

2.2.1 The Inertial frame

An inertial frame of reference is a frame of reference in which a body is not accelerating or moving at a constant speed in a straight line [34]. This inertial frame of reference has its origin located at the center of the Earth. The frame is not rotating with respect to the fixed stars.

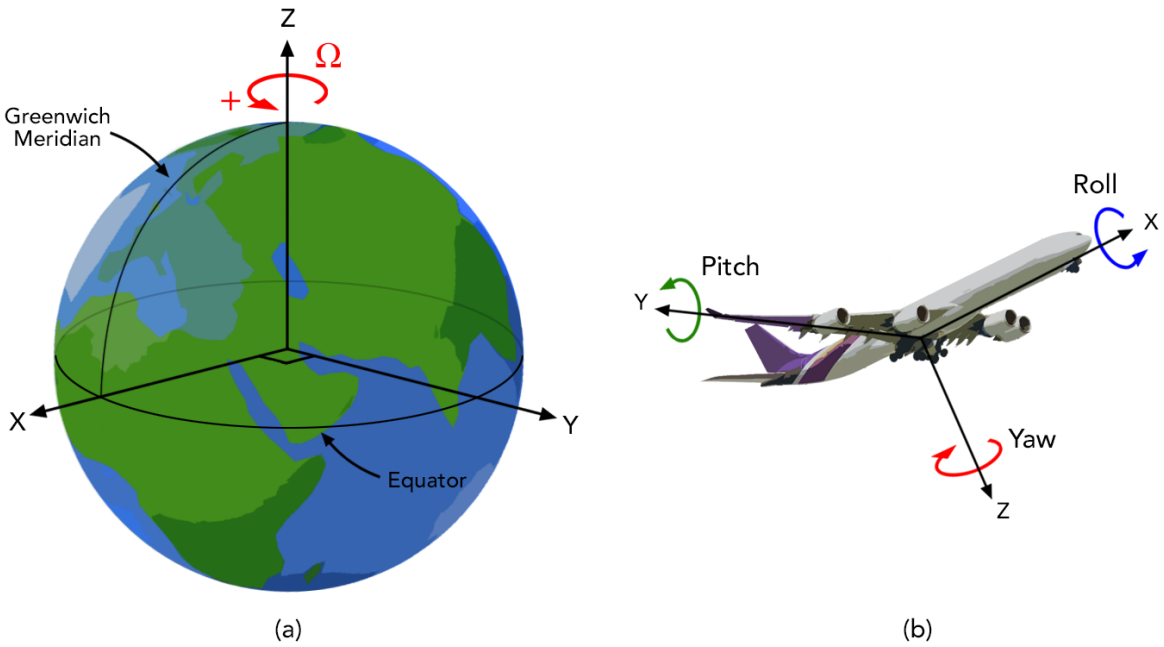


Figure 2.2: (a) Earth frame and (b) body frame

2.2.2 The Earth Frame

The Earth frame is rotating with a constant angular velocity Ω about the Earth's polar axis (z-axis). The x-axis of the Earth's frame is fixed at the intersection of the Greenwich meridian and the Earth's equator. The y-axis is then perpendicular to the xz-plane. This frame is illustrated in Figure 2.2(a).

2.2.3 The Body Frame

The body frame is the frame of reference that is attached to the vehicle or object under study. The z-axis is pointing perpendicularly downward, and the rotation about this axis describes the heading angle of the vehicle (Yaw). The y-axis is pointing to right of the vehicle, and the rotation about y-axis indicates the elevation angle of the vehicle (Pitch). The x-axis is paralell with the vehicle, pointing to where

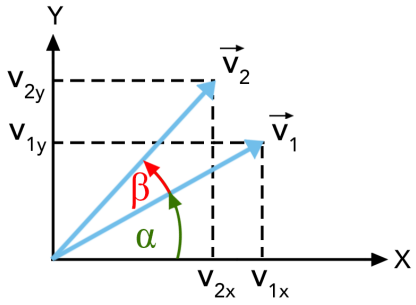


Figure 2.3: Vector rotation on XY-plane

the vehicle is heading. The rotation about x-axis describes the bank angle of the vehicle (Roll). The body frame is depicted on an aircraft as shown in Figure 2.2(b).

2.3 Euler Angles and Direct Cosine Matrix

2.3.1 Rotational Matrices

The rotation in three-dimensional space can be described using rotational matrices. To rotate a point or vector in three-dimensional space, a 3-by-3 rotational matrix R is used to pre-multiply a 3-by-1 coordinate point or vector, resulting in a rotated 3-by-1 point or vector, as shown in Equation 2.1.

$$\begin{aligned} \vec{\mathbf{v}}_2 &= R\ \vec{\mathbf{v}}_1 \\ \begin{bmatrix} v_{2x} \\ v_{2y} \\ v_{2z} \end{bmatrix} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} v_{1x} \\ v_{1y} \\ v_{1z} \end{bmatrix} \end{aligned} \tag{2.1}$$

Consider Figure 2.3, which shows the rotation of a vector on the XY plane about the Z-axis by the angle of β, where the Z-axis is pointing out of the page.

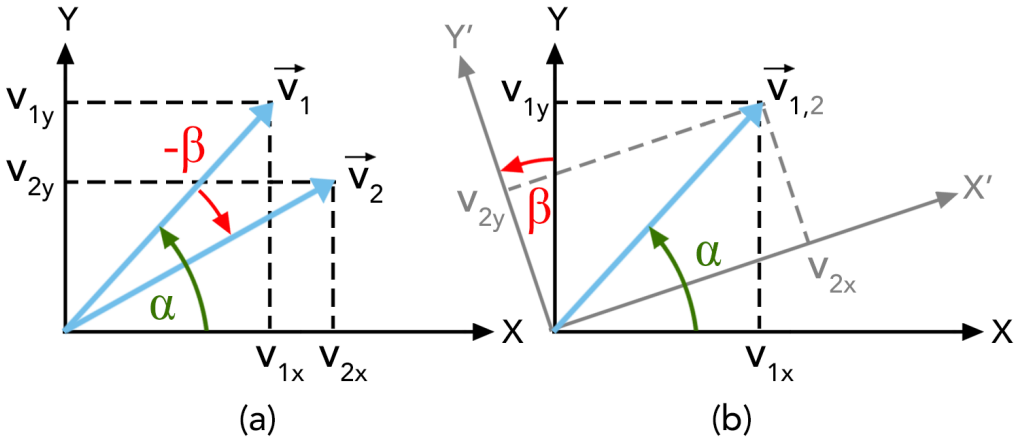


Figure 2.4: Difference perspective of rotation on XY-plane (frame rotation)

The relationship of the coordinates for $\vec{\mathbf{v}}_2$ in term of the components of $\vec{\mathbf{v}}_1$ can be written as shown in Equation 2.2.

$$\begin{aligned}\vec{\mathbf{v}}_2 = \begin{bmatrix} v_{2x} \\ v_{2y} \\ v_{2z} \end{bmatrix} &= \begin{bmatrix} \|\vec{\mathbf{v}}_1\| \, cos(\alpha+\beta) \\ \|\vec{\mathbf{v}}_1\| \, sin(\alpha+\beta) \\ v_{1z} \end{bmatrix} \\ &= \begin{bmatrix} \|\vec{\mathbf{v}}_1\| \, cos(\alpha)cos(\beta) - \|\vec{\mathbf{v}}_1\| \, sin(\alpha)sin(\beta) \\ \|\vec{\mathbf{v}}_1\| \, sin(\alpha)cos(\beta) + \|\vec{\mathbf{v}}_1\| \, cos(\alpha)sin(\beta) \\ v_{1z} \end{bmatrix}\end{aligned} \tag{2.2}$$

Since $v_{1x} = \|\vec{\mathbf{v}}_1\| \, cos(\alpha)$, and $v_{1y} = \|\vec{\mathbf{v}}_1\| \, sin(\alpha)$. Then, we can substitute $\|\vec{\mathbf{v}}_1\| = \frac{v_{1x}}{cos(\alpha)} = \frac{v_{1y}}{sin(\alpha)}$ in Equation 2.2, resulting in Equation 2.3.

$$\vec{\mathbf{v}}_2 = \begin{bmatrix} v_{1x}cos(\beta) - v_{1y}sin(\beta) \\ v_{1y}cos(\beta) + v_{1x}sin(\beta) \\ v_{1z} \end{bmatrix} \tag{2.3}$$

When we consider the different perspectives of rotation, the vector rotation with negative angle $(-\beta)$ shown in Figure 2.4(a) produces the same result as the rotation of coordinate frame with positive angle (β) as shown in Figure 2.4(b). If we substitute $-\beta$ into Equation 2.3, we can obtain Equation 2.4. The coordinate-frame rotational matrix can be rewritten in a form $\vec{\mathbf{v}}_2 = R\ \vec{\mathbf{v}}_1$ as shown in Equation 2.5.

$$\vec{\mathbf{v}}_2 = \begin{bmatrix} v_{1x}cos(-\beta) - v_{1y}sin(-\beta) \\ v_{1y}cos(-\beta) + v_{1x}sin(-\beta) \\ v_{1z} \end{bmatrix} = \begin{bmatrix} v_{1x}cos(\beta) + v_{1y}sin(\beta) \\ v_{1y}cos(\beta) - v_{1x}sin(\beta) \\ v_{1z} \end{bmatrix} \tag{2.4}$$

$$\vec{\mathbf{v}}_2 = \begin{bmatrix} cos(\beta) & sin(\beta) & 0 \\ -sin(\beta) & cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{1x} \\ v_{1y} \\ v_{1z} \end{bmatrix} \tag{2.5}$$

Notice that once a rotation about an axis is performed, the coordinate point in that axis will not change its value. Therefore, we can create unique rotational matrices for rotations about x-, y- and z- axis by using Equation 2.6, 2.7 and 2.8, respectively. The angles ϕ, θ and ψ are the angles that are rotated about each orthogonal axis, these angles are calle Euler angles.

$$R_\phi^x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \tag{2.6}$$

$$R_\theta^y = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{2.7}$$

$$R_{\psi}^{z} = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

2.3.2 Euler Angles

The Euler angles are one of the common systems used to describe the rotation of points, vectors, or geometries in two- or three-dimensional Euclidean space. An Euler angle is an angle of rotation about a coordinate axis. Euler's theorem stated that *"Any two independent orthogonal coordinate frames can be related by a sequence of rotations about coordinate axes, where no two successive rotations may be about the same axis"* [35]. Therefore, any rotations in three-dimensional space can be described using Euler angles by indicating rotations about multiple axes. In this process, the sequence of rotation for each axis matters because the successive rotation will also rotate the previous coordinate axes. By the stated Euler's theorem, any one of these following twelve Euler Angle-axis sequences can be used for a three-dimensional rotation:

xyx xyz xzx xzy yxy yxz yzx yzy zxy zxz zyx zyz

2.3.3 Euler Angle-Axis Sequence: ZYX

In Aircraft and Aerospace applications, the Euler Angle-axis sequence ZYX is commonly used for aircraft's heading and attitude tracking. The Euler Angles describe the aircraft's orientation with respect to the Earth coordinate frame. The first rotational angle of the sequence is an Euler angle *Psi* (ψ), which is the angle rotated about Z-axis that defines the aircraft's heading angle. Then, the rotation about the new y'-axis is considered, indicated by an Euler angle *Theta* (θ) which

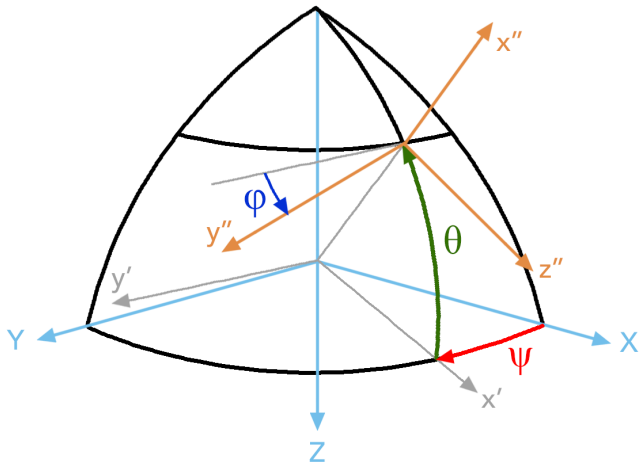


Figure 2.5: The aerospace Euler angle-axis sequence

decries the aircraft's elevation angle. The last rotation of the sequence is about the newest x"-axis, defined by Euler angle *Phi* (ϕ), which defines the aircraft's bank angle. The Aerospace Euler-axis sequence is visually described, as shown in Figure 2.5. The rotational matrix product of this sequence of rotations is derived in Equation 2.9, showing the pre-multiplications of rotational matrices for the successive rotations. The final result of the rotational matrix R consists of three Euler angles and several trigonometrical functions. This rotational matrix that describes the rotation in three-dimensional space is also called Direct Cosine Matrix (DCM). Even though, Euler angles and the Direct Cosine Matrix can clearly describe the rotation in three-dimensional space, Equation 2.9 shows several multiplications and the uses of trigonometrical functions, which require some computational time. In the next section, another representation of rotation called *quaternion* will be presented. Using quaternion as a rotational operator will require less computational time and contains only 4 components instead of 9 elements when using Discrete Cosine Matrix.

$$
\begin{aligned}
R &= R_{\phi}^{x} R_{\theta}^{y} R_{\psi}^{z} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \left(\begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} cos(\theta)cos(\psi) & cos(\theta)sin(\psi) & -sin(\theta) \\ -sin(\psi) & cos(\psi) & 0 \\ sin(\theta)cos(\psi) & sin(\theta)sin(\psi) & cos(\theta) \end{bmatrix} \\
&= \begin{bmatrix} cos(\theta)cos(\psi) & cos(\theta)sin(\psi) & -sin(\theta) \\ \begin{pmatrix} sin(\phi)sin(\theta)cos(\psi) \\ -cos(\phi)sin(\psi) \end{pmatrix} & \begin{pmatrix} sin(\phi)sin(\theta)sin(\psi) \\ +cos(\phi)cos(\psi) \end{pmatrix} & sin(\phi)cos(\theta) \\ \begin{pmatrix} cos(\phi)sin(\theta)cos(\psi) \\ +sin(\phi)sin(\psi) \end{pmatrix} & \begin{pmatrix} cos(\phi)sin(\theta)sin(\psi) \\ -sin(\phi)cos(\psi) \end{pmatrix} & cos(\phi)cos(\theta) \end{bmatrix}
\end{aligned}
\tag{2.9}
$$

2.4 Quaternion

Euler Angles can be used to intuitively describe a rotation in three-dimensional space. However, in some situations, when the pitch angle reaches +/- 90 degree, the same changes in roll and yaw angles will result in the same rotation. Thus, the degrees of freedom of the rotation is reduced from 3 to 2. This phenomenon is called *gimbal lock* or singularity problem, which is one of the common problems in robotics. To prevent this singularity problem, Quaternions will be used as alternative representations to describe rotations in three-dimensional space, overcoming the gimbal lock problem found when using Euler angles.

2.4.1 Definition of Quaternion

Quaternion is a hyper-complex number of rank 4 invented by William Rowan Hamilton in 1843 [35]. It consists of three imaginary components and one real number, denoted with a symbol $\mathbb{H}$ (for Hamilton), as shown in Equation 2.10. There are several ways to denote a quaternion. Equation 2.11 defines a quaternion q as the grouping of a vector $\vec{\mathbf{q}}$ in 3 dimensional space and a scalar q_w. The vector $\vec{\mathbf{q}}$ can be decomposed as the sum of products between the scalars q_x, q_y, q_z and the orthogonal basis in $\mathbb{R}^3$: $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$, respectively.

$$q = \mathbb{H}(\vec{\mathbf{q}}, q_w) \tag{2.10}$$

$$q = \vec{\mathbf{q}} + q_w = q_x\hat{\mathbf{i}} + q_y\hat{\mathbf{j}} + q_z\hat{\mathbf{k}} + q_w \tag{2.11}$$

Notice that q_w, q_x, q_y and q_z are all scalars, thus, the quaternion can be simply defined using just 4 scalar quantities in $\mathbb{R}^4$. In several sources, the real part of the quaternion might be put as the first component. But thorough this research, the scalar q_w that indicates the real part of the quaternion will be placed as the last component, as denoted in Equation 2.12.

$$q = [q_x, q_y, q_z, q_w] \tag{2.12}$$

2.4.2 Quaternion Properties and Calculations

In this section, the essential properties of quaternions will be explained. It is necessary to understand the basic quaternion definitions because they will be required in several calculations when performing the orientation correction algorithm. The quaternion properties are described in the following pages.

a.) Addition

Let a be a quaternion which is defined by Equation 2.13 and b be another quaternion which is defined, as shown in Equation 2.14.

$$a = a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}} + a_w \tag{2.13}$$

$$b = b_x\hat{\mathbf{i}} + b_y\hat{\mathbf{j}} + b_z\hat{\mathbf{k}} + b_w \tag{2.14}$$

The summation between two quaternions a and b is then defined by the adding the corresponding scalar quantities, as shown in Equation 2.15.

$$a + b = (a_x + b_x)\hat{\mathbf{i}} + (a_y + b_y)\hat{\mathbf{j}} + (a_z + b_z)\hat{\mathbf{k}} + (a_w + b_w) \tag{2.15}$$

b.) Multiplication

Let a be a quaternion which is defined by Equation 2.16 and let C be any scalar number in $\mathbb{R}$. The scalar multiplication of Ca is defined by multiplying the scalar C to each of the four components in quaternion a, as shown in Equation 2.17.

$$a = a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}} + a_w \tag{2.16}$$

$$Ca = (Ca_x)\hat{\mathbf{i}} + (Ca_y)\hat{\mathbf{j}} + (Ca_z)\hat{\mathbf{k}} + (Ca_w) \tag{2.17}$$

In order to perform multiplication of two quaternions, the definition has to satisfy the fundamental products between the elements $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ [35], as shown in equations 2.18 to 2.21.

$$\hat{\mathbf{i}}^2 = \hat{\mathbf{j}}^2 = \hat{\mathbf{k}}^2 = \hat{\mathbf{i}}\hat{\mathbf{j}}\hat{\mathbf{k}} = -1 \tag{2.18}$$

$$\hat{\mathbf{i}}\hat{\mathbf{j}} = \hat{\mathbf{k}} = -\hat{\mathbf{j}}\hat{\mathbf{i}} \tag{2.19}$$

$$\hat{\mathbf{j}}\hat{\mathbf{k}} = \hat{\mathbf{i}} = -\hat{\mathbf{k}}\hat{\mathbf{j}} \tag{2.20}$$

$$\hat{\mathbf{k}}\hat{\mathbf{i}} = \hat{\mathbf{j}} = -\hat{\mathbf{i}}\hat{\mathbf{k}} \tag{2.21}$$

Let a be a quaternion which is defined by Equation 2.22 and b be another quaternion which is defined, as shown in Equation 2.23. Then, Equation 2.24 defines the multiplication between two quaternions a and b. It can be straightforwardly determined by multiplying each term in quaternion a into every term in quaternion b, similar to the multiplication between two polynomials. Note that quaternion multiplication is associative and distributive but not commutative, meaning that $a \otimes b$ is not always equal to $b \otimes a$.

$$a = a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}} + a_w \tag{2.22}$$

$$b = b_x\hat{\mathbf{i}} + b_y\hat{\mathbf{j}} + b_z\hat{\mathbf{k}} + b_w \tag{2.23}$$

$$\begin{aligned} a \otimes b &= (a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}} + a_w)(b_x\hat{\mathbf{i}} + b_y\hat{\mathbf{j}} + b_z\hat{\mathbf{k}} + b_w) \\ &= a_x b_x \hat{\mathbf{i}}^2 + a_x b_y \hat{\mathbf{i}}\hat{\mathbf{j}} + a_x b_z \hat{\mathbf{i}}\hat{\mathbf{k}} + a_x b_w \hat{\mathbf{i}} \\ &\quad + a_y b_x \hat{\mathbf{j}}\hat{\mathbf{i}} + a_y b_y \hat{\mathbf{j}}^2 + a_y b_z \hat{\mathbf{j}}\hat{\mathbf{k}} + a_y b_w \hat{\mathbf{j}} \\ &\quad + a_z b_x \hat{\mathbf{k}}\hat{\mathbf{i}} + a_z b_y \hat{\mathbf{k}}\hat{\mathbf{j}} + a_z b_z \hat{\mathbf{k}}^2 + a_z b_w \hat{\mathbf{k}} \\ &\quad + a_w b_x \hat{\mathbf{i}} + a_w b_y \hat{\mathbf{j}} + a_w b_z \hat{\mathbf{k}} + a_w b_w \end{aligned} \tag{2.24}$$

By substituting the Hamilton's fundamental products decribed in equations 2.18 to 2.21, the multiplication of quaternions a and b can be simplified, as shown in Equation 2.25

$$\begin{aligned} a \otimes b = & -a_x b_x + a_x b_y \hat{\mathbf{k}} - a_x b_z \hat{\mathbf{j}} + a_x b_w \hat{\mathbf{i}} \\ & - a_y b_x \hat{\mathbf{k}} - a_y b_y + a_y b_z \hat{\mathbf{i}} + a_y b_w \hat{\mathbf{j}} \\ & + a_z b_x \hat{\mathbf{j}} - a_z b_y \hat{\mathbf{i}} - a_z b_z + a_z b_w \hat{\mathbf{k}} \\ & + a_w b_x \hat{\mathbf{i}} + a_w b_y \hat{\mathbf{j}} + a_w b_z \hat{\mathbf{k}} + a_w b_w \end{aligned} \tag{2.25}$$

Then, the terms are rearranged and combined, which results as Equation 2.26.

$$\begin{aligned} a \otimes b = & -(a_x b_x + a_y b_y + a_z b_z) + a_w b_w \\ & + a_w (b_x \hat{\mathbf{i}} + b_y \hat{\mathbf{j}} + b_z \hat{\mathbf{k}}) + b_w (a_x \hat{\mathbf{i}} + a_y \hat{\mathbf{j}} + a_z \hat{\mathbf{k}}) \\ & + (a_y b_z - a_z b_y) \hat{\mathbf{i}} - (a_x b_z - a_z b_x) \hat{\mathbf{j}} + (a_x b_y - a_y b_x) \hat{\mathbf{k}} \end{aligned} \tag{2.26}$$

By simplifying Equation 2.26 using the dot product and cross product of two three-dimensional vectors, the final quaternion multiplication definition is shown in Equation 2.27.

$$a \otimes b = a_w b_w - (\vec{\mathbf{a}} \cdot \vec{\mathbf{b}}) + a_w \vec{\mathbf{b}} + b_w \vec{\mathbf{a}} + (\vec{\mathbf{a}} \times \vec{\mathbf{b}}) \tag{2.27}$$

Once again, if we rearrange Equation 2.25, by grouping the coefficients for each quaternion component: $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$ and real part together, we can obtain Equation 2.28. Then, the quaternion multiplication can be rewritten in matrix form as shown in Equation 2.29.

$$
\begin{aligned}
a \otimes b = & [a_x b_w + a_y b_z + a_z(-b_y) + a_w b_x]\hat{\mathbf{i}} \\
& + [a_x(-b_z) + a_y b_w + a_z b_x + a_w b_y]\hat{\mathbf{j}} \\
& + [a_x b_y + a_y(-b_x) + a_z b_w + a_w b_z]\hat{\mathbf{k}} \\
& + [a_x(-b_x) + a_y(-b_y) + a_z(-b_z) + a_w b_w]
\end{aligned} \tag{2.28}
$$

$$
a \otimes b = \begin{bmatrix} b_w & b_z & -b_y & b_x \\ -b_z & b_w & b_x & b_y \\ b_y & -b_x & b_w & b_z \\ -b_x & -b_y & -b_z & b_w \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \\ a_w \end{bmatrix} \tag{2.29}
$$

c.) Conjugation

Let a be a quaternion which is defined by Equation 2.30. The complex conjugate of quaternion a is denoted as a^*. The conjugation can be made by negating the three imaginary components of the quaternion, as shown in Equation 2.31.

$$
a = \vec{\mathbf{a}} + a_w = a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}} + a_w \tag{2.30}
$$

$$
a^* = -\vec{\mathbf{a}} + a_w = -a_x\hat{\mathbf{i}} - a_y\hat{\mathbf{j}} - a_z\hat{\mathbf{k}} + a_w \tag{2.31}
$$

d.) Norm

The norm of a quaternion a is denoted by $|a|$, and described as shown in Equation 2.32. The quaternion multiplication has been applied to simplify the equation. The norm of a quaternion is simply a square root of the sum of each component squared, as shown in Equation 2.33. Note that the norm of a quaternion is a scalar.

$$\|a\| = \sqrt{a^* \otimes a}$$

$$= \sqrt{\begin{bmatrix} a_w & a_z & -a_y & a_x \\ -a_z & a_w & a_x & a_y \\ a_y & -a_x & a_w & a_z \\ -a_x & -a_y & -a_z & a_w \end{bmatrix} \begin{bmatrix} -a_x \\ -a_y \\ -a_z \\ a_w \end{bmatrix}} \tag{2.32}$$

$$= \sqrt{\begin{bmatrix} -a_x a_w - a_y a_z + a_y a_z + a_x a_w \\ a_x a_z - a_y a_w - a_x a_z + a_y a_w \\ -a_x a_y + a_x a_y - a_z a_w + a_z a_w \\ a_x^2 + a_y^2 + a_z^2 + a_w^2 \end{bmatrix}} = \sqrt{\begin{bmatrix} 0 \\ 0 \\ 0 \\ a_x^2 + a_y^2 + a_z^2 + a_w^2 \end{bmatrix}}$$

$$\|a\| = \sqrt{a_x^2 + a_y^2 + a_z^2 + a_w^2} \tag{2.33}$$

e.) Unit Quaternion

Let a be a quaternion which is defined by Equation 2.34. A unit quaternion of a is denoted by a quaternion u, which is equal to the scalar multiplication of $\frac{1}{\|a\|}$ to the quaternion a, as shown in Equation 2.35. The norm of a unit quaternion, $\|u\| = 1$.

$$a = a_x \hat{\mathbf{i}} + a_y \hat{\mathbf{j}} + a_z \hat{\mathbf{k}} + a_w \tag{2.34}$$

$$u = \frac{1}{\|a\|}(a_x \hat{\mathbf{i}} + a_y \hat{\mathbf{j}} + a_z \hat{\mathbf{k}} + a_w) \tag{2.35}$$

f.) Inverse

Let a be a quaternion which is defined by Equation 2.36. To satisfy the definition of the inverse of a quaternion, a quaternion a^{-1} is an inverse of a quaternion a if and only if $a^{-1} \otimes a = a \otimes a^{-1} = 1$.

$$a = \vec{\mathbf{a}} + a_w = a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}} + a_w \tag{2.36}$$

$$a^{-1} \otimes a = 1 \tag{2.37}$$

In Equation 2.38, the conjugate of the quaternion a^* is multiplied on both sides of Equation 2.37. Then, the associative property and the quaternion multiplication formula from Equation 2.27 are applied, as shown in Equation 2.39.

$$a^{-1} \otimes a \otimes a^* = a^* \tag{2.38}$$

$$a^{-1} \otimes [a_w^2 - (\vec{\mathbf{a}} \cdot (-\vec{\mathbf{a}})) + a_w(-\vec{\mathbf{a}}) + a_w\vec{\mathbf{a}} + (\vec{\mathbf{a}} \times (-\vec{\mathbf{a}}))] = a^* \tag{2.39}$$

Since the second and third terms in the multiplication formula can be cancelled out and the cross product of $\vec{\mathbf{a}}$ and $(-\vec{\mathbf{a}})$ is equal to zero, the simplification is shown in Equation 2.40. Then, the sum $a_w^2 + a_x^2 + a_y^2 + a_z^2$ is basically the square of the norm of the quaternion a. Therefore, the definition of the inverse of a quaternion is finally described as shown in Equation 2.42.

$$a^{-1} \otimes [a_w^2 + a_x^2 + a_y^2 + a_z^2] = a^* \tag{2.40}$$

$$a^{-1}(\|a\|^2) = a^* \tag{2.41}$$

$$a^{-1} = (\frac{1}{\|a\|^2})a^* \tag{2.42}$$

Note that if a is a unit quaternion, the inverse of a unit quaternion a is simply its conjugate, as show in Equation 2.43.

$$a^{-1} = a^* \quad \text{(for unit quaternion } a\text{)} \tag{2.43}$$

g.) Exponential

Let q be a quaternion which is defined by Equation 2.44. The definition of quaternionic exponential is given by e^q, as stated in Equation 2.45.

$$q = q_w + \vec{\mathbf{q}} = q_w + (q_x\hat{\mathbf{i}} + q_y\hat{\mathbf{j}} + q_z\hat{\mathbf{k}}) \tag{2.44}$$

$$e^q = e^{(q_w + \vec{\mathbf{q}})} = e^{q_w}e^{\vec{\mathbf{q}}} \tag{2.45}$$

e^{q_w} is a scalar quantity whereas $e^{\vec{\mathbf{q}}}$ needs to be derived by using the Taylor series expansion for the exponential function as shown in Equation 2.46

$$e^q = e^{q_w}(\sum_{k=0}^{\infty}\frac{\vec{\mathbf{q}}^k}{k!}) = e^{q_w}(1 + \frac{\vec{\mathbf{q}}}{1!} + \frac{\vec{\mathbf{q}}^2}{2!} + \frac{\vec{\mathbf{q}}^3}{3!} + \frac{\vec{\mathbf{q}}^4}{4!} + \frac{\vec{\mathbf{q}}^5}{5!} + \frac{\vec{\mathbf{q}}^6}{6!} + ...) \tag{2.46}$$

Consider equations 2.47 to 2.51, they are the derivations of $\vec{\mathbf{q}}^2$, $\vec{\mathbf{q}}^3$, $\vec{\mathbf{q}}^4$, $\vec{\mathbf{q}}^5$, and $\vec{\mathbf{q}}^6$. Then, the terms are substituted in Equation 2.46, resulting in Equation 2.52.

$$\begin{aligned}
\vec{\mathbf{q}}^2 &= (q_x\hat{\mathbf{i}} + q_y\hat{\mathbf{j}} + q_z\hat{\mathbf{k}})(q_x\hat{\mathbf{i}} + q_y\hat{\mathbf{j}} + q_z\hat{\mathbf{k}}) \\
&= q_x^2\hat{\mathbf{i}}^2 + q_xq_y\hat{\mathbf{i}}\hat{\mathbf{j}} + q_xq_z\hat{\mathbf{i}}\hat{\mathbf{k}} \\
&\quad + q_xq_y\hat{\mathbf{j}}\hat{\mathbf{i}} + q_y^2\hat{\mathbf{j}}^2 + q_yq_z\hat{\mathbf{j}}\hat{\mathbf{k}} \\
&\quad + q_xq_z\hat{\mathbf{k}}\hat{\mathbf{i}} + q_yq_z\hat{\mathbf{k}}\hat{\mathbf{j}} + q_z^2\hat{\mathbf{k}}^2 \\
&= -q_x^2 - q_y^2 - q_z^2 \\
\vec{\mathbf{q}}^2 &= -(q_x^2 + q_y^2 + q_z^2) = -\|\vec{\mathbf{q}}\|^2
\end{aligned} \tag{2.47}$$

$$\vec{\mathbf{q}}^3 = -\|\vec{\mathbf{q}}\|^2\vec{\mathbf{q}} \tag{2.48}$$

$$\vec{\mathbf{q}}^4 = \|\vec{\mathbf{q}}\|^4 \tag{2.49}$$

$$\vec{\mathbf{q}}^5 = \|\vec{\mathbf{q}}\|^4\vec{\mathbf{q}} \tag{2.50}$$

$$\vec{\mathbf{q}}^6 = -\|\vec{\mathbf{q}}\|^6 \tag{2.51}$$

$$e^q = e^{q_w}\left(1 + \frac{\vec{\mathbf{q}}}{1!} - \frac{\|\vec{\mathbf{q}}\|^2}{2!} - \frac{\|\vec{\mathbf{q}}\|^2\vec{\mathbf{q}}}{3!} + \frac{\|\vec{\mathbf{q}}\|^4}{4!} + \frac{\|\vec{\mathbf{q}}\|^4\vec{\mathbf{q}}}{5!} - \frac{\|\vec{\mathbf{q}}\|^6}{6!} + ...\right) \tag{2.52}$$

To simplify the order for each term, $\frac{\|\vec{\mathbf{q}}\|}{\|\vec{\mathbf{q}}\|}$ is multiplied to the odd factorial terms, and the equation is rearranged with the odd and even terms separated, as shown in Equation 2.53.

$$\begin{aligned} e^q &= e^{q_w}\left(1 + \frac{\|\vec{\mathbf{q}}\|\vec{\mathbf{q}}}{1!\|\vec{\mathbf{q}}\|} - \frac{\|\vec{\mathbf{q}}\|^2}{2!} - \frac{\|\vec{\mathbf{q}}\|^3\vec{\mathbf{q}}}{3!\|\vec{\mathbf{q}}\|} + \frac{\|\vec{\mathbf{q}}\|^4}{4!} + \frac{\|\vec{\mathbf{q}}\|^5\vec{\mathbf{q}}}{5!\|\vec{\mathbf{q}}\|} - \frac{\|\vec{\mathbf{q}}\|^6}{6!} + ...\right) \\ &= e^{q_w}\left[\frac{\vec{\mathbf{q}}}{\|\vec{\mathbf{q}}\|}\left(\frac{\|\vec{\mathbf{q}}\|}{1!} - \frac{\|\vec{\mathbf{q}}\|^3}{3!} + \frac{\|\vec{\mathbf{q}}\|^5}{5!} - ...\right) + \left(1 - \frac{\|\vec{\mathbf{q}}\|^2}{2!} + \frac{\|\vec{\mathbf{q}}\|^4}{4!} - \frac{\|\vec{\mathbf{q}}\|^6}{6!} + ...\right)\right] \end{aligned} \tag{2.53}$$

By using the Taylor series expansion formulas for sine and cosine, we can obtain the Equation 2.54 as the exponential of a quaternion q.

$$exp(q) = e^q = e^{q_w}\left(\frac{\vec{\mathbf{q}}}{\|\vec{\mathbf{q}}\|}sin(\|\vec{\mathbf{q}}\|) + cos(\|\vec{\mathbf{q}}\|)\right) \tag{2.54}$$

2.4.3 Quaternion as a Rotation Operator

A unit quaternion q, where its norm $\|q\| = 1$, can be used to rotate points or vectors in three-dimensional space. There are two different perspectives of how we consider

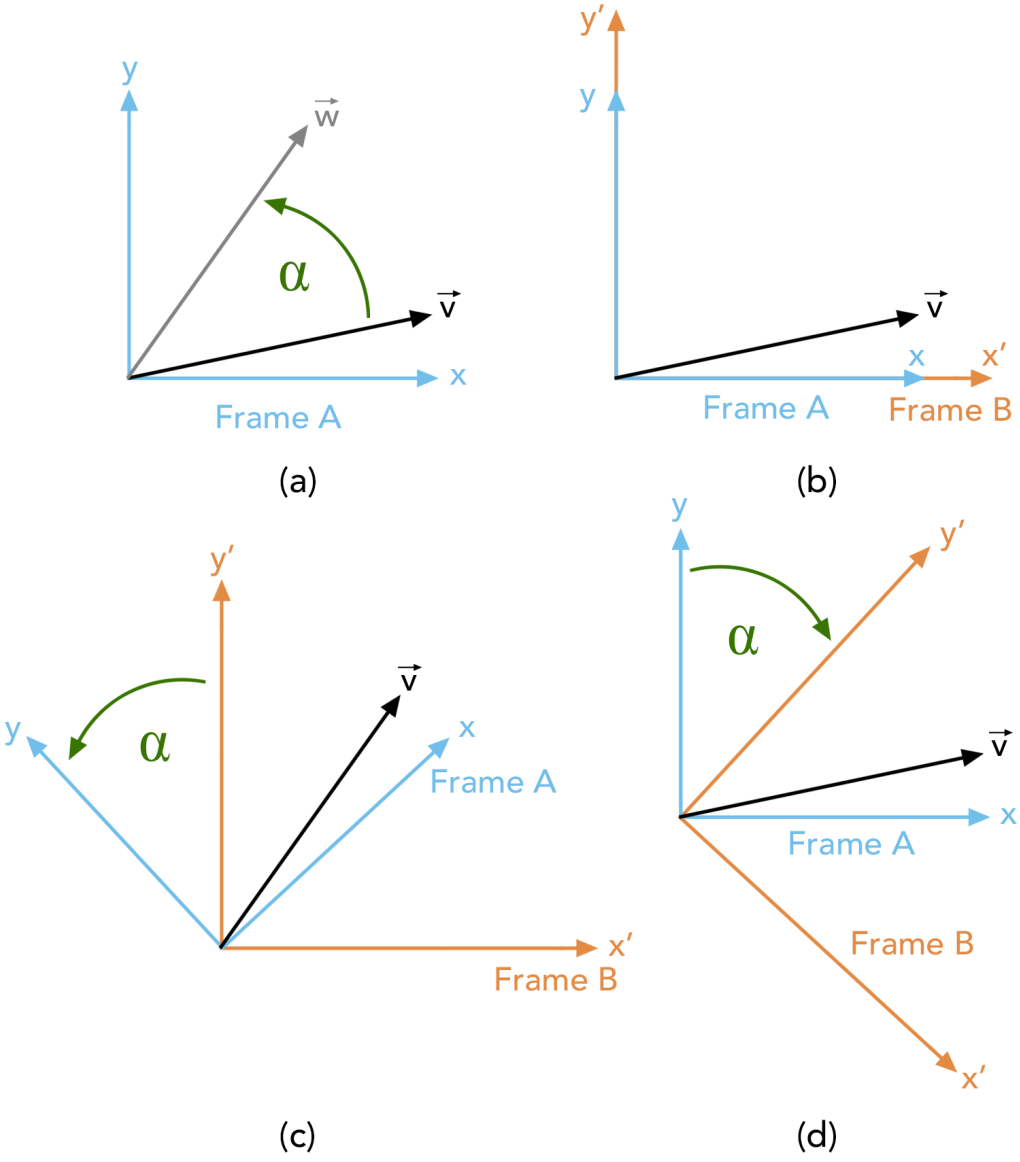


Figure 2.6: Vector rotation and referencing frame rotation

any rotations performed. Firstly, a rotation can be considered as a rotation of a point or vector performed in one referencing coordinate frame. Consider the triple-product quaternion operator described in Equation 2.55, vector ${}^{A}\vec{\mathbf{v}}$ represents any three-dimensional vector referenced in a frame "A". This vector can be transformed by mean of rotation into another vector ${}^{A}\vec{\mathbf{w}}$ which is also referenced in the same frame "A". An example of two-dimensional rotation of a vector is shown in Figure 2.6(a).

$$ {}^{A}\vec{\mathbf{w}} = q \otimes {}^{A}\vec{\mathbf{v}} \otimes q^{*} \tag{2.55} $$

Secondly, the rotation can be considered as a rotation of a reference coordinate frame with respect to another coordinate frame. In Equation 2.56, vector ${}^{A}\vec{\mathbf{v}}$ represents any three-dimensional vector referenced in a frame "A" while vector ${}^{B}\vec{\mathbf{v}}$ represents the same vector but referenced in frame "B". The quaternion ${}^{A}_{B}q$ indicates a rotation of frame "A" with respect to frame "B", as shown in Figure 2.6(c). In another point of view for a frame rotation, we can also consider that frame "B" is rotated with respect to frame "A", as visualized in Figure 2.6(d). The quaternion ${}^{B}_{A}q$ represents this counter-rotation. This pair of opposite rotations can be decribed algebraically in a quaternion form as ${}^{B}_{A}q = {}^{A}_{B}q^{*}$ or ${}^{A}_{B}q = {}^{B}_{A}q^{*}$. With this relationship, we can rewrite Equation 2.56 as shown in Equation 2.57.

$$ {}^{B}\vec{\mathbf{v}} = {}^{B}_{A}q \otimes {}^{A}\vec{\mathbf{v}} \otimes {}^{B}_{A}q^{*} \tag{2.56} $$

$$ {}^{B}\vec{\mathbf{v}} = {}^{A}_{B}q^{*} \otimes {}^{A}\vec{\mathbf{v}} \otimes {}^{B}_{A}q \tag{2.57} $$

Regardless of any perspectives of rotation, to perform a rotation using the triple-product quaternion operator, the vectors or points in $\mathbb{R}^3$ have to be transformed into

quaternion space ($\mathbb{R}^4$). To do so, the three components of the vectors or points are treated as the 3 imaginary parts in quaternion space. Then a zero real part will be attached. This type of quaternion with its real part equal to zero is called a *pure quaternion*.

$$\vec{\mathbf{v}} \in \mathbb{R}^3 = v_x\hat{\mathbf{i}} + v_y\hat{\mathbf{j}} + v_z\hat{\mathbf{k}} = [v_x, v_y, v_z] \tag{2.58}$$

$$\vec{\mathbf{v}} \in \mathbb{R}^4 = \vec{\mathbf{v}} + 0 = v_x\hat{\mathbf{i}} + v_y\hat{\mathbf{j}} + v_z\hat{\mathbf{k}} + 0 = [v_x, v_y, v_z, 0] \tag{2.59}$$

2.4.4 Geometry of a Rotation using Quaternion

After we have learned how to use a quaternion to rotate a vector in three-dimensional space, it is important to understand the relation between the magnitude of the rotating angle and the values of the quaternion components or how the quaternion described the rotation in three-dimensional space, geometrically. The unit quaternion q which can be used as a rotation operator is rewritten in trigonometrical form, as shown in Equation 2.60. The vector $\vec{\mathbf{u}}$ is a unit vector which represents the axis of rotation and θ is a half of the rotating angle about $\vec{\mathbf{u}}$. Therefore, The quaternion q represents a rotation of an angle 2θ, having $\vec{\mathbf{u}}$ as its rotational axis.

$$q = \vec{\mathbf{q}} + q_w = \vec{\mathbf{u}}\ sin(\theta) + cos(\theta) \tag{2.60}$$

Quaternion That Represents a Rotation Between Two Vectors

In Chapter 4, the orientation correction algorithm will be introduced. The chapter includes the correction which requires the construction of quaternion that represents the angular difference between two vectors. In order to achieve that, we need to

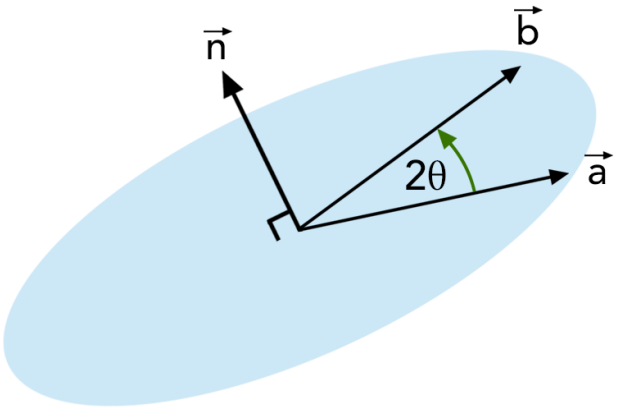


Figure 2.7: Rotation operator geometry for 2θ angle between vectors $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$

know two pieces of information which are the axis of rotation and the magnitude of the rotating angle. The simplest way to obtain these two pieces of information is to treat the two vectors we are studying as if they are on the same plane and have the plane's normal as the axis or rotation, where the magnitude of rotating angle is the angle between the two vectors. Let $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$ be the two vectors in three-dimensional space and 2θ be the angle between them, as shown in Figure 2.7. Consider the dot product and cross product of vectors $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$ in equations 2.61 and 2.62, the trigonometrical identities for double angle are used to derive the equations in order to obtain trigonometrical functions with a half of an angle between two vectors.

$$\begin{aligned}
\vec{\mathbf{a}} \times \vec{\mathbf{b}} &= ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||sin(2\theta)\vec{\mathbf{n}} \\
\vec{\mathbf{a}} \times \vec{\mathbf{b}} &= 2\ ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||sin(\theta)cos(\theta)\vec{\mathbf{n}} \\
\vec{\mathbf{a}} \times \vec{\mathbf{b}} &= \Big[2\ ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||cos(\theta)\Big]\ sin(\theta)\vec{\mathbf{n}}
\end{aligned} \tag{2.61}$$

$$\begin{aligned}
\vec{\mathbf{a}} \cdot \vec{\mathbf{b}} &= ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||cos(2\theta) \\
\vec{\mathbf{a}} \cdot \vec{\mathbf{b}} &= ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||(2cos^2(\theta) - 1) \\
\vec{\mathbf{a}} \cdot \vec{\mathbf{b}} &= 2\ ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||cos^2(\theta) - ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}|| \\
\vec{\mathbf{a}} \cdot \vec{\mathbf{b}} + ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}|| &= \Big[2\ ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||cos(\theta)\Big]\ cos(\theta)
\end{aligned} \tag{2.62}$$

For ease of calculation, the term $\Big[2\ ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||cos(\theta)\Big]$ in Equations 2.61 and 2.62 are then substituted by a variable m.

$$\text{Let } m = 2\ ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||cos(\theta)$$

$$\vec{\mathbf{a}} \times \vec{\mathbf{b}} = (m)\ \vec{\mathbf{n}}\ sin(\theta) \tag{2.63}$$

$$\vec{\mathbf{a}} \cdot \vec{\mathbf{b}} + ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}|| = (m)cos(\theta) \tag{2.64}$$

Since $\vec{\mathbf{a}} \times \vec{\mathbf{b}}$ is a three-dimensional vector and $\vec{\mathbf{a}} \cdot \vec{\mathbf{b}} + ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||$ is scalar, therefore, a quaternion q' can be constructed from these components, as shown in Equation 2.65. Equation 2.66 shows the calculation for the norm of quaternion q'. Since $\vec{\mathbf{n}}$ is a unit quaternion, then its norm squared ($\|q'\|^2 = n_x^2 + n_y^2 + n_z^2$) equals to 1. Therefore, we can conclude as shown in Equation 2.66 that $\|q'\| = m$.

$$q' = \mathbb{H}(\vec{\mathbf{a}} \times \vec{\mathbf{b}},\ \vec{\mathbf{a}} \cdot \vec{\mathbf{b}} + ||\vec{\mathbf{a}}||\ ||\vec{\mathbf{b}}||) = (m)\ \vec{\mathbf{n}}\ sin(\theta) + (m)\ cos(\theta) \tag{2.65}$$

$$\|q'\| = \sqrt{(m)^2\left[(sin^2(\theta))\ (n_x^2 + n_y^2 + n_z^2) + cos^2(\theta)\right]} = m \tag{2.66}$$

$$q = \frac{q'}{\|q'\|} = \vec{\mathbf{n}}\ sin(\theta) + cos(\theta) \tag{2.67}$$

Equation 2.67 satisfies the definition of a unit quaternion that represents the rotation of angle 2θ about $\vec{\mathbf{n}}$. Therefore, the quaternion $q = \frac{q'}{\|q'\|}$ can be used to

describe the angular difference between vectors $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$, where

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

2.5 Relationships between Euler Angles and Quaternion

2.5.1 Euler Angles to Quaternion

To convert Euler angles to quaternion, we know that each Euler angle is the magnitude of the rotation about each orthogonal axis. Psi (ψ) is the heading angle and the angle to rotate about z-axis. Theta (θ) is called elevation angle, rotating about y-axis. And Phi (ϕ), bank angle, is the rotation about x-axis. Since a unit quaternion q_α can describe the rotation of any α angle by Equation 2.68, where $\vec{\mathbf{u}}$ is the axis of rotation, therefore, we can construct the rotations about z-, y- and x-axis as shown in equations 2.69 to 2.71.

$$q_\alpha^{\vec{\mathbf{u}}} = cos(\frac{\alpha}{2}) + \vec{\mathbf{u}} sin(\frac{\alpha}{2}) \tag{2.68}$$

$$q_\psi^{\hat{\mathbf{k}}} = cos(\frac{\psi}{2}) + \hat{\mathbf{k}}\ sin(\frac{\psi}{2}) \tag{2.69}$$

$$q_\theta^{\hat{\mathbf{j}}} = cos(\frac{\theta}{2}) + \hat{\mathbf{j}}\ sin(\frac{\theta}{2}) \tag{2.70}$$

$$q_\phi^{\hat{\mathbf{i}}} = cos(\frac{\phi}{2}) + \hat{\mathbf{i}}\ sin(\frac{\phi}{2}) \tag{2.71}$$

The multiplication of unit quaternions where each one of them describes a rotation, represents the sequence of rotations. Thus, the quaternion product in Equation

2.72 can describe the Euler angle-axis sequence of rotation (ZYX). The value of each quaternion has been substituted as shown in Equation 2.73, then Equation 2.74 is the quaternion equivalence of the rotation using Euler angles with a sequence ZYX.

$$q = q_{\psi}^{\hat{\mathbf{k}}} \otimes q_{\theta}^{\hat{\mathbf{j}}} \otimes q_{\phi}^{\hat{\mathbf{i}}} \tag{2.72}$$

$$q = \left[cos(\frac{\psi}{2}) + \hat{\mathbf{k}}\ sin(\frac{\psi}{2})\right] \otimes \left[cos(\frac{\theta}{2}) + \hat{\mathbf{j}}\ sin(\frac{\theta}{2})\right] \otimes \left[cos(\frac{\phi}{2}) + \hat{\mathbf{i}}\ sin(\frac{\phi}{2})\right] \tag{2.73}$$

$$q = q_x\hat{\mathbf{i}} + q_y\hat{\mathbf{j}} + q_z\hat{\mathbf{k}} + q_w \tag{2.74}$$

where,

$$\begin{aligned}
q_x &= cos(\tfrac{\psi}{2})cos(\tfrac{\theta}{2})sin(\tfrac{\phi}{2}) - sin(\tfrac{\psi}{2})sin(\tfrac{\theta}{2})cos(\tfrac{\phi}{2}) \\
q_y &= cos(\tfrac{\psi}{2})sin(\tfrac{\theta}{2})cos(\tfrac{\phi}{2}) + sin(\tfrac{\psi}{2})cos(\tfrac{\theta}{2})sin(\tfrac{\phi}{2}) \\
q_z &= sin(\tfrac{\psi}{2})cos(\tfrac{\theta}{2})cos(\tfrac{\phi}{2}) - cos(\tfrac{\psi}{2})sin(\tfrac{\theta}{2})sin(\tfrac{\phi}{2}) \\
q_w &= cos(\tfrac{\psi}{2})cos(\tfrac{\theta}{2})cos(\tfrac{\phi}{2}) + sin(\tfrac{\psi}{2})sin(\tfrac{\theta}{2})sin(\tfrac{\phi}{2})
\end{aligned}$$

2.5.2 Quaternion to Euler Angles

Given that a unit quaternion q represents a rotation in three-dimensional space as shown in Equation 2.75. The Euler angles ψ, θ and ϕ that indicate the rotation about z-, y- and x-axis, respectively, can be determined as shown in Equations 2.76 to 2.78 [36].

$$q = q_x\hat{\mathbf{i}} + q_y\hat{\mathbf{j}} + q_z\hat{\mathbf{k}} + q_w \tag{2.75}$$

$$\psi = tan^{-1}\left[\frac{2(q_w q_z + q_x q_y)}{1 - 2(q_y^2 + q_z^2)}\right] \tag{2.76}$$

$$\theta = sin^{-1}\left[2(q_w q_y - q_x q_z)\right] \tag{2.77}$$

$$\phi = tan^{-1}\left[\frac{2(q_w q_x + q_y q_z)}{1 - 2(q_x^2 + q_y^2)}\right] \tag{2.78}$$

CHAPTER 3

MEMS INERTIAL SENSORS

3.1 Inertial Sensors

Microelectromechanical systems (abbreviated as MEMS) are systems in which miniature mechanical and electronic elements are integrated in a single module [37]. MEMS sensors are commonly made of silicon-based microelectronic elements [38] and used to acquire the information of interest in the environment [39]. For example, MEMS sensors can be used to measure air pressure, temperature, applied force or concentration of chemical substances. The MEMS sensors are mostly chosen for commercial-grade applications due to their miniature sizes, lower cost of manufacturing and less power consumption. However, the measurements from MEMS sensors can be very noisy and require calibrations and filtering processes before using their signals. MEMS sensors are also popular in navigation system. MEMS inertial sensors or sometimes called Inertial Measurement Units (IMUs) mostly consists of accelerometers and gyroscopes. They are used to aid the navigation system in providing more accurate position, heading or attitude tracking of a vehicle.

3.2 Accelerometers

An accelerometer can be used to measure the force that is applied to its sensing direction in the body frame with respect to the inertial frame of reference. There are several types of MEMS accelerometers, and they are classified based on how the mechanical displacement is converted into electrical signal [37]. Examples of MEMS accelerometer types are: Piezoresistive, Capacitive, Piezoelectric and Tunneling accelerometers. The mechanical elements of the capacitive MEMS accelerometer con-

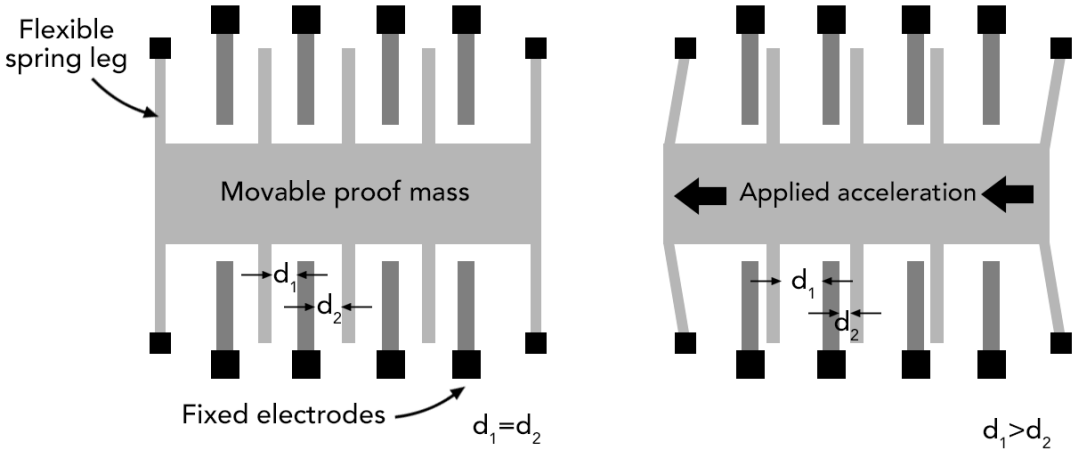


Figure 3.1: Mechanical structure of MEMS accelerometer

sists of a movable proof mass which is anchored to flexible spring legs. When the force is applied the acceleration is sensed in the applying direction, the capacitances between the moving legs which are attached to the movable proof mass and the fixed micro-electrodes are changed due to the change in distance between them. The internal mechanical structure of the MEMS acceleration is illustrated in Figure 3.1. Therefore, the output signal of the MEMS accelerometer is simply generated based on the potential difference across the fixed and moving electrodes. The mathematical model of this MEMS accelerometer is equivalent to the Mass-Damper-Spring system. In the past, measuring the acceleration in three-dimensional space, required three MEMS accelerometers to be arranged in three sensing orthogonal axes but nowadays 3-axis MEMS accelerometers are commonly sold in the market.

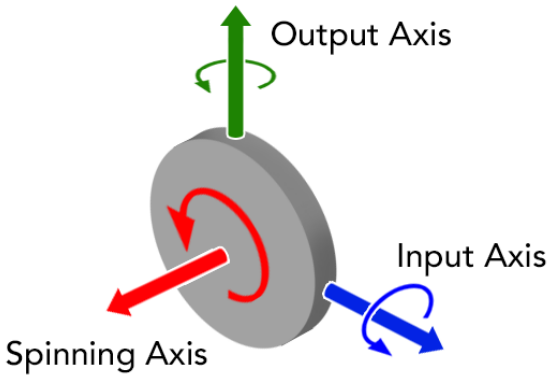


Figure 3.2: Gyroscopic effect

3.3 Gyroscope

A gyroscope is a sensor used to measure angular velocity. The conventional gyroscope is made of a spinning wheel, which is mounted on three freely rotatable gimbals. When the wheel is spinning about its spin axis, the spinning can maintain its spinning axis, and is not affected by the rotation of the external gimbals or the mounting due to the conservation of angular momentum. However, if one attempts to apply a rotation to the input axis of the spinning wheel, the gyroscope will produce a rotational force about the output axis which is perpendicular to the plane of spinning and input axes by following the right-hand rule. This phenomenon is called gyroscopic effect as depicted in Figure 3.2. In contrast, MEMS gyroscopes do not contain any rotating elements. Instead, the sensing unit in MEMS gyroscopes contains a vibratory proof mass and employs the concept of Coriolis effect [37]. When a body is moving from one point to another in a straight line as observed in the inertial frame of reference, it is instead observed as moving on a curved line in a rotating frame due to the Coriolis force produced in the rotation frame. The Coriolis

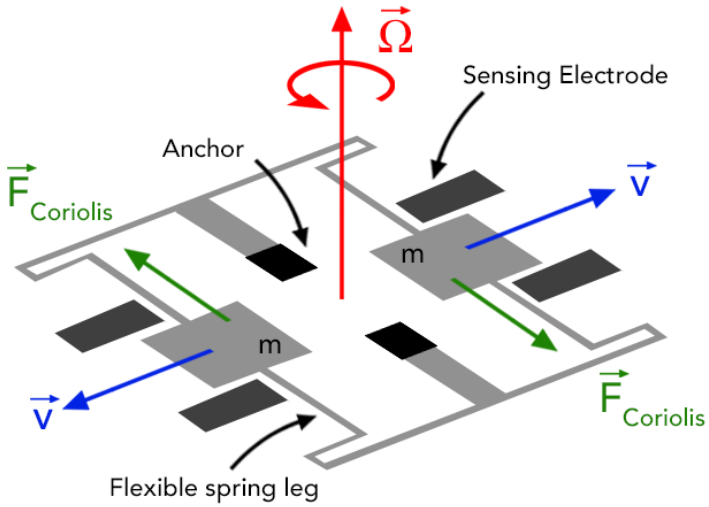


Figure 3.3: Coriolis effect in vibratory MEMS gyroscope

force has its direction perpendicular to both the direction of the moving body and the rotational axis. Consider the internal mechanical structure of MEMS gyroscope illustrated in Figure 3.3. It consists of proof masses which are able to move with two degrees of freedom. The proof masses are attached to flexible spring legs, with their anchors being the only parts that is fixed to the MEMS substrate. When the proof masses are driven with a velocity $\vec{\mathbf{v}}$ as the body frame is rotating with angular rate $\vec{\Omega}$, the Coriolis forces ($\vec{\mathbf{F}}_{Coriolis}$) are produced, having the relationship to the driven velocity and the angular velocity as shown in Equation 3.1. The Coriolis forces are detected by the sensing electrodes using an idea similar to that used in the MEMS accelerometer.

$$\vec{\mathbf{F}}_{Coriolis} = -2m\vec{\Omega} \times \vec{\mathbf{v}} \tag{3.1}$$

3.4 Errors in MEMS Inertial Sensors

As mentioned previously, MEMS inertial sensors are popular for commercial uses because of their miniature sizes and low prices. However, they may produce very large errors in their measurements, in comparison to the tactical-grade inertial measurement devices [40], [41]. There are several types of error in low-cost MEMS initial sensor [42], such as bias offset error, non-orthogonality and scaling factor errors. It is necessary to understand the nature of these errors in MEMS inertial sensors because they can severely disrupt the accuracy of the measurements. Thus, these errors should be addressed before using the MEMS inertial sensors for measurements or navigations. The errors in MEMS inertial sensors can be categorized into two types: systematic (or deterministic) errors and stochastic (or random) errors.

3.4.1 Systematic Errors

Systematic errors (or deterministic) errors are caused by environment or manufacturing inaccuracies, which can be determined or modeled using mathematical models. Some possible systematic errors found in MEMS inertial sensors are the following.

a.) Bias Offset Error

When the inertial sensors are in their static states, in which there is no input force applied on the sensor modules, the inertial sensors should ideally provide the output signals of zero. Instead, the low-cost MEMS inertial sensors could provide positive or negative output signals that deviate from zero. The mean of the MEMS inertial sensor output signal obtained during a static period of time is called bias offset error. The bias offset error is one of major problems in inertial measurements. For a MEMS accelerometer, the bias offset error in the acceleration reading can cause

the linear velocity measurement error to grow proportionally through time (t), and even cause the error of the distance measurements proportional to time squared (t^2) [43]. The equations 3.2 and 3.3 describe the errors of measurement caused by bias offset error in MEMS accelerometer.

$$v_\epsilon = \int (b_a)\ dt = b_a\ t \tag{3.2}$$

$$\text{Error in velocity} \propto t$$

$$d_\epsilon = \int (b_a\ t) dt = \frac{1}{2} b_a\ t^2 \tag{3.3}$$

$$\text{Error in distance} \propto t^2$$

For a MEMS gyroscope, the bias offset error affects the measurement of angle. Since the MEMS gyroscope provide the output signal in a form of angular velocity. Similar to the accelerometer, the bias offset error in MEMS gyroscope causes the error of angle measurement proportional to time (t). Equation 3.4 shows the relationship between the bias offset error in a MEMS gyroscope and the angle measurement, where θ_ϵ is the error in angle measurement and b_ω is the bias offset error magnitude.

$$\theta_\epsilon = \int (b_\omega)\ dt = b_\omega\ t \tag{3.4}$$

$$\text{Error in angle} \propto t$$

b.) Scaling Factor Error

The scaling factor describes a ratio of the electrical output signal to the sensitivity of mechanical input force applied to the MEMS inertial sensor. Ideally, the scaling factor is equal to 1. If the MEMS inertial sensor produces an output signal compared

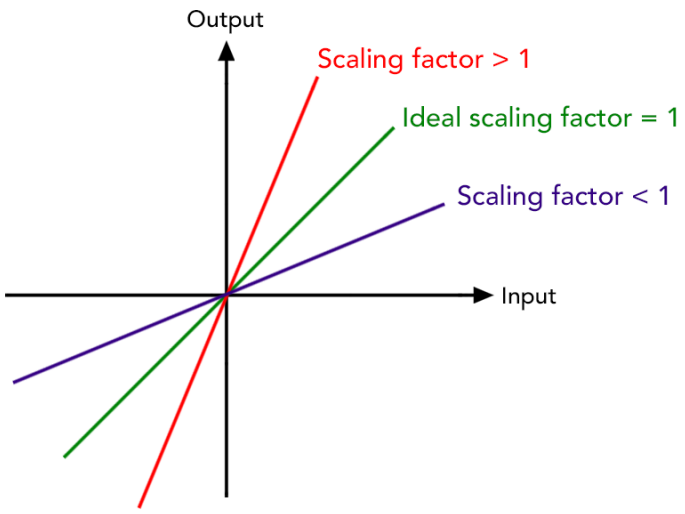


Figure 3.4: Scaling factor errors

to the input applied, greater or smaller than a factor of 1, the MEMS inertial sensor is presenting a scaling factor error, which is generally measured in percentage [37]. These situations are illustrated in Figure 3.4 which depicts the relationship between the input and the output of the MEMS inertial sensor. The scaling factor error can affect the inertial measurements in ways similar to the bias offset error.

c.) Misalignment Error

A misalignment error is typically an error due to inaccuracies during the manufacturing of the MEMS inertial sensor. The misalignment error can be also called "axes non-orthogonality". The term "axes non-orthogonality" itself already describes the nature of this error, in which one or more of the sensing axes are not orthogonal or perpendicular to each other, as depicted in Figure 3.5. The misalignment error can cause both additive and subtractive amounts of inertial measurement error in one or more axes, depending on the misalignment angle of the mounting axis that deviated from the ideal orthogonal axes.

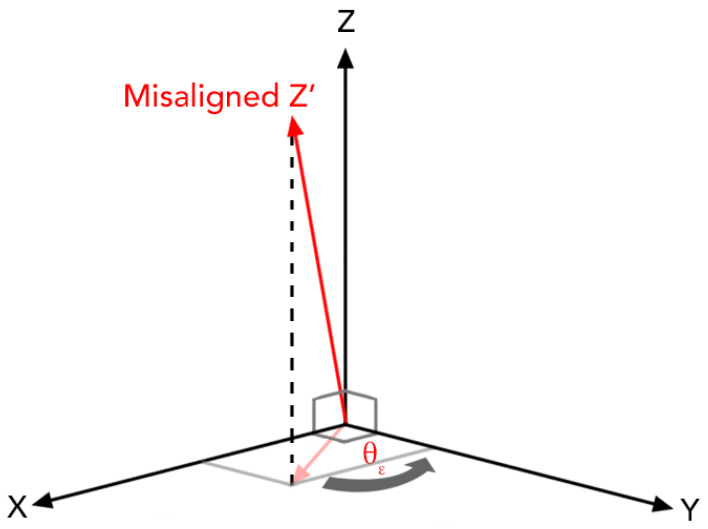


Figure 3.5: Misalignment or axes non-orthogonality errors

d.) Run-to-Run Bias and Scaling Factor

Run-to-run bias offset and scaling factor error are the bias offset and scaling factor that occur in differently for different runs of the MEMS inertial sensor. When the sensor is turned off and turned on again, the bias offset error and scaling factor are slightly changed but remain constant in each run [42]. The MEMS Gyroscope from Yost Labs (used in the implementation of this research) was tested to observe the phenomenon of run-to-run bias offset error. The Yost labs 3-Space Sensor was tested by placing the sensor statically on a table, and the signals from its tri-axial gyroscope were recorded. In Figure 3.6, the plots show the magnitude of the bias offset errors of each gyroscope axis for 45 runs. All 45 runs were recorded from the same MEMS gyroscope, placed in the same testing environment. Each run shows three data points from three axes of the gyroscope, which represent the average of the recorded angular velocity for 60 seconds. The recordings were divided into three parts which are Part 1: runs 1 to 15, Part 2: runs 16 to 30 and Part 3: runs 31

to 45. For the 15 runs in each part, the sensor had been continuously connected to the host PC and turned on. Between each part, the sensor had been turned off and disconnected from the host PC. The recordings in Part 1 (runs 1 to 15) were recorded on a different day from Parts 2 and 3 (runs 16 to 45) which were recorded on the same day. The plots show that the difference in values of bias offset error between runs follow an interesting pattern.

e.) In-Run Bias and Scaling Factor

In-run bias offset error and scaling factor error occur as the value of these errors change during a run. The in-run bias offset error and scaling factor error relate to the inertial sensor's stability to maintain the accuracy of the inertial measurements. A part of the in-run bias offset error and scaling factor error is considered as deterministic error which might be able to be modeled mathematically, because it is caused by environmental conditions or a change in temperature. However, the remainder of the error is a stochastic process.

f.) Effect of Temperature

Since the MEMS inertial sensors have small dimensions, the temperature also plays a roll in affecting the bias offset and scaling factor error in MEMS inertial sensors [34]. Some of the MEMS inertial sensors nowadays also provide a temperature sensor integrated within the same module, which could enable the users to monitor the MEMS inertial sensor's temperature. Then, the variations of the sensor's bias offset and scaling factor error due to the changes in temperature could be modeled and used to correct these errors under different operating temperatures.

3.4.2 Stochastic Errors

Stochastic errors in MEMS inertial sensors are the errors that occur randomly over time and cannot be modeled using deterministic functions. These errors can occur due to random electromagnetic interferences in the environment. However, we can apply several stochastic processes to model these random errors. The simplest model commonly used is a Gaussian Random Process. It can define the random error ($x(t)$) as the normal distribution by using its probability density function for any time t as stated in Equation 3.5.

$$f[x(t)] = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.5}$$

Another common random process that can be used to model the random error is called Random Walk process, where its random variable ($\dot{b}(t)$) describes the integration of a white noise process ($w(t)$) occurred between two consecutive value of a random variable, as described in Equation 3.6. For MEMS inertial sensors, like accelerometer and gyroscope, noise output can be considered to be white noise. The white noise in the output from these devices will be integrated into velocity and angle. The random error of velocity and angle obtained by integrating these white noises are call velocity random walk (VRW) and angular random walk (ARW) [37] for MEMS accelerometers and gyroscopes, respectively. They are the average deviation rates from the correct values [44], which needed to be modeled using stochastic process and removed to obtain more accurate results.

$$\dot{b}(t) = w(t) \tag{3.6}$$

Stochastic Modeling

One of the methods that is used to model the stochastic errors in MEMS inertial sensors is the "autocorrelation function". The parameters that describe the random error are calculated from the autocorrelation of the raw recordings of MEMS inertial outputs. To obtain an accurate autocorrelation function, it is required to collect large amounts of the raw recording data over a long period of time. The accurate autocorrelation function can help in identifying the type of stochastic process of the random error in a MEMS inertial sensor. The stochastic modeling using autocorrelation function can also be transformed into power spectrum density using the Fourier transform.

However, the most common method to model random errors in MEMS inertial sensors is called the Allan Variance [37]. It is a method to characterize the root mean square of the velocity random walk (VRW) and the angular random walk (ARW) as a function of averaged time [40]. The Allan variance method is usually presented in a log-log plot, showing in the y-axis the square root of the Allan variance (Allan standard deviation) versus the cluster time (T) in the x-axis. The Allan variance can be written as shown in Equation 3.7, where $\sigma(T)$ is the Allan standard deviation, and Q is the velocity random walk (VRW) or angular random walk (ARW) coefficient. The cluster time (T) is equal to nt_0, where n is the number of samples in each cluster and t_0 is the sampling time. The value of Q represents the slope of the log-log plot, which can be directly used in a noise matrix for filtering process. The coefficient Q has units of $\frac{m/s}{\sqrt{hr}}$ for velocity random walk (VRW), and $\frac{deg}{\sqrt{hr}}$ for angular random walk (ARW).

$$\sigma(T) = \frac{Q}{\sqrt{T}} \tag{3.7}$$

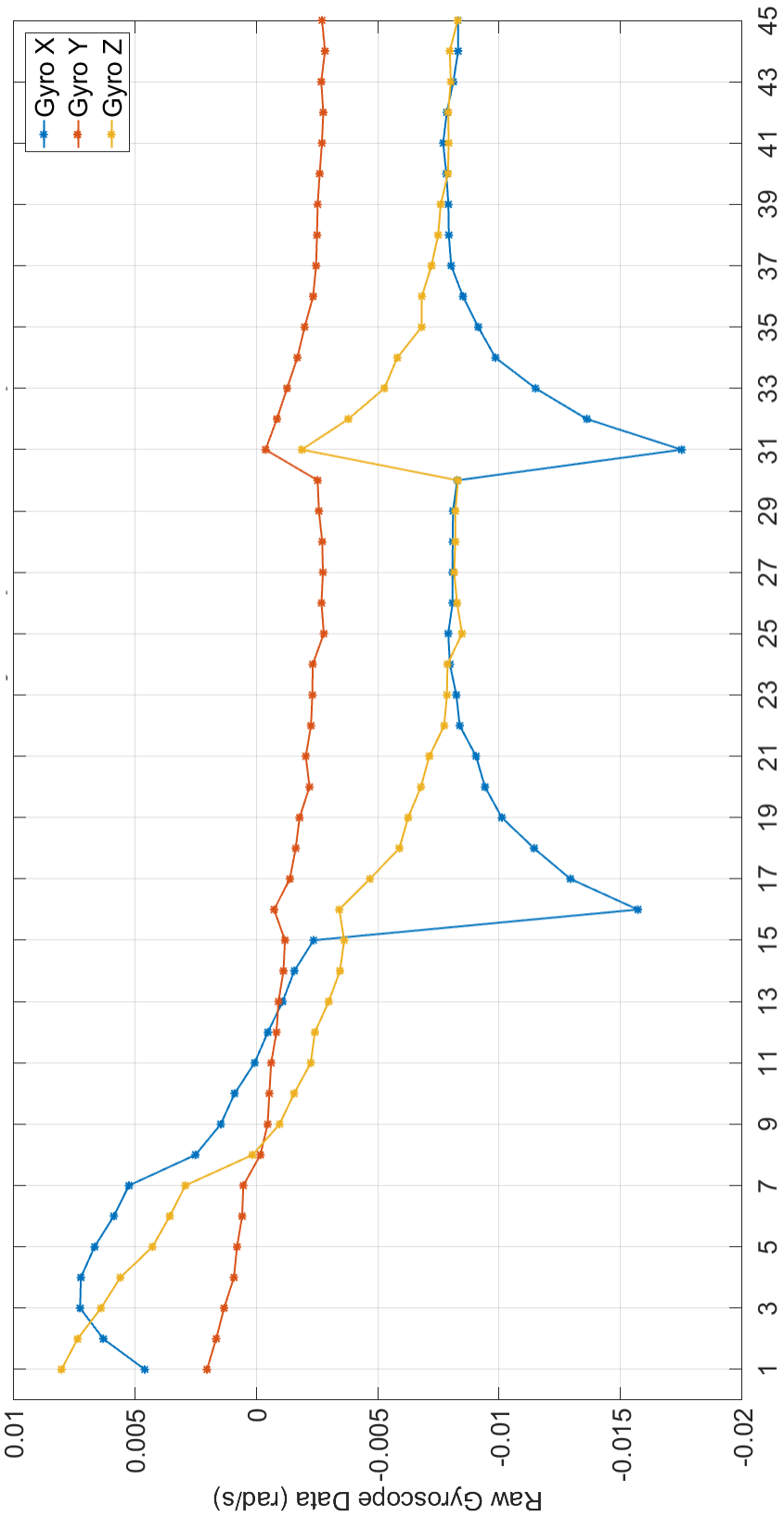


Figure 3.6: Run-to-run bias offset error of MEMS gyroscope: Yost Labs 3-Space Sensor

CHAPTER 4

ORIENTATION CORRECTION ALGORITHM

4.1 Bias Offset Estimation

When there is no input applied to a gyroscope within the IMUs, the sensor is expected to ideally generate zero readings in all axes. But for low-cost MEMS, their output signal could provide the deviation of the measurement from zero called "gyroscope bias". The bias is in part a deterministic error for IMUs, caused by manufacturing defects, environmental conditions or a change in temperature, which can be modeled. However, there are other components of the bias that are considered as stochastic processes [37], resulting in the change of bias randomly through time. Therefore, the bias offset error should be estimated for every period of time. In a real-time situation, the new bias offset error will be re-calculated and updated only when the IMU is in a static period. (i.e. the gyroscope reading has its magnitudes less than predefined thresholds.) To determine the bias offset error, a simple linear regression model [45], as shown in 4.1, is used. The model coefficients (β_1 and β_0) can be found by using Equations 4.2 and 4.3, where b_i and t_i are the measured gyroscope bias and time at the i^{th} sample, respectively.

$$\hat{b} = \beta_0 + \beta_1 t \tag{4.1}$$

$$\beta_1 = \frac{\sum_{t=1}^{n}(t_i - \bar{t})(b_i - \bar{b})}{\sum_{t=1}^{n}(t_i - \bar{t})^2} \tag{4.2}$$

$$\beta_0 = \bar{b} - \beta_1 \bar{t} \tag{4.3}$$

The unbiased angular velocity ($\vec{\omega}_B$) is then calculated by subtracting the calculated bias offset error ($\hat{b}$) from the raw gyroscope reading ($\vec{\omega}_0$) as described in Equation 4.4

$$\vec{\omega}_B = \vec{\omega}_0 - \hat{b} \tag{4.4}$$

4.2 Quaternion Estimation

To avoid gimbal lock problems, quaternion notation is used to represent the rotation, having its real part in the forth component. The quaternion rate ($\dot{q}$) can be calculated by using the unbiased angular velocity ($\vec{\omega}_0$) and $\hat{q}_0$ which is the quaternion estimation of the orientation from the previous iteration. At the beginning of the algorithm, $\hat{q}_0$ is initialized as $[0, 0, 0, 1]$, indicating a rotation of zero degree. An unbiased angular velocity vector ($\vec{\omega}_B$) which is in $\mathbb{R}^3$ can be transformed into quaternion space ($\mathbb{R}^4$) by augmenting zero as its real part (the fourth component), this type of quaternion is called a pure quaternion [35]. Since the real part of ω_B in $\mathbb{R}^4$ is zero, the equation to determine quaternion rate can be written in a matrix form as shown in Equation 4.5.

$$\dot{q} = \frac{1}{2}\hat{q}_0 \otimes \vec{\omega}_B = \frac{1}{2}\begin{bmatrix} 0 & \omega_{Bz} & -\omega_{By} & \omega_{Bx} \\ -\omega_{Bz} & 0 & \omega_{Bx} & \omega_{By} \\ \omega_{By} & -\omega_{Bx} & 0 & \omega_{Bz} \\ -\omega_{Bx} & -\omega_{By} & -\omega_{Bz} & 0 \end{bmatrix} \begin{bmatrix} \hat{q}_{0x} \\ \hat{q}_{0y} \\ \hat{q}_{0z} \\ \hat{q}_{0w} \end{bmatrix} \tag{4.5}$$

$$q_G = exp((\Delta t)\dot{q} \otimes \hat{q}_0^*) \otimes \hat{q}_0 \tag{4.6}$$

The quaternion q_G in Equation 4.6 is a representation of the estimated orientation. To calculate this estimated quaternion q_G, the quaternion rate obtained

from Equation 4.5 is integrated, where Δt is the sampling internal used by the IMU for recording data. This estimated quaternion (q_G) can be used to rotate points or vectors in three-dimensional space, similar to the use of Discrete Cosine Matrix (DCM) as a rotational matrix. (i.e. It can be used to describe the orientation of the sensor module or the object that the IMU is attached to, as a rotation from its initial position.)

4.3 Quaternion Correction

To correct the estimated quaternion (q_G) obtained from Section 4.2, additional information from other sensing units in the IMU will be used. During the sensor's static period (when there is no external forces applied to the sensor), the accelerometer in the IMU would ideally provide only a measurement of the acceleration due to gravity. The magnetometer would provide the direction of the Earth's magnetic field. In this study, the terms ***gravity vector*** and ***magnetic North vector*** will be used to describe the measurements of the direction of the acceleration due to gravity and the direction of the Earth's magnetic field referenced to the sensor's body frame, respectively. In the Earth frame, both acceleration due to gravity and the direction of the Earth's magnetic field are assumed to be constant. But with respect to the sensor's body frame, these vectors have different values depending on the orientation of the sensor's body frame.

Given that a unit quaternion ${}^{E}_{B}q$ represents a rotation in three-dimensional space of a sensor's body frame (denoted by B) with respect to the Earth frame (denoted by E). Then, its conjugate (${}^{E}_{B}q^{*}$) represents the inverse rotation or the rotation of the Earth frame (E) with respect to the sensor's body frame (B) denoted by ${}^{B}_{E}q$, as shown in Equation 4.7

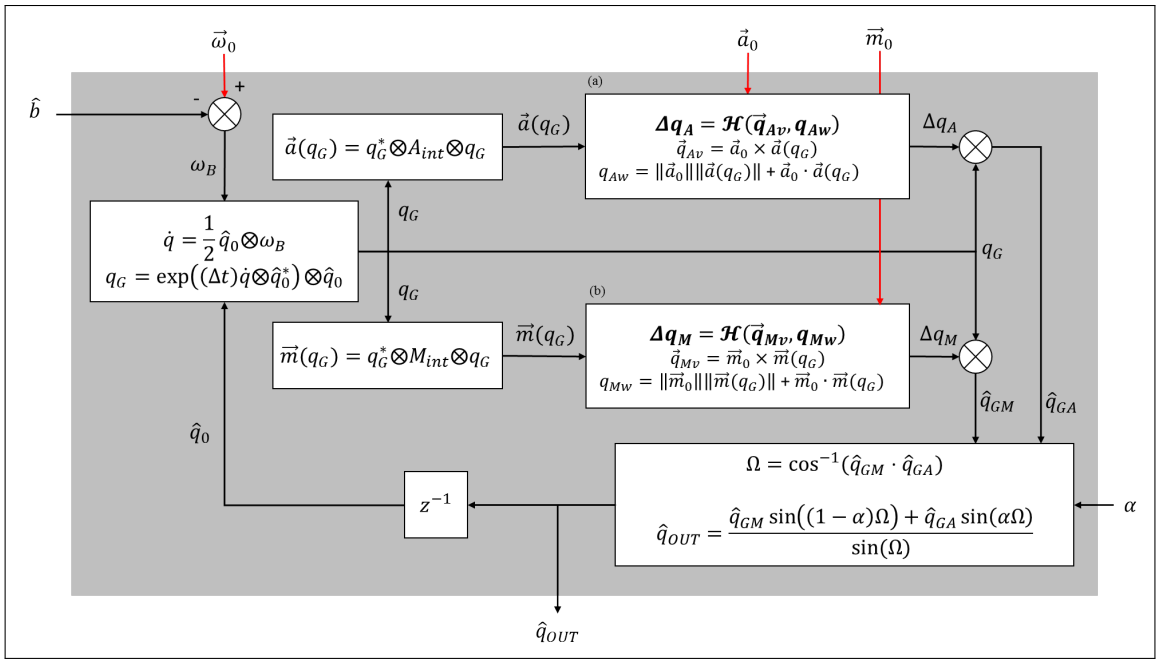


Figure 4.1: Block diagram of the proposed drift correction algorithm using both gravity vector and magnetic North vector correction

$$ {}^{B}_{E}q = {}^{E}_{B}q^* \tag{4.7} $$

$$ {}^{B}\vec{v} = {}^{B}_{E}q \otimes {}^{E}\vec{v} \otimes {}^{B}_{E}q^* \tag{4.8} $$

$$ {}^{B}\vec{v} = ({}^{E}_{B}q^*) \otimes {}^{E}\vec{v} \otimes ({}^{E}_{B}q^*)^* \tag{4.9} $$

The triple-product quaternion operator shown in Equation 4.8 can be applied to rotate any vector referenced to the Earth frame (${}^{E}\vec{v}$) [35, 46] and express that vector in the sensor's body frame as ${}^{B}\vec{v}$. Equation 4.9 shows the substitution of Equation 4.7 into Equation 4.8, then another triple-product quaternion operator is obtained, as shown in Equation 4.10. This quaternion operator will be used for quaternion correction in the following steps since the estimated quaternion (q_G) obtained in

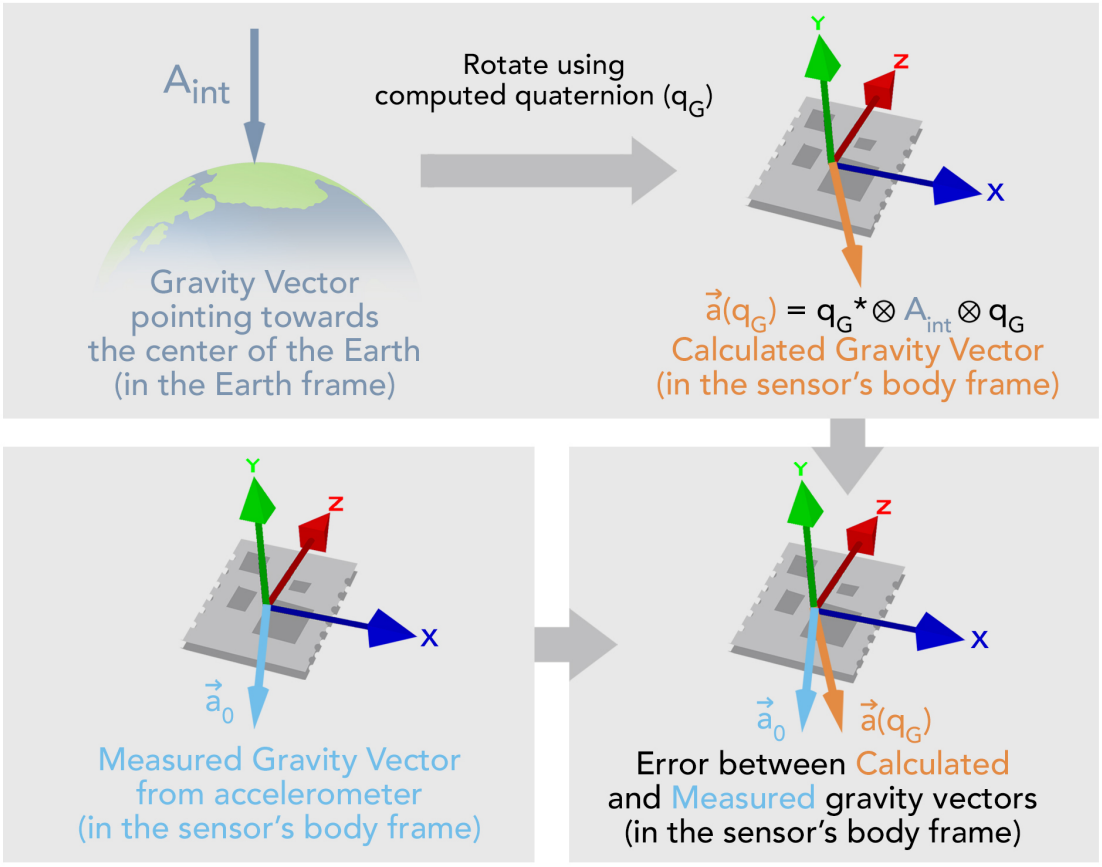


Figure 4.2: Quaternion correction using the gravity vector

Section 4.2 represents the estimated orientation of the sensor's body frame with respect to the Earth frame.

$$^{B}\vec{v} = {}^{E}_{B}q^{*} \otimes {}^{E}\vec{v} \otimes {}^{E}_{B}q \tag{4.10}$$

4.3.1 Using The Gravity Vector

Each axis of the accelerometer measures the magnitude of the acceleration caused by the force applied to the sensor in its sensing direction. When the IMU is in a static period, the accelerometer ideally measures the acceleration due to gravity.

The gravity vector referenced in the Earth frame always points vertically down to the Earth's center. In order to compare this gravity vector (referenced in the Earth's frame) with the *measured gravity vector* ($\vec{a}_0$) from the accelerometer, it is necessary to transform the gravity vector that is "vertical" in the Earth frame, in order to represent its value with respect to the sensor's body frame. The graphical representation of this approach is shown in Figure 4.2. The *calculated gravity vector* ($\vec{a}(q_G)$) referenced in the sensor's body frame can be determined by adapting Equation 4.10, having the result as shown in Equation 4.11.

$$\vec{a}(q_G) = {q_G}^* \otimes A_{int} \otimes q_G \tag{4.11}$$

If there is no error in the orientation estimation (q_G), the *calculated gravity vector* ($\vec{a}(q_G)$) from Equation 4.11 would be equal to the *measured gravity vector* ($\vec{a}_0$) obtained from accelerometer readings. The error between the *calculated gravity vector* ($\vec{a}(q_G)$) and the *measured gravity vector* ($\vec{a}_0$) can represent the error of orientation estimation. The angular difference between these two vectors can be calculated in the form of a quaternion denoted by Δq_A, as shown in Equation 4.12, where its first three components are the three components of a vector $\vec{q}_{Av}$ which is the cross product of the *calculated gravity vector* ($\vec{a}(q_G)$) and the *measured gravity vector* ($\vec{a}_0$). The fourth component (real part) of the difference in quaternion Δq_A is a scalar q_{Aw} which can be calculated by using Equation 4.14.

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

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

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

$$\boxed{\hat{q}_{GA} = q_G \otimes \Delta q_A} \tag{4.15}$$

To correct the orientation estimation (q_G), the rotation that represents the angular difference between *measured* and *calculated gravity vector* (Δq_A) is accumulated with q_G by means of quaternion multiplication. The estimated quaternion with gravity vector correction ($\hat{q}_{GA}$) is described in Equation 4.15, showing the quaternion multiplication of the orientation estimation (q_G) and the difference in quaternion between the *calculated gravity vector* and the *measured gravity vector* (Δq_A). The orientation estimation with gravity vector correction ($\hat{q}_{GA}$) needs to be normalized before using it as a rotation operator, to continue the orientation estimation process.

4.3.2 Using The Magnetic North Vector

In this section, the magnetic North vector is used as the referencing vector to correct the estimated quaternion (q_G). A similar approach to the quaternion correction using gravity vector is performed. Even though the directions of the magnetic North vector at different locations on Earth are different, the magnetic North vector at any location is always pointing to the North magnetic pole. Thus, the magnetometer in the IMU should ideally measure the strength and the direction of the magnetic field that is applied to the IMU. Unlike the gravity vector from the accelerometer reading, the magnetometer would provide the direction of the magnetic North vector even the sensor is not in a static period. When the IMU module is rotating, the magnetic North vector is decomposed into each measuring axis in the sensor's body frame. Therefore, the referencing magnetic North vector in the Earth frame (M_{int}) can be

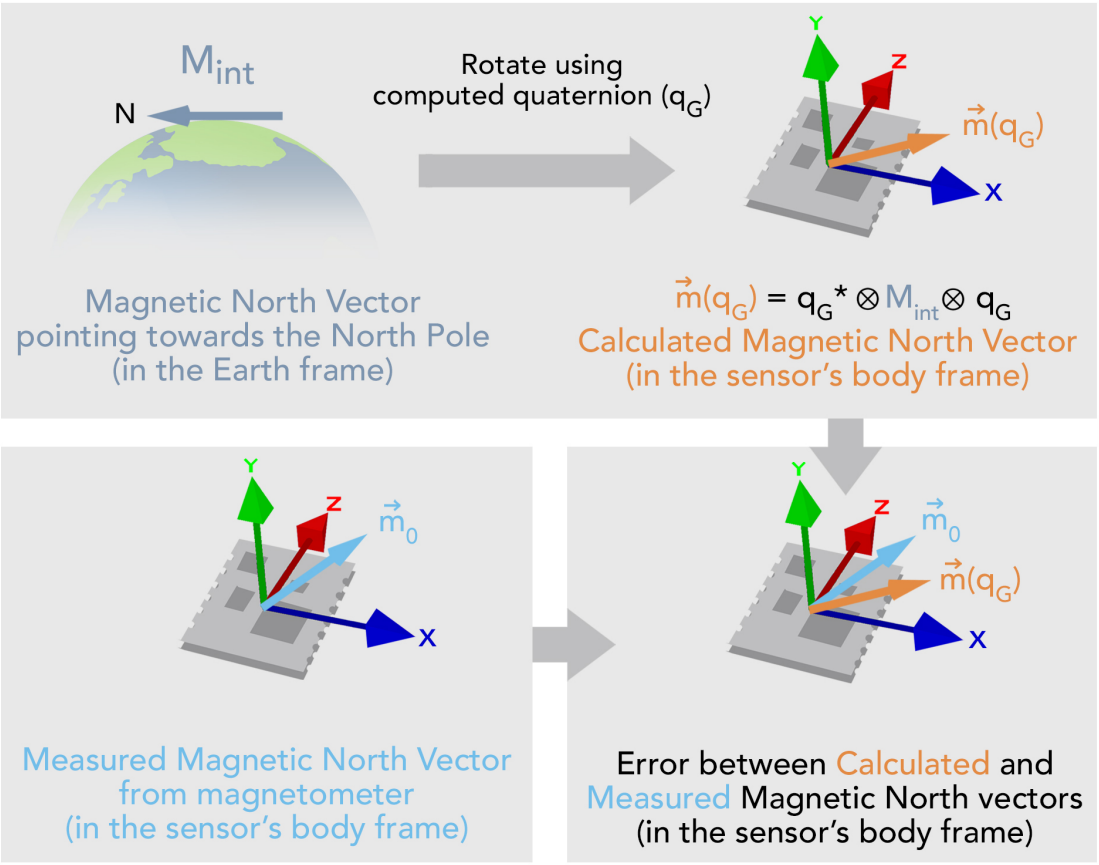


Figure 4.3: Quaternion correction using the magnetic North vector

rotated using the estimated quaternion (q_G) from Section 4.2. Equation 4.16 defines the calculation of the magnetic North vector referenced in the sensor's body frame ($\vec{m}(q_G)$) from the referencing magnetic North vector in the Earth frame (M_{int}). This *calculated magnetic North vector* ($\vec{m}(q_G)$) will then be used to correct the estimated quaternion by comparing with the *measured magnetic North vector* from the magnetometer readings ($\vec{m}_0$). The graphical representation of this approach is shown in Figure 4.3.

$$\vec{m}(q_G) = {q_G}^* \otimes M_{int} \otimes q_G \tag{4.16}$$

If there is no error in the orientation estimation (q_G), the *calculated magnetic North vector* ($\vec{m}(q_G)$) from Equation 4.16 would be equal to the *measured magnetic North vector* ($\vec{m}_0$) obtained from the magnetometer readings. The error between the *calculated magnetic North vector* ($\vec{m}(q_G)$) and the *measured magnetic North vector* ($\vec{m}_0$) can represent the error of orientation estimation. The angular difference between these two vectors can be calculated in the form of a quaternion denoted by Δq_M, as shown in Equation 4.17, where its first three components are the three components of a vector $\vec{q}_{Mv}$ which is the cross product of the *calculated magnetic North vector* ($\vec{m}(q_G)$) and the *measured magnetic North vector* ($\vec{m}_0$). The fourth component (real part) of the difference in quaternion Δq_M is a scalar q_{Mw} which can be calculated by using Equation 4.19.

$$\Delta q_M = \mathbb{H}(\vec{q}_{Mv}, q_{Mw}) \tag{4.17}$$

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

$$q_{Mw} = \|\vec{m}_0\|\|\vec{m}(q_G)\| + \vec{m}_0 \cdot \vec{m}(q_G) \tag{4.19}$$

$$\boxed{\hat{q}_{GM} = q_G \otimes \Delta q_M} \tag{4.20}$$

To correct the orientation estimation (q_G), the rotation that represents the angular difference between *measured* and *calculated magnetic North vector* (Δq_M) is accumulated with q_G by means of quaternion multiplication. The estimated quaternion with magnetic North vector correction ($\hat{q}_{GM}$) is described in Equation 4.20, showing the quaternion multiplication of the orientation estimation (q_G) and the difference in quaternion between the *calculated magnetic North vector* and the *mea-*

sured magnetic North vector (Δq_M). The orientation estimation with magnetic North vector correction ($\hat{q}_{GM}$) needs to be normalized before using it as a rotation operator, to continue the orientation estimation process.

The step of quaternion corrections using both the gravity vector and magnetic North vector are performed simultaneously. While $\hat{q}_{GA}$ represents the estimated orientation that is fully corrected depending on the measurement of accelerometer, $\hat{q}_{GM}$ represents the estimated orientation that is fully corrected based on only the measurement of magnetometer. Quaternion interpolation will be used to determine the final estimated orientation based on the values of $\hat{q}_{GA}$ and $\hat{q}_{GM}$, which will be discussed in section 4.4.

4.4 Quaternion Interpolation

By correcting the gyroscope data using the estimated bias offset error and correcting the estimated quaternion using only the gravity vector, the orientation correction algorithm is able to improve the orientation tracking of the human hand using IMUs [47]. But in some situations, the accelerometer does not measure only acceleration due to gravity but also includes the acceleration due to linear motion, when the sensor is moving. Therefore, in this case, the accelerometer no longer provides a reliable representation of the gravity vector in the sensor's body frame. In contrast, the rapid movement of the hand does not directly affect the magnetometer measurements. On the other hand, in some circumstances, the magnetic field around the user might not be completely uniform, due to the presence of ferromagnetic objects.

4.4.1 Spherical Linear Interpolation

During the moments when the hand is not in a static situation (i.e. the hand is moving). The orientation estimation should depend more on the orientation estimates using magnetic North vector correction ($\hat{q}_{GM}$) than the orientation estimates using gravity vector correction ($\hat{q}_{GA}$) because the measurements of the accelerometer include the linear acceleration due to the rapid translation applied to the sensor module. A suitable approach to determine the final orientation estimates from two quaternions by using a parametric weight is the approach commonly used for creating the continuous rendering of 3D objects in computer graphic animation called Spherical Linear Interpolation (SLERP) or great arc interpolation [48].

$$SLERP(q_0, q_1, h) = \frac{q_0 sin((1-h)\Omega) + q_1 sin(h\Omega)}{sin(\Omega)} \tag{4.21}$$

$$cos(\Omega) = q_0 \cdot q_1 \tag{4.22}$$

Equation 4.21 defines the method of Spherical Linear Interpolation of two quaternions, which is the method to calculate the intermediate quaternion between any two quaternions (q_0 and q_1). The control parameter (h) indicates the interpolated points, which ranges from 0 to 1. When the control parameter (h) is close to 0, the output quaternion will have a tendency towards the starting rotation (q_0), and when the control parameter (h) is close to 1, the output quaternion will have a tendency towards the ending rotation (q_1). The control parameter in the Spherical Linear Interpolation divides the interpolated rotation linearly. This interpolation method also preserves the magnitude of a unit quaternion and all the interpolated rotations lie on the same great arc [48].

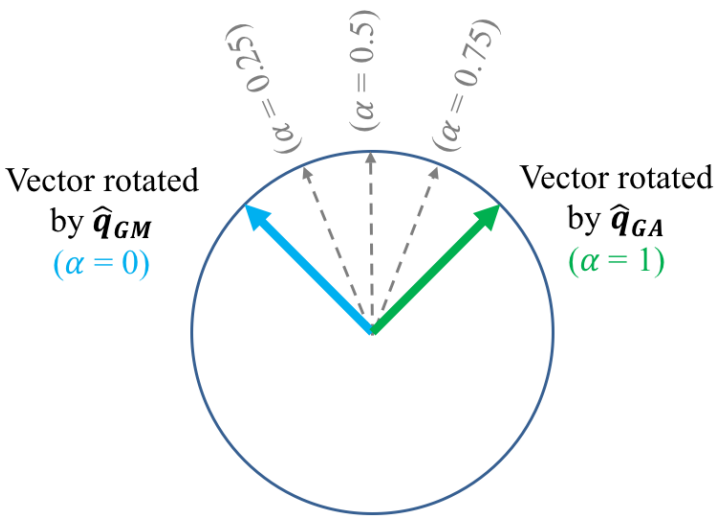


Figure 4.4: Spherical linear interpolation between $\hat{q}_{GM}$ and $\hat{q}_{GA}$

4.4.2 The Sensor's Stillness and The Control Parameter (α)

The Spherical Linear Interpolation is adapted to use for the calculation of the final estimated orientation ($\hat{q}_{OUT}$), which is the rotation interpolated between the orientation estimate using magnetic North vector correction ($\hat{q}_{GM}$) and the orientation estimate using gravity vector correction ($\hat{q}_{GA}$), as described in Equation 4.27. The angle of the interpolated arc subtended by $\hat{q}_{GM}$ and $\hat{q}_{GA}$ is denoted by Ω, where $cos(\Omega)$ equals to the 4-dimensional inner product of $\hat{q}_{GM}$ and $\hat{q}_{GA}$. The control parameter of quaternion interpolation (α) is the parameter that indicates the stillness of the sensor module and can be used to interpolate two quaternions linearly as visualized in Figure 4.4.

To determine the value of the quaternion interpolation's control parameter (α), the first-order Gamma memory filter is used to smoothen the signal of *Confidence Value* provided by the Yost Labs 3-Space Sensor, which indicates the stillness state of the IMU. When the control parameter μ in a Gamma memory filter has a value

between 0 to 1, the filter performs low-pass filtering to the input signal [49]. The transfer function of the first-order Gamma memory filter in z-domain [50] is shown in Equation 4.23.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\mu}{z - (1 - \mu)} \tag{4.23}$$

$$Y(z) = (\mu)z^{-1}X(z) + (1 - \mu)z^{-1}Y(z) \tag{4.24}$$

$$y[n] = (\mu)x[n - 1] + (1 - \mu)y[n - 1] \tag{4.25}$$

$$\alpha[n] = (\mu)(Confidence\ Value_{avg})^2[n - 1] + (1 - \mu)\alpha[n - 1] \tag{4.26}$$

The first-order Gamma memory filter output ($y[n]$) is derived from the filter's transfer function, as shown in Equation 4.25. In Equation 4.26, the value of the quaternion interpolation's control parameter (α) is substituted as the output of the Gamma memory filter, having the square of the average of *Confidence Value* as the input of the filter. The filter control parameter (μ) is set to 0.25 in order to perform signal smoothing and prevent a large amount of the delay.

Consider Equation 4.27, when α equals to 1, which indicates that the sensor is in a static period (not moving), it will make the first term ($\hat{q}_{GM}sin((1 - \alpha)\Omega)$) be equal to zero. Therefore, the final estimated quaternion orientation is equal to the orientation estimation using only the gravity vector correction ($\hat{q}_{GA}$). When the sensor is in motion, the accelerometer will also measure the acceleration due to linear motion. Accordingly, the value of (α) drops and has a tendency towards zero. Thus, the final estimated quaternion is the interpolated orientation that tends towards the orientation estimates using magnetic North vector correction ($\hat{q}_{GM}$).

$$\hat{q}_{OUT} = \frac{\hat{q}_{GM} sin((1-\alpha)\Omega) + \hat{q}_{GA} sin(\alpha\Omega)}{sin(\Omega)} \tag{4.27}$$

$$\Omega = cos^{-1}(\hat{q}_{GM} \cdot \hat{q}_{GA}) \tag{4.28}$$

This algorithm to estimate the orientation in quaternion form is also visually described in the block diagram shown in Figure 4.1. The red arrows in the diagram indicate the measurements of angular velocity ($\vec{\omega}_0$), acceleration due to gravity ($\vec{a}_0$) and the direction of magnetic North vector ($\vec{m}_0$) from the gyroscope, accelerometer and magnetometer in the IMU, respectively. The final orientation estimates ($\hat{q}_{OUT}$) is also fed back for the calculation of the quaternion estimates in the initial stage (q_G), for the next iteration.

CHAPTER 5

HAND MOTION TRACKING SYSTEM SETUP

This chapter describes how the hand motion tracking system has been set up. The hand motion tracking system using infrared cameras and inertial measurement units is able to detect hand position and orientation in three-dimensional space. The infrared cameras are capable of determining the Cartesian coordinates of an infrared-reflective marker attached to a glove worn by the user. Simultaneously, the inertial measurement units attached to the glove, which consist of tri-axial gyroscope, accelerometer and magnetometer, will be used to determine the orientation of the hand, proximal phalange and middle phalange of the index finger. The hand motion tracking system is designed to combine both position and orientation tracking data acquired from the infrared cameras and the inertial measurement units and apply them as the transformation (position and orientation) of a pre-designed 3D hand model. The overview of the hand motion tracking system is visually described, as shown in Figure 5.1. The visualization of the 3D hand model is implemented within Unity3D. The 3D hand model is capable of representing the position and orientation information acquired from the infrared cameras and the inertial measurement units in real-time while the hand is in motion. The visualization of the hand is updated accordingly to the actual hand movement. In the this research, we use the OptiTrack V120: Trio infrared cameras for position tracking and Yost Labs 3-Space sensors for orientation tracking.

5.1 Position Tracking using OptiTrack V120: Trio

V120: Trio is the optical tracking technology from OptiTrack that consists of three infrared sensing units which are aligned in the horizontal direction, within a single

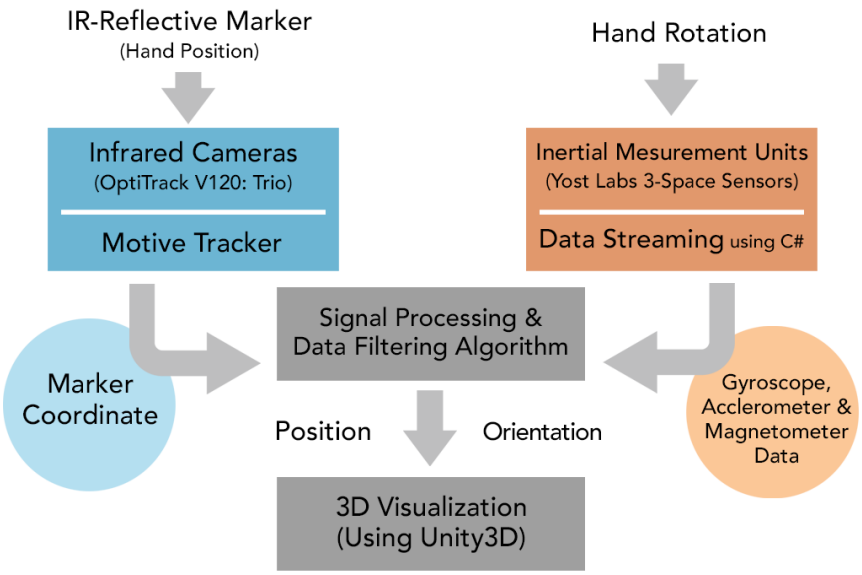


Figure 5.1: Overview of hand motion tracking system using inertial measurement units and infrared cameras

Figure 5.2: The setup of OptiTrack V120: Trio with Motive:Tracker software running on the host PC

module of following dimensions: 23 inches in length, 2 inches in width and 1.6 inches in height. Each infrared sensing unit in this system, is composed of an image sensing unit surrounded by a circular array of 26 infrared LEDs. Each image sensing unit is capable of providing an image resolution of 640×480 pixels with a frame rate up to 120 frames per second. Since the three infrared sensing units in the OptiTrack V120: Trio have fixed distances between each other and it was pre-calibreated by the manufacturer, the system is ready to use and no calibration is required. The OptiTrack V120: Trio can easily be connected the host PC via USB 2.0 port or later version and requires an operating power supply of 12 VDC with a current rating of 3A. The OptiTrack V120: Trio will be used to determine the position of the human hand in three-dimensional space. It has a field of view of 43 degrees in the vertical axis and 47 degrees in the horizontal axis. It can operate with a visible distance from 2 to 17 feet away from the device. The field of view (FOV) of the OptiTrack V120: Trio is illustrated in Figure 5.3.

For our hand motion tracking system, the OptiTrack V120: Trio is attached to a vertical stand, placing it above and behind the host PC monitor, as shown in Figure 5.2. The OptiTrack V120: Trio is operated using an engineering-grade rigid body and marker tracking software called Motive:Tracker. Five IR-reflective dot markers are attached around the wrist of the glove (as shown in Figure 5.4) so that only one marker would be visible to the infrared cameras at a time. The IR-reflective dot marker will appear as a single point in three-dimensional space, represented as a parent referencing point for the orientation of the hand and joints of the finger beyond the wrist. Within Motive:Tracker, there are options for the users to adjust the intensity of the infrared LEDs, the exposure of the sensing image and the detection threshold. Therefore, the unwanted reflections that may appear on the detected images apart from the reflections of the dot markers can be

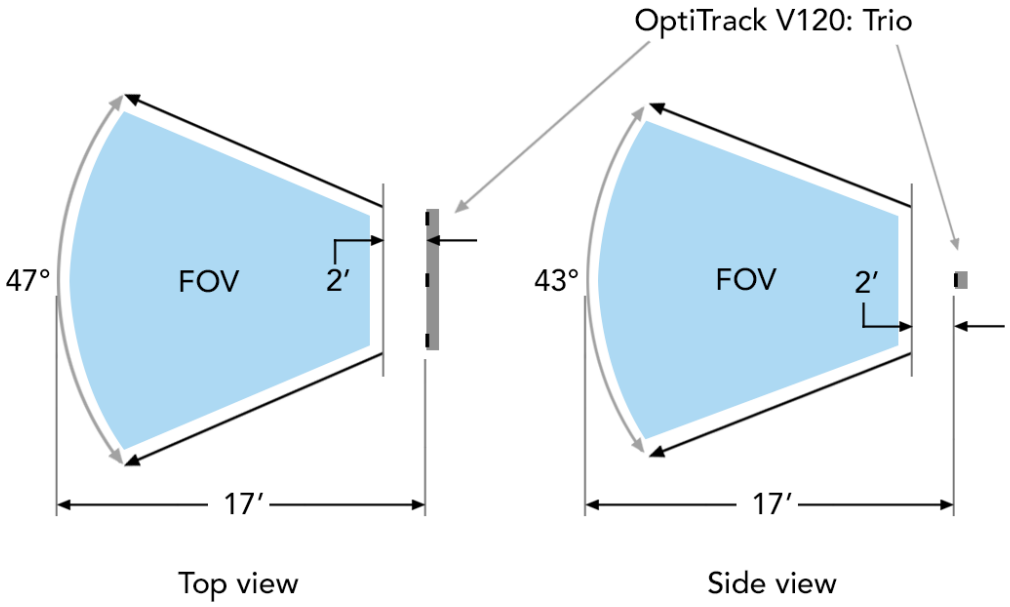


Figure 5.3: The field of view (FOV) of OptiTrack V120: Trio

Figure 5.4: The glove with IR-reflective dot markers attached on the wrist, one Yost Labs 3-space sensor attached on the back of the hand, and two Yost Labs 3-space sensors attached on the index finger

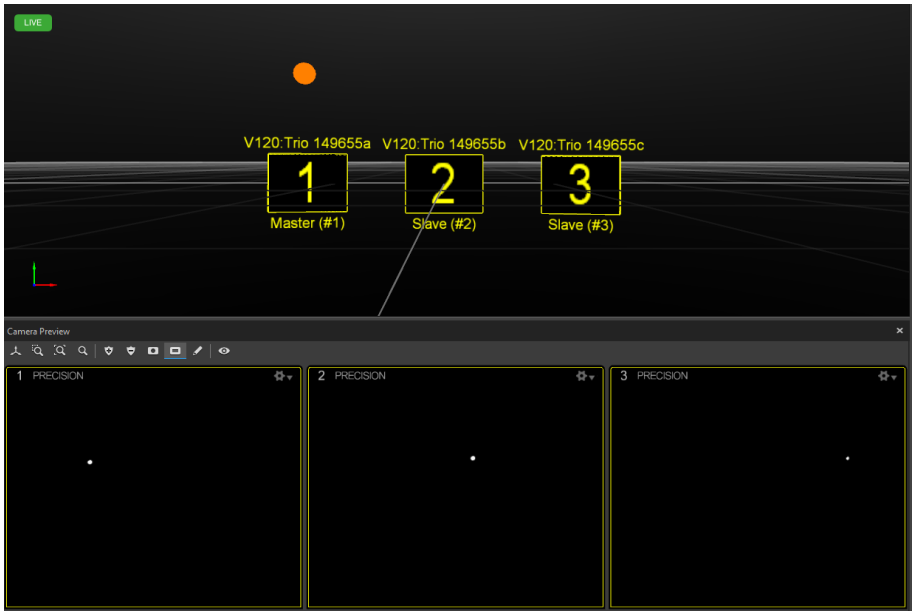


Figure 5.5: Visible dot marker on three infrared cameras (white dots) and position of the marker in 3D space shown as an orange dot

eliminated by adjusting those parameters. The dot markers attached around the wrist reflect the infrared light back to the image sensing units. When the silhouettes of the reflection are visible to all there infrared cameras, Motive:Tracker can then compute the position of the dot marker in the form of three-dimensional Cartesian coordinates (position in x, y and z) in real-time. Motive:Tracker would also be able to display a visualization of the detected marker in its 3D environment, as a single point (orange dot), as shown in Figure 5.5.

Motive:Tracker also allows the user to send out the coordinate data to other applications by using the provided software development kit (SDK) for cross-platform data streaming called NatNet. By using NatNet SDK, a console application was written in C++, to stream the marker coordinates detected within Motive:Tracker. This console application is able to establish the local connection to a NatNet server in Motive:Tracker. The application processes a marker data stream and encodes it

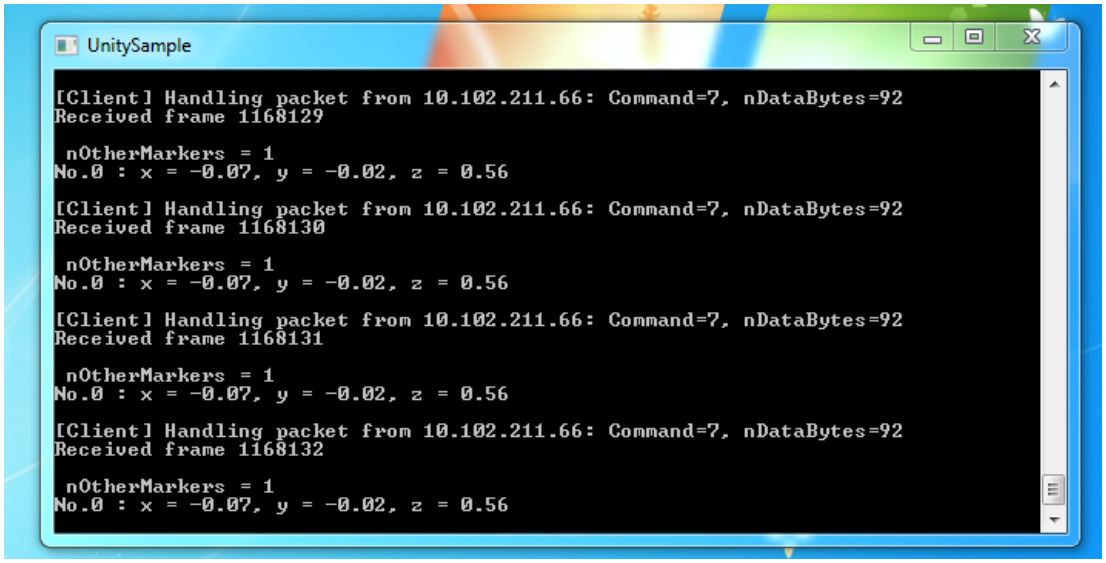


Figure 5.6: Console application to transport marker coordinate data from Motive: Tracker to Unity

to an XML document. Then, the application transports the XML document locally over UDP to Unity. A snapshot of the console application using NatNet SDK to stream marker coordinates is shown in Figure 5.6.

On the Unity side, additional scripts must be created in order to receive the XML document sent from Motive:Tracker NatNet server via the UDP connection. The script written in C# is created to parse the marker coordinate data from the streaming. That marker coordinate is considered as the position of the tracked hand in three-dimensional space. The raw marker coordinate data was adjusted with a position offset and then applied as the translation of the GameObject: 3D Hand model, as shown in Figure 5.7. The 3D hand model was previously prepared for the visualization of the hand motion tracking system.

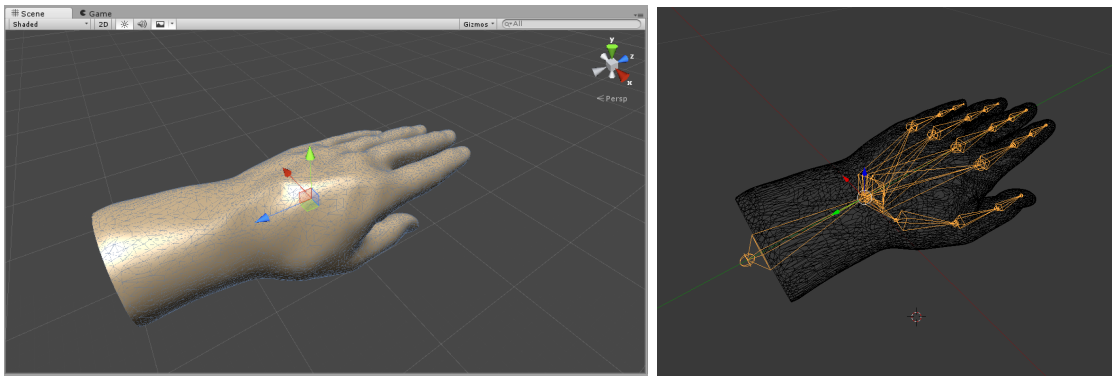


Figure 5.7: 3D hand model with rigged skeleton in Unity

5.2 Orientation Tracking using Yost Labs 3-Space Sensors

For our hand motion tracking system, three Yost Labs 3-Space sensors are used to determine the orientations. The first sensor is placed at the center of the back of the hand. The second sensor is attached to the glove at the location of the proximal phalange of the index finger and used to track the orientation of the index finger's proximal phalange with respect to the hand orientation. The last sensor is placed at the location of the middle phalange of the index finger in order to determine its orientation with respect to the index finger's proximal phalange. All three Yost Labs 3-Space sensors are attached on the glove as shown in Figure 5.4. Determination of the orientation of the furthest joint of the index finger, the distal phalange, does not require an additional (4^{th}) IMU. The orientation of the distal phalange can be calculated based on the angle of middle phalange. An empirical study and experimental observations [51] found the approximate orientation dependency of these two joints: The joint angle of the distal phalange is equal to two thirds of the joint angle of the middle phalange, as described in Equation 5.1. The illustration of the finger's joint angles and positions of three Yost Labs 3-Space sensors are shown in Figure 5.9.

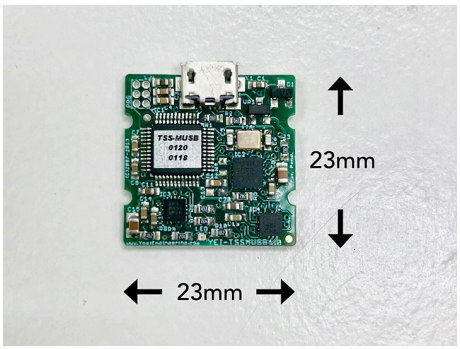


Figure 5.8: Yost Labs 3-Space^{TM} micro USB

$$\gamma_{distal} = \frac{2}{3}\beta_{middle} \tag{5.1}$$

The Yost Labs 3-Space^{TM} sensor, depicted in Figure 5.8, is a commercial-grade MEMS inertial measurement unit that consists of accelerometer, gyroscope and magnetometer. Each sensing unit is capable of measuring the body's inertia and the Earth's magnetic field in three-dimensional space. The Yost Labs 3-Space^{TM} sensor is a low-cost and ultra-miniature inertial measurement unit. Its dimensions are of 23mm×23mm×2.2mm and weighs only 1.3 grams. To operate, the Yost Labs 3-Space^{TM} sensor requires a DC supply of 3.3-6.0 volts. The data communications can be performed using USB 2.0 or through an asynchronous serial connection. The sensor provides several types of data which are raw, corrected or normalized data for the inertial measurements. Examples of measurements obtained from the sensor are linear acceleration, acceleration due to gravity, direction and strength of the Earth's magnetic field, angular velocity, etc. Moreover, this MEMS sensor also provides measurement of the temperature. The specifications of MEMS accelerometer, gyroscope and magnetometer contained in the Yost Labs 3-Space^{TM} sensor are shown in Tables 5.1, 5.2 and 5.3, respectively.

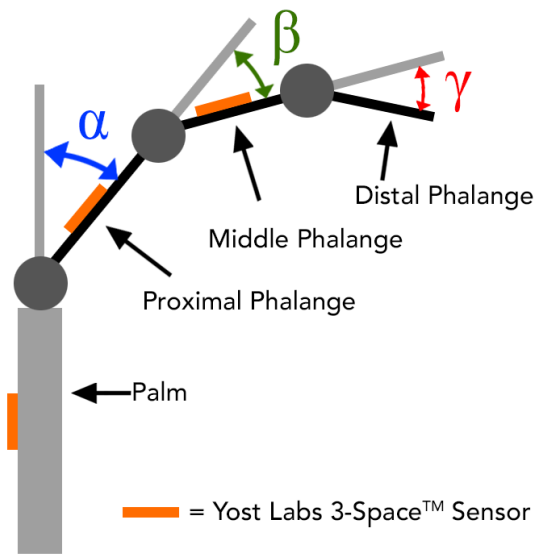


Figure 5.9: Finger joint angles and position of Yost Labs 3-Space™ sensors

Table 5.1: Accelerometer specifications

Specifications	Value
Scale	±2g / ±4g / ±8g selectable
Resolution	14 bit
Noise Density	$99\mu g/\sqrt{Hz}$
Sensitivity	0.00024g/digit-0.00096g/digit
Temperature sensitivity	±0.008%/°C

Table 5.2: Gyroscope specifications

Specifications	Value
Scale	±250/±500/±1000/±2000 °/sec selectable
Resolution	16 bit
Noise Density	$0.009°/sec/\sqrt{Hz}$
Sensitivity	0.00833°/sec/digit for 250°/sec 0.06667°/sec/digit for 2000°/sec
Temperature sensitivity	±0.03/°C
Bias stability @ 25°C	2.5°/hr average for all axes
Non-linearity	0.2% full-scale

Table 5.3: Magnetometer specifications

Specifications	Value
Scale	±0.88 Ga to ±8.1 Ga selectable (±1.3 Ga default)
Resolution	12 bit
Sensitivity	0.73 mGa/digit
Non-linearity	0.1% full-scale

Three Yost Labs 3-Space^{TM} sensors are connected to the host PC using micro USB to type-A USB cables. All three sensors were calibrated using Yost Labs 3-Space Software Suite before being used to implement the orientation tracking. To acquire the sensor's data, a set of specific commands has to be sent to the sensor. The set of commands can be sent in the form of binary codes or ASCII characters. Within Unity, a C# script was written to send a set of streaming commands to all three sensors and also receive from them the streamed measurement data and timestamps, via serial communication ports. The script parses the streaming data into timestamps (4 bytes), sensor's confidence value (4 bytes), accelerometer data (12 bytes), angular velocity (12 bytes) and magnetometer data (12 bytes). The orientation correction algorithm using gravity vector and magnetic North vector corrections was implemented within Unity using the parsed data from the streaming. Then, the output quaternions that estimate the orientations of the hand, proximal phalange, middle phalange and distal phalange of the index finger are applied as the rotations of the 3D hand model and its respective joints.

When the marker coordinate from OptiTrack V120: Trio is applied as the position of the 3D hand model, and the orientations of the hand and its respective joints are applied as the rotations of the same 3D hand model, simultaneously, the hand motion tracking system can then be used to determine the position and orientation of the human hand in three-dimensional space.

CHAPTER 6

IMPLEMENTATION OF THE ORIENTATION CORRECTION ALGORITHM

Before it was used for the real-time hand motion tracking system, the orientation correction algorithm, as explained in Chapter 4, was implemented offline, using MATLAB in order to validate its robustness. In the early stages of the study, we proposed the idea to correct the orientation using only bias offset estimation, quaternion estimation and quaternion correction using the gravity vector (not involving the use of the magnetometer). The algorithm was able to reduce the drift and improve the orientation tracking [47]. However, the accelerometer also measures the acceleration due to linear motion, when the sensor is in motion. Accordingly, the accelerometer becomes an unreliable source for orientation correction, under those circumstances. Later in our study, the direction of Earth's magnetic field measured through the magnetometer was used as a secondary reference vector when there is a rapid motion applied to the measurement unit [52]. This is appropriate because the magnetometer does not measure the body's inertial properties. The implementation of the orientation correction algorithm has evolved throughout the study and will be presented in this chapter in the chronological order of its evolution.

6.1 Implementation of Orientation Correction Algorithm Using Gravity Vector

In this stage of the study, the orientation correction algorithm consisted of only bias offset estimation, quaternion estimation, and quaternion correction using the gravity vector as depicted in the diagram shown in Figure 6.1. The key element

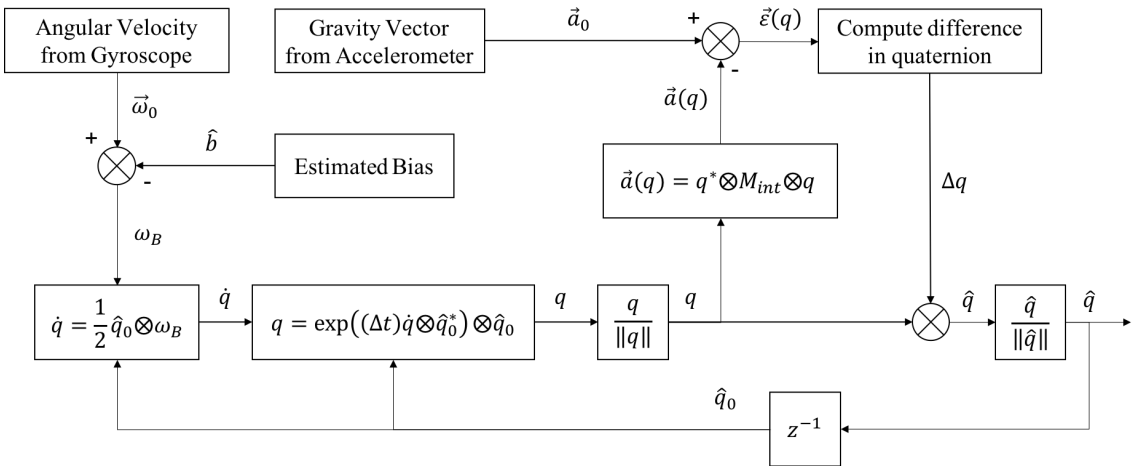


Figure 6.1: Block diagram of the orientation correction algorithm using only gravity vector correction

to effectively apply the orientation correction algorithm to hand motion tracking application is to be capable of implementing the algorithm in real-time.

6.1.1 Implementation

The angular velocity from the gyroscope and the acceleration measurement from the accelerometer were streamed and the recordings were stored in a text file with a sampling rate of 260 samples per second. The data was imported into the MATLAB workspace, and the algorithm was also implemented using MATLAB. In order to mimic the real-time constraints, each iteration of the algorithm was aware of only one sample of data at a time. The reason for implementing the algorithm in MATLAB is to easily evaluate the result obtained from the algorithm. The result of the estimated orientation in quaternion form was also exported to a text file in order to be visually evaluated using OpenGL 3D visualization. As shown in Figure 6.2, the implementation of the algorithm begins by calculating the maximum value in the three axes of the 50-sample window average of gyroscope data (*gyroMaxAvg*).

This value is the variable that was used to judge the level of movement of the sensor module. If this value is less than a pre-defined threshold value (*gyroThreshold*), the sensor module is considered to be not moving. To verify the static period of the sensor, this condition has to be true for 25 consecutive samples of gyroscope data only then the module will be considered static and a new predicted bias error will be calculated. Otherwise, the unbiased angular rate will continue to be determined by using the previous predicted bias error. The orientation quaternion is then calculated based on the unbiased angular rate.

The second part of the algorithm, which is the quaternion correction using gravity vector, used a different parameter to determine when sensor is static. This parameter is called stillness. Stillness is the value that reflects the movement of the sensor module, which ranges from 0 to 1. A high value of *Stillness* indicates that the sensor is in a static condition and thus, the accelerometer measurements will only or almost only contain the components of acceleration due to gravity. In our algorithm, if the value of *Stillness* is greater than pre-defined threshold (*stillnessThreshold*), the calculated gravity vector using the quaternion that represents the current orientation will be determined, and the difference in quaternion (Δq) with the measured gravity vector from the accelerometer will be determined. In the sensor's static period, where the *Stillness* is greater than *stillnessThreshold*, the estimated orientation will be updated by the accumulation of Δq to the pre-calculated quaternion (q) by means of quaternion multiplication. Otherwise, the estimated orientation will be equal to the pre-calculated quaternion (q). The Yost Labs 3-Space Sensor module was connected to a host PC via USB connection. A C program was written to stream the data from the sensor and store it in a text file. The evaluation experiment for this approach was performed by rotating the sensor module in different axes in order to verify the validity for any rotating movements. While the sensor module was

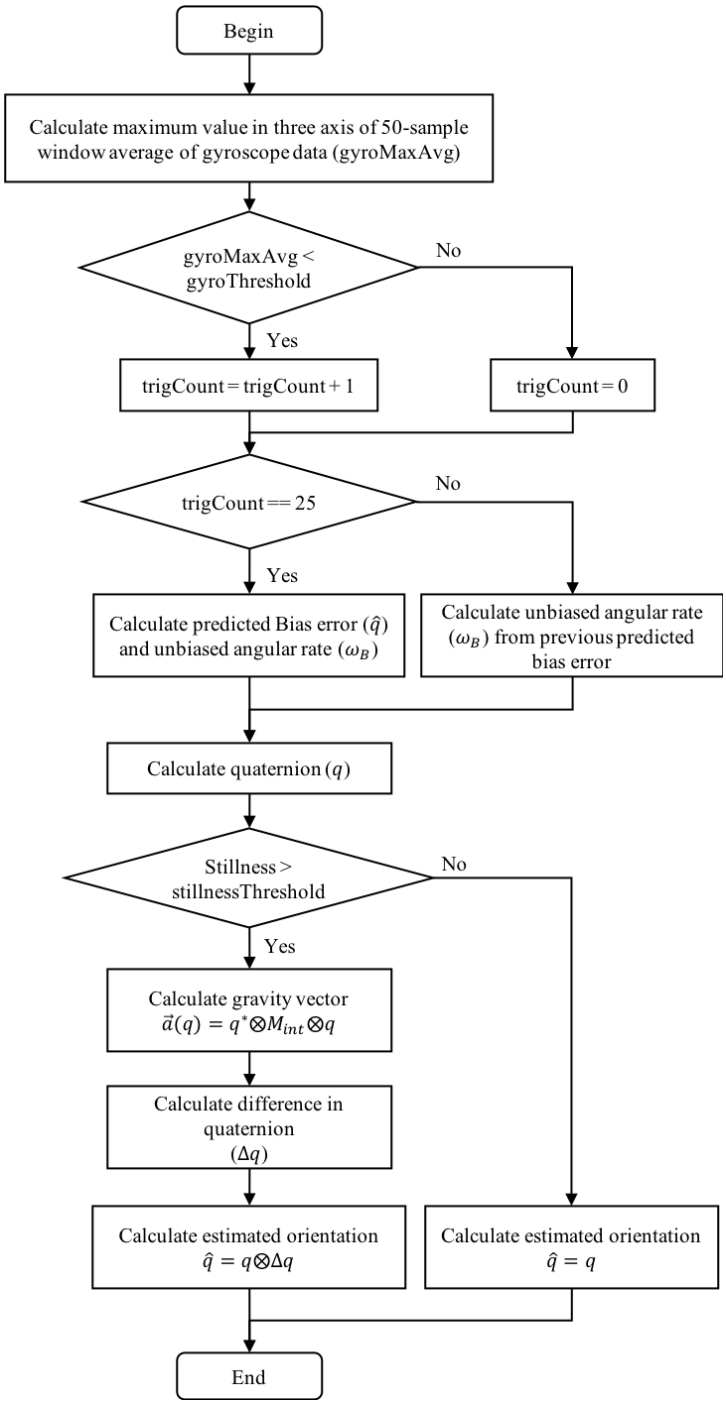


Figure 6.2: Flowchart showing the implementation of drift correction algorithm using only gravity vector compensation for one iteration with the condition that the module should be in a static period (Stillness)

rotating, angular velocity and acceleration were recorded with a sampling rate of 260 samples per second. The recorded data was processed by the proposed algorithm, following the flowchart in Figure 6.2.

6.1.2 Results

The gyroscope and acceleration data recorded from the experiment have the data length of 17,267 samples from the recording duration of 66.33 seconds. The first two plots in Figure 6.3 are the raw gyroscope data recordings in units of radians per second and the predicted bias errors for three coordinate axes (x, y and z) of the gyroscope. The third plot in Figure 6.3 is the orientation result presented as the computed quaternion using the unbiased angular rate value. This computed quaternion is the result of the orientation representation of the sensor module before applying the correction using gravity vector. The measured gravity vector from the accelerometer is shown in the first plot of Figure 6.4. Its vertical axis is the strength of gravity components (in units of g) in each coordinate axis of the sensor frame. The second plot in Figure 6.4 shows the difference of the measured gravity vector and the computed gravity vector using the computed quaternion (the 3 color traces correspond to the differences in the three body frame axes). The resulting estimated orientation in the form of quaternion after gravity-vector correction, is shown in the third plot of Figure 6.4. It consists of the four components of the quaternion, which can be used to describe the orientation of the sensor module.

Both of the quaternion results before and after gravity vector correction were exported as text files and used in 3D visualization to re-orient a pre-constructed 3D model of the IMU board. Then, they were verified and compared with the top view of the actual sensor movement, as shown in Figure 6.5. Figure 6.5(a) shows the se-

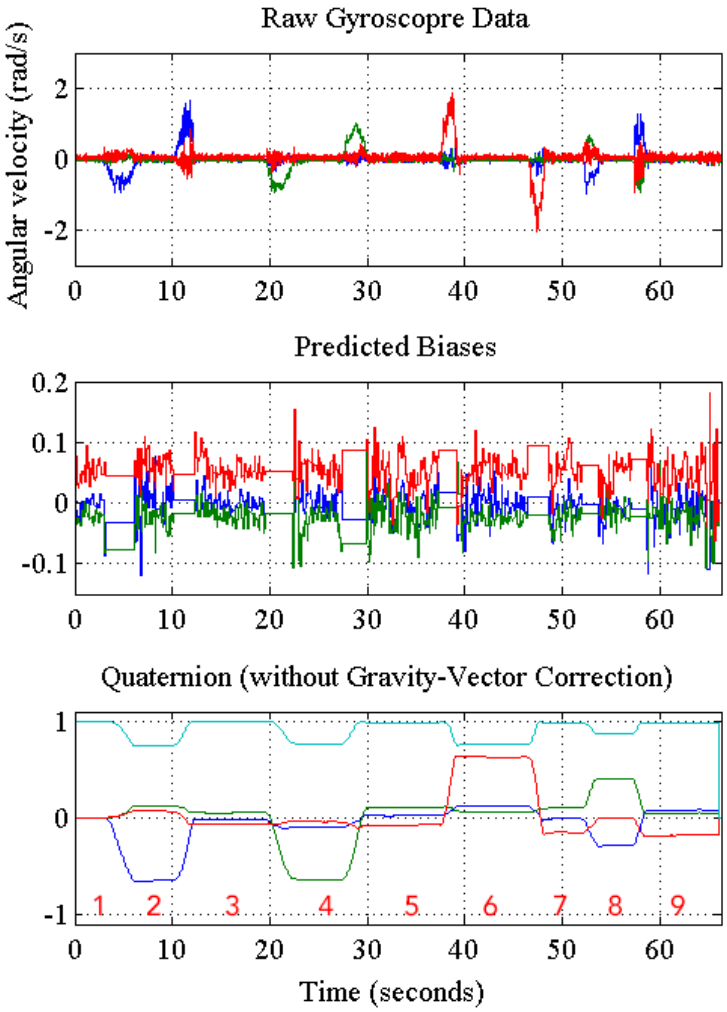


Figure 6.3: The plots of raw gyroscope data including bias offset error, predicted bias error, quaternion result (without gravity-vector correction). The numbers written at the bottom of the lower plot identify 9 "poses" or "stages" in which the module was held temporarily static.

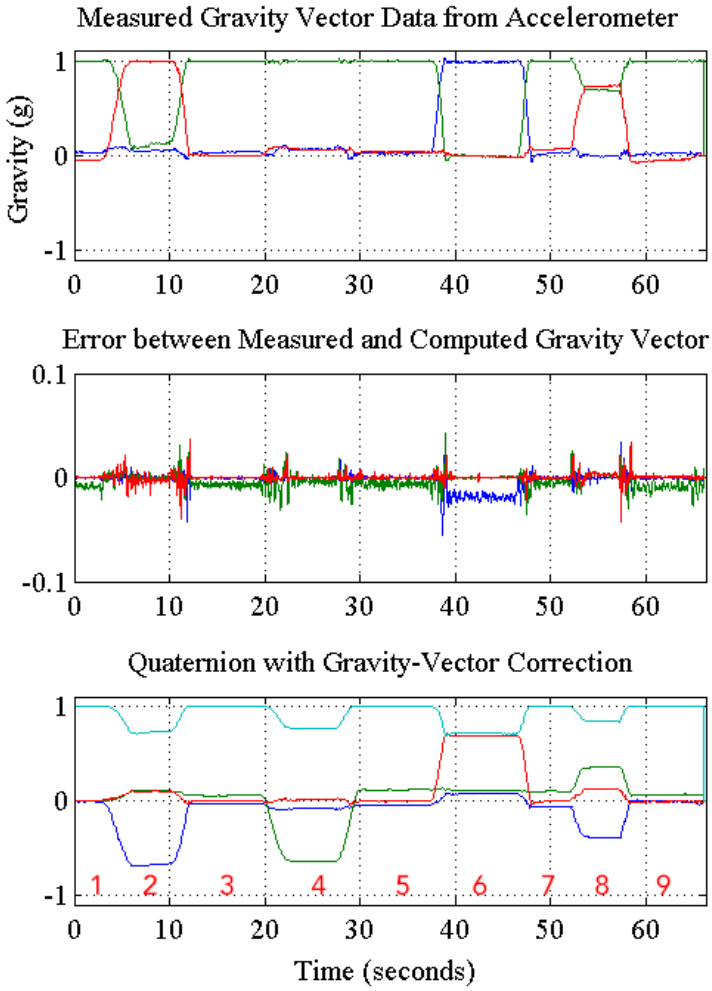


Figure 6.4: The plots of measured gravity vector from accelerometer, error between measured and computed gravity vector, estimated quaternion result (with gravity-vector correction). The numbers written at the bottom of the lower plot identify 9 "poses" or "stages" in which the module was held temporarily static.

quence of 9 actual orientations of the sensor module captured during the recording. Figure 6.5(b) is the 3D visualization using the orientation quaternion before gravity vector correction. Figure 6.5(c) shows the estimated orientation sequence of the sensor module rotation using the estimated quaternion after gravity vector correction. The 9-stage sequences of orientation in Figure 6.5(b) and (c) are the visualization of the results in the third plots of Figures 6.3 and 6.4, respectively. The numbers (1-9) in the first column of Figure 6.5 correspond with the (1-9) intervals labeled at the bottom of Figures 6.3 and 6.4. These "poses" or "stages" were shot intervals in which the module was held approximately static.

The results obtained by predicting the bias offset error shown in Figure 6.3 lead us to support the use of this approach every time the sensor module is static to remove the bias offset error in the gyroscope reading and produce less drift in orientation. Notice that during the sensor's static periods, the algorithm calculated the predicted bias offset error. But when the sensor was rotating, the algorithm held the previous value of the bias offset. The result of orientation in the form of quaternion (q) indicated that we can approximate the orientation of the sensor module but we can still observe some residual error that deviated from zero in some axes. This could be because the bias offset error prediction might be over-compensating, yielding error levels that still impacted the orientation results. The second part in the algorithm that could help solving this problem is the quaternion estimation using gravity vector correction. THe following paragraphs provide additional discussion of the results of this experiment, shown in Figures 6.3, 6.4 and 6.5

The measured gravity vector, referenced in the sensor frame, is shown in the first plot of Figure 6.4. This measurement indicates the direction of the gravity vector in the sensor frame, affected by the rotation of the sensor module. When the sensor is static, this measurement reflects only the acceleration due to gravity. The

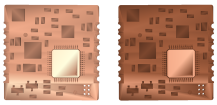
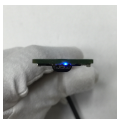
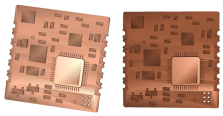
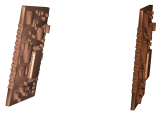
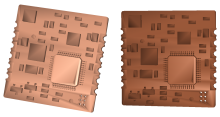
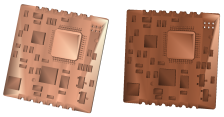
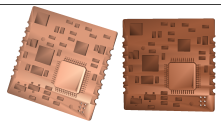
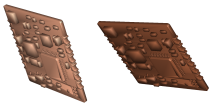
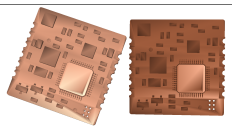
Sequence	(a)	(b) (c)
1		
2		
3		
4		
5		
6		
7		
8		
9		

Figure 6.5: Comparison between sequences of (a) actual sensor module orientation at each of the 9 "poses" or "stages" identified in Figures 6.3 and 6.4, (b) 3D visualization of orientation using computed quaternion and (c) 3D visualization of estimated orientation after gravity vector correction

calculated gravity vector using the quaternion result in the first part can be used to compare with this gravity vector measurement. The error between the measured and calculated gravity vectors, shown in Figure 6.4, provides the mechanism to determine the error between two different sources of measurement (e.g. gyroscope and accelerometer). We can see that with this idea, we could also determine the sensor orientation using the measurement of acceleration due to gravity measured in the sensor's body frame. The difference between two vectors can be represented in the form of rotation. In other words, we could achieve the expected orientation of the sensor if we can determine how much we have to rotate the calculated gravity vector to match the measured gravity vector. The angular difference between these two vectors was determined and represented as quaternion (rotation). This angle difference was then included in the previous calculated quaternion and resulted in the estimated orientation of the sensor module as shown in Figure 6.4. The result shows that the method of using the gravity vector to correct the orientation estimation can help to improve the output. The inclination through time of the quaternion result, which is the indication of drift problem, was reduced using the gravity vector as the reference, as can be seen in the quaternion result of Figure 6.4.

To verify these results, the quaternion output from both processes (before and after application of the gravity vector correction method) were applied to the rotation of the 3D model (IMU model) as shown in Figures 6.5(b) and 6.5(c). Comparing both results to the actual movement that had been captured as the pictures in Figure 6.5(a), leads us to confirm that the estimated quaternion after gravity vector correction ($\hat{q}$) provides a better result than before we applied the correction. This visualization shows that the 3D model using the original computed quaternion in Figure 6.5(b) becomes misaligned after some rotations were applied to the sensor module failing to re-align to its initial orientation in stage 9. For example, in Figure

6.5, the rotation from stage 1 to stage 2 is the rotation in x-axis by -90 degrees and then from stage 2 to stage 3, the sensor module was rotated back 90 degrees in x-axis. The quaternion result in the first part of the algorithm is clearly mismatched with the actual movement while the 3D model (with gravity-vector correction) in Figure 6.5(c) improves the orientation. The images captured at different stages of the rotating sequence can help to visually verify the improvement of orientation estimation for the IMU module achieved using the prediction of the bias error and the gravity vector correction.

6.1.3 Verification of Orientation Correction Algorithm on Hand Orientation Tracking

An IMU was attached on the glove, at the back of the hand. The angular velocity and acceleration measurement were streamed from the IMU and stored in a text file with a sampling rate of 370 samples per second while a sequence of hand movements was performed. The data was imported into the MATLAB workspace, and the orientation correction algorithm, based on the diagram in Figure 6.1, was implemented. The idea of correcting the orientation estimate and the gyroscope offset value every time the sensor is not moving is well-suited to the hand motion tracking application, as the human hand is not expected to be moving all the time.

The gyroscope and acceleration data recorded in this evaluation contains 7,581 samples of each variable (duration: 20.51 second). In Figure 6.6, the first plot is the angular velocity recorded from the gyroscope (raw data). The second plot shows the 4 quaternion components resulting from the first part of the algorithm (without gravity vector compensation). The last plot shows the 4 estimated quaternion com-

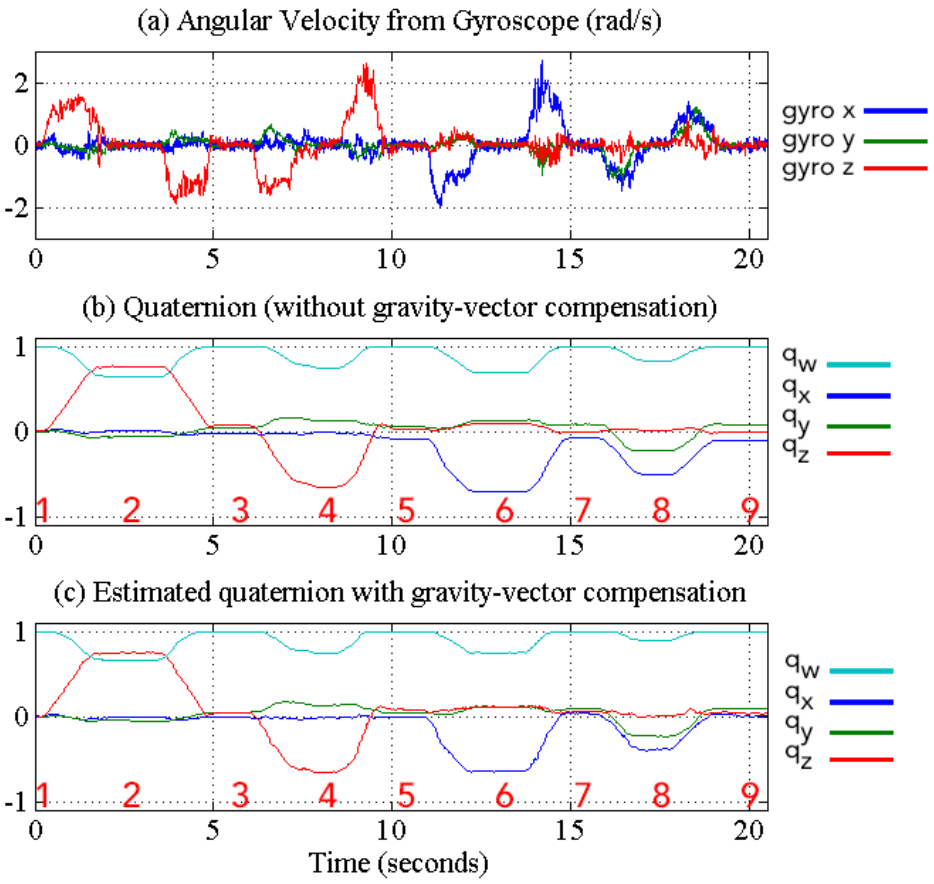


Figure 6.6: (a) Angular Velocity, (b) Quaternion without gravity vector compensation, (c) Estimated quaternion with gravity vector compensation

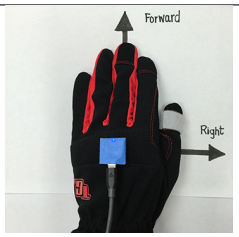
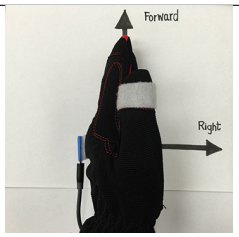
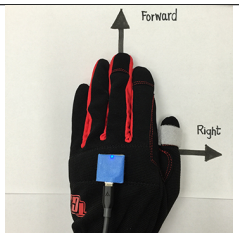
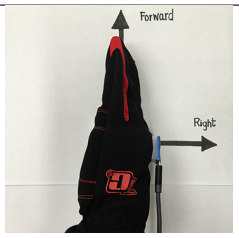
Sequence	(a)	(b)	(c)
1			
2			
3			
4			

Figure 6.7: Comparison between (a) sequences of actual hand orientation, (b) 3D visualization of hand orientation using computed quaternion and (c) 3D visualization of estimated hand orientation after gravity vector compensation. (Stages 1 to 4) [All the pictures in column (a) are taken from the top, except #6 and #8, which are taken in the back-to-front direction. All the simulated hands are also viewed in the back-to-front direction.]

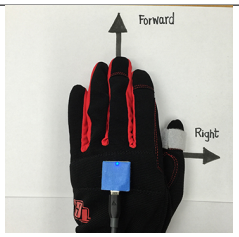
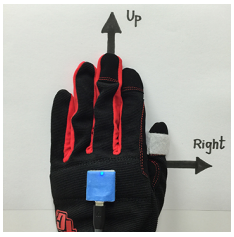
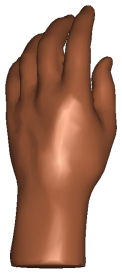
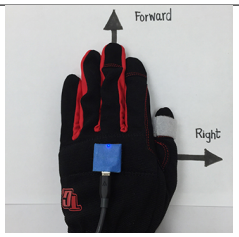
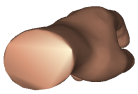
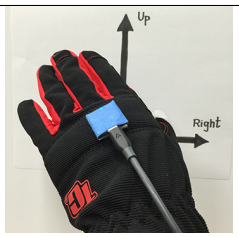
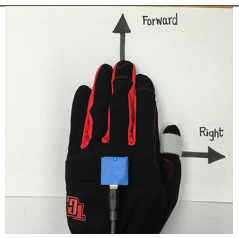
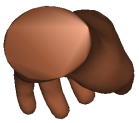
Sequence	(a)	(b)	(c)
5			
6			
7			
8			
9			

Figure 6.8: Comparison between (a) sequences of actual hand orientation, (b) 3D visualization of hand orientation using computed quaternion and (c) 3D visualization of estimated hand orientation after gravity vector compensation. (Stages 5 to 9) [All the pictures in column (a) are taken from the top, except #6 and #8, which are taken in the back-to-front direction. All the simulated hands are also viewed in the back-to-front direction.]

ponents after applying gravity-vector compensation to the orientation estimation process.

The quaternion output results from Figure 6.6(b) and Figure 6.6(c) were used to visualize the orientation of the hand by applying the rotations they indicate to a 3D hand model. The sequence of orientations in Figure 6.7 shows the actual hand orientation, and the corresponding hand visualization of the results from Figure 6.6(b) and from Figure 6.6(c), respectively. The numbers (1-9) used to label the rows of Figure 6.7 correspond to the (1-9) time points labeled at the bottom of Figures 6.6(b) and 6.6(c), which identify the 9 "poses" or "stages" at which the orientations of the real and simulated hands were captured.

The quaternion computed in the first part of the proposed algorithm (without gravity vector correction) can approximately represent the orientation of the hand. There was still some drift present in this result but it was better than just the integration of the angular velocity, without removing the bias offset error. Comparing the orientation estimates indicated by the original quaternion and the (gravity vector) corrected quaternion shows that the latter approximates the real hand orientation better. The drift in rotation was corrected and the orientation estimation was improved.

6.2 Implementation of Orientation Correction Algorithm Using Gravity Vector and Magnetic North Vector

This section describes the full implementation of the orientation correction algorithm fully implemented, the orientation correction algorithm includes the bias offset estimation, quaternion estimation, quaternion correction using the gravity vector, quaternion correction using the magnetic North vector and quaternion interpolation.

Figure 6.9: The glove with IMUs attached on the back of the hand and on the tip of index finger

The implementation of the algorithm is based on the diagram shown in Figure 4.1. In this stage of the study, the orientation correction algorithm was implemented on the data recorded from two IMU sensors which tracked the movement of the hand and the index finger.

6.2.1 Implementation

Two Yost Labs 3-Space Sensor modules were attached on the back of the hand and on the tip of the index finger as shown in Figure 6.9. The gyroscope, accelerometer and magnetometer data were recorded and stored in a text file while a sequence of hand motions were being performed. The orientation correction algorithm was applied to the recorded data within MATLAB. During the implementation of the algorithm in MATLAB, the programming code was aware of only one sample of data at a time in order to imitate real-time performance.

The bias offset error must be estimated only when the IMUs detect periods of stillness. A flowchart showing the implementation of the orientation correction algorithm is displayed in Figure 6.10. The implementations of bias offset estimation

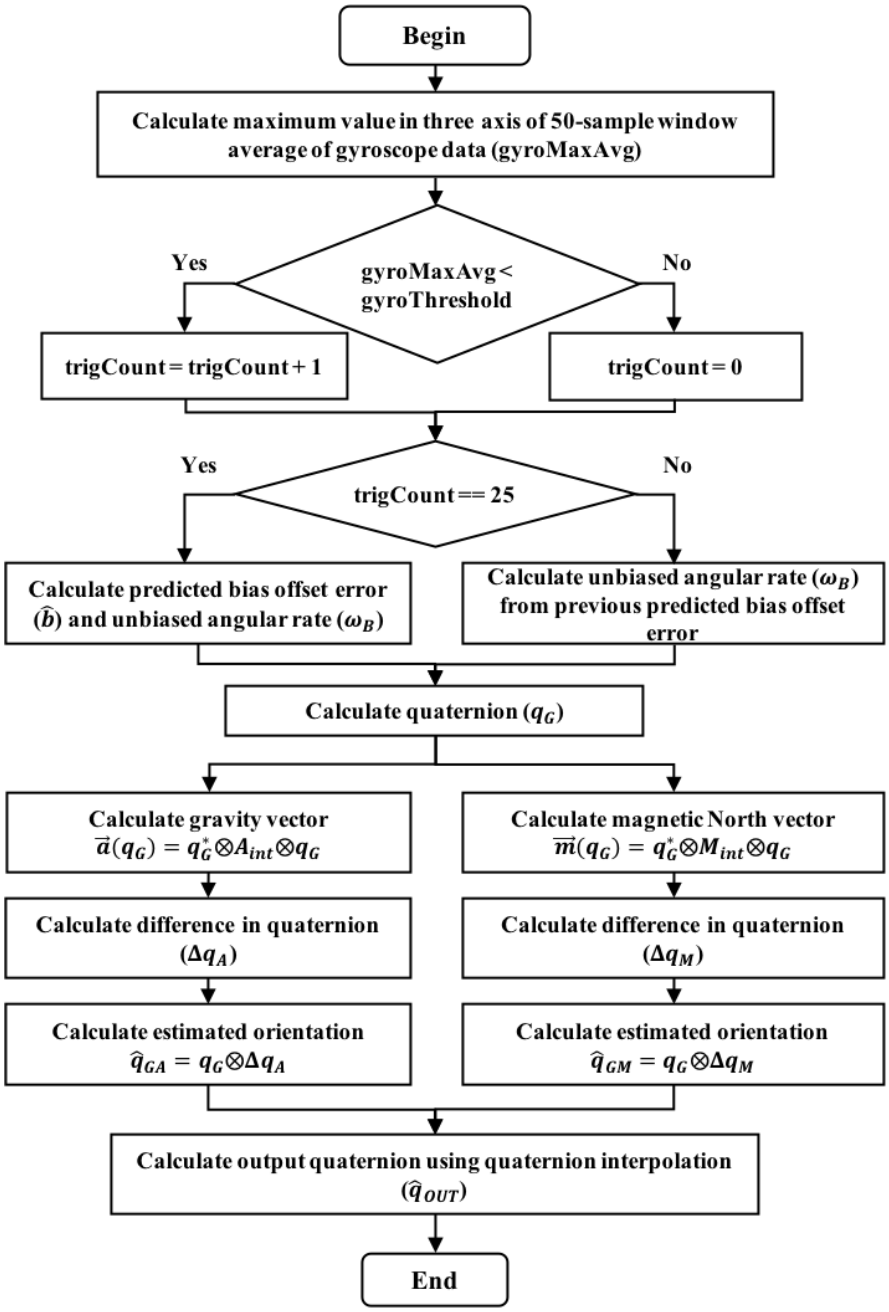


Figure 6.10: Flowchart showing the implementation of orientation correction algorithm for one iteration using both gravity and magnetic North vector corrections

and quaternion estimation are the same as for the implementation of the orientation correction algorithm using only gravity vector, in Section 6.1. In this section, The quaternion correction using gravity vector and magnetic North vector are implemented simultaneously. The output quaternion ($\hat{q}_{OUT}$) is calculated by using quaternion interpolation between the estimated quaternion correction using gravity vector ($\hat{q}_{GA}$) and the estimated quaternion correction using magnetic North vector ($\hat{q}_{GM}$). In our experiments, the resulting orientations were exported as a text file and visualized by the animation of a 3D hand model in Unity. The motion of the hand model and the index finger with corrected orientation was validated and compared with snapshots of the hand motion sequence.

6.2.2 Results

Figure 6.11(a) and Figure 6.12(a) show the orientation in quaternion for IMUs attached on the back of the hand and the index finger, respectively. They were recorded directly from the sensor modules for a duration of 22.348 seconds. These recordings are the quaternion output from the Kalman-based orientation filtering inside the Yost Labs 3-Space sensor module. The resulting estimated orientation after gravity vector and magnetic North vector correction for IMUs attached on the back of the hand and the index finger are shown in Figure 6.11(b) and Figure 6.12(b), respectively. Comparing the quaternion results from the orientation correction algorithm using gravity and magnetic North vector and the quaternion recording from the on-board Kalman-based filtering in Figure 6.11, it was found that the result from the algorithm using gravity and magnetic North vector correction can reduce the amount of drift that distinctly occurred in Kalman-based filtering estimation during the 2^{nd} to 5^{th} seconds and 17^{th} to 21^{st} seconds.

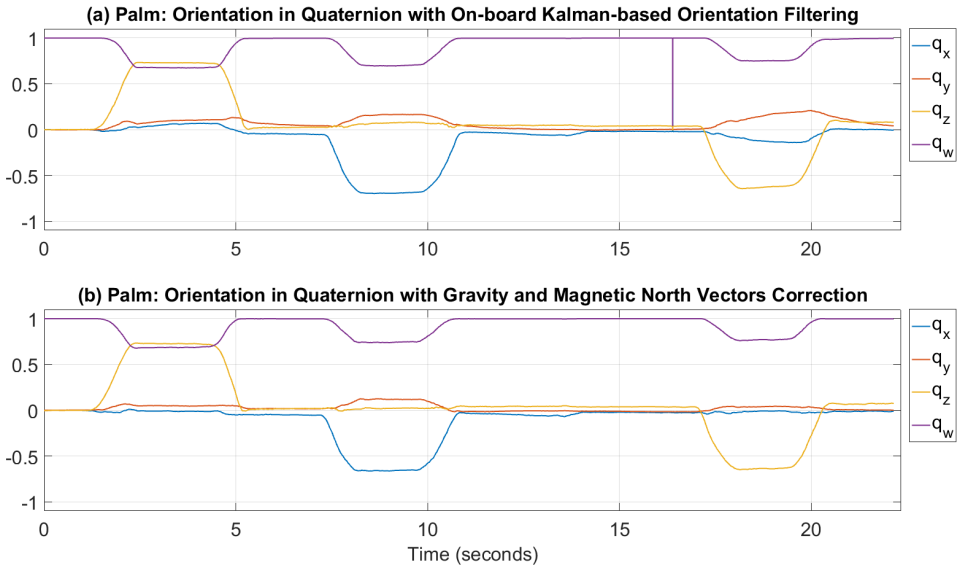


Figure 6.11: The plots of estimated quaternion results (a) with On-board Kalman-based Orientation Filtering and (b) with gravity and magnetic North vectors correction for IMU attached on the back of the hand

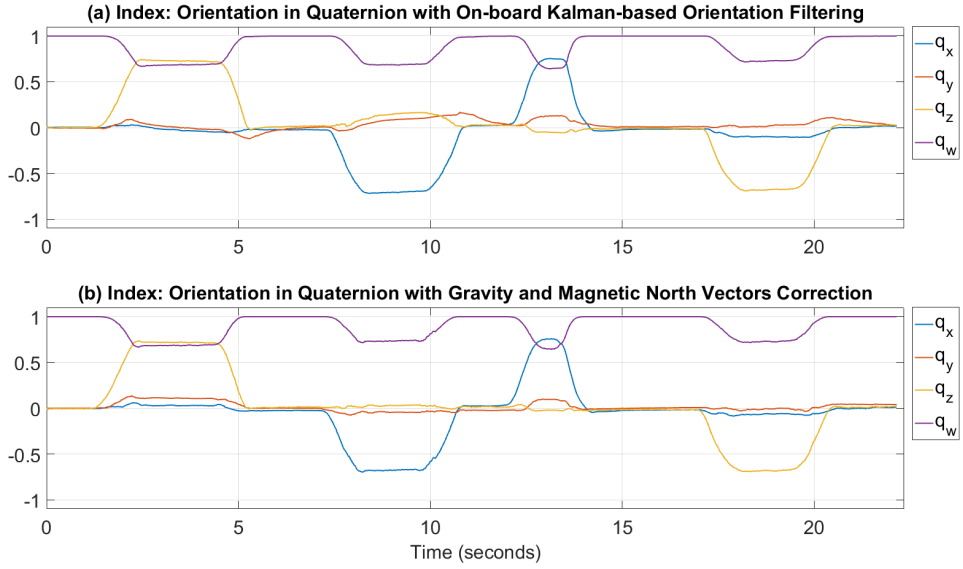


Figure 6.12: The plots of estimated quaternion results (a) with On-board Kalman-based Orientation Filtering and (b) with gravity and magnetic North vectors correction for IMU attached on the index finger

Similarly in Figure 6.12, the quaternion result of the IMU attached on the index finger from the orientation correction algorithm using gravity and magnetic North vector and the quaternion recording from the on-board Kalman-based filtering are compared. This figure shows that the result from the algorithm using gravity and magnetic North vector correction can also reduce the amount of drift in the quaternion result using Kalman-based filtering method that occurred during the 7^{th} to 11^{th} seconds, 12^{th} to 14^{th} seconds and 17^{th} to 21^{st} seconds of the recording.

To verify these results, the quaternion outputs obtained from the gravity and magnetic North correction algorithm were exported as text files and were applied to the rotation of the 3D hand model in Unity. Figure 6.13 shows the comparison between a sequence of actual photographs of the hand orientation and 3D visualizations of estimated hand orientation after gravity vector and magnetic North vector correction. Comparing the results to the actual hand orientation, leads us to confirm that the estimated quaternion after gravity vector and magnetic North vector correction provides an acceptable result of orientation tracking using the IMUs. The 9-stage sequences of 3D hand orientation in Figure 6.13 are the visualization of the quaternion results in Figure 6.11(b) and Figure 6.12(b). The numbers (1-9) to the left of the stages in Figure 6.13 correspond with the (1-9) intervals labeled at the bottom of Figure 6.11(b) and Figure 6.12(b). The similarity between the pictures and the 3D simulations in Figure 6.13 seems to confirm that the proposed approach to correct the drift in the gyroscope measurements and compensate the orientation estimation using the gravity vector and magnetic North vector will be useful in orientation tracking of human hand motion. Since the precise data of the hand or the fingers orientation can be obtained, this can lead to the improvement of 3D user interfaces to become more realistic. Thus, it could also enhance the experiences of the natural interaction of humans with a 3D virtual environment.

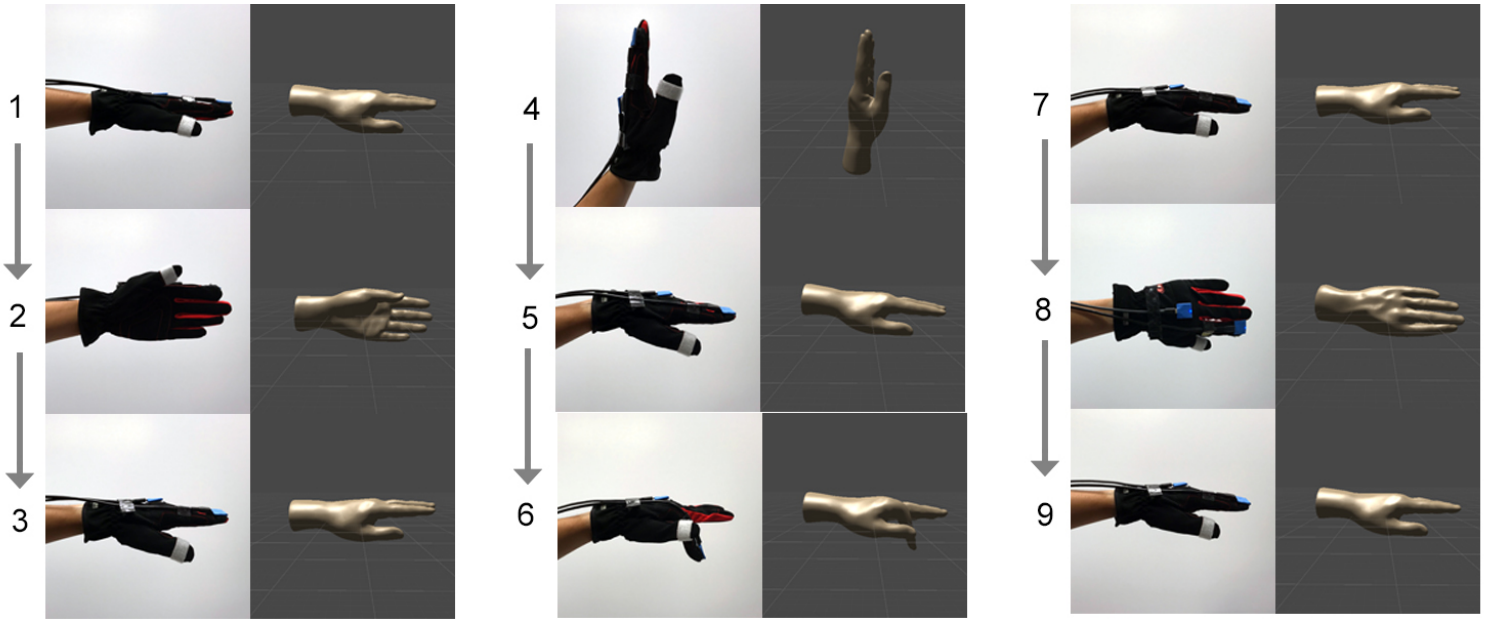


Figure 6.13: Comparison between a sequence of actual hand orientations and 3D visualizations of estimated hand orientation after gravity vector and magnetic North vector correction

CHAPTER 7

REAL-TIME IMPLEMENTATION OF HAND MOTION TRACKING SYSTEM

The hand motion tracking system using inertial measurement units and infrared cameras has been set up as explained earlier in Chapter 5. In this chapter, the real-time implementation of hand motion tracking will be presented. To validate the capability of the hand motion tracking system for 3D user interface purposes, a 3D environment was created. In this test environment, a human subject was prompted to perform a task that included the acquisition of several target objects using the hand motion tracking system in which the orientation correction algorithm was enabled. The evaluation process is explained in the following sections.

7.1 Creating 3D Environment in Unity

In order to verify the real-time performance of the orientation correction algorithm using gravity vector and magnetic North vector for the hand motion tracking interface, the algorithm described in Chapter 4 (Orientation Correction Algorithm) was implemented for real-time execution using a C# script within Unity. To evaluate the compatibility of this orientation correction algorithm on the hand motion tracking interface, a game scene (3D environment) in Unity was created, as shown in Figure 7.1.

The 3D hand model shown in Figure 7.1 was attached to the C# scripts that receive the streamed marker position in 3D space and raw accelerometer, gyroscope and magnetometer data. The marker position from OptiTrack V120:Trio were assigned as the position of the 3D hand model. The C# script that receives the streamed raw accelerometer, gyroscope and magnetometer data also implemented the orientation correction algorithm using gravity vector and magnetic North vec-

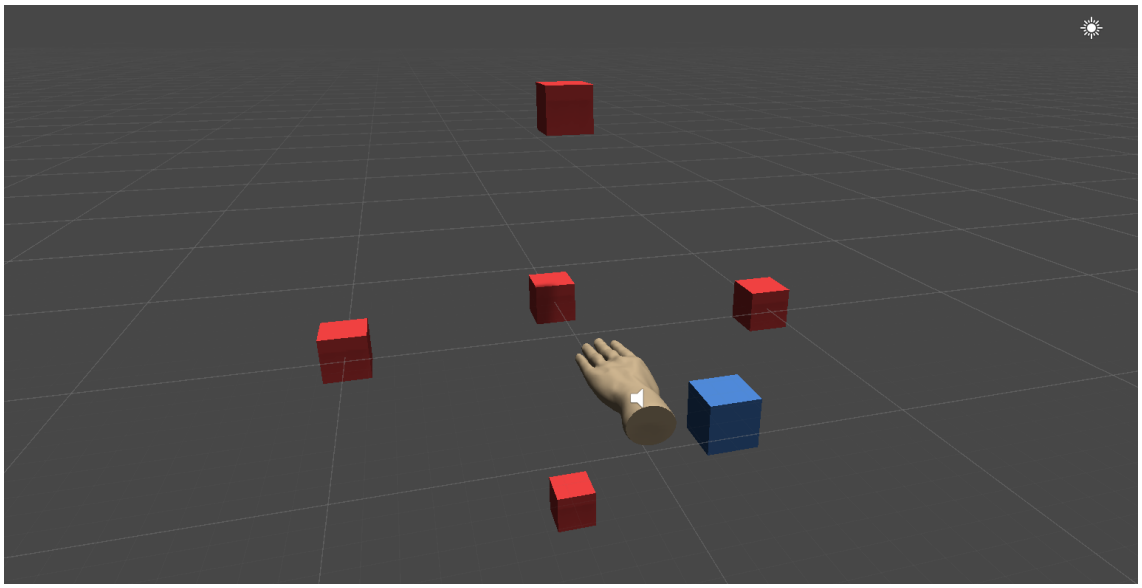


Figure 7.1: Unity game scene for testing real-time implementation of the orientation correction algorithm

tor. For every frame of rendering, the output estimated quaternion ($\hat{q}_{OUT}$) was calculated for the hand, proximal phalange, middle phalange and distal phalange. The quaternions were assigned as the rotations for the respective parts of the hand. By using the position tracking and orientation tracking results to animate the 3D hand model, a user will be able to complete the actions required in the virtual task proposed for the evaluation.

7.2 Evaluating The Hand Motion Tracking Interface Performance

An initial test to assess the strength of the bias offset error and the ability of the orientation correction algorithm to compensate for it was performed on a static basis. For this, three inertial measurement units (attached to the back of the hand, proximal and middle phalanges, in a glove) were just left resting on a table. In the

absence of motion, all the gyroscopes would be expected to output a value of zero. The observation, presented in the following sections did not meet this expectation.

To evaluate the performance of the hand motion tracking interface on a dynamic basis, a game scene with five red cubes (targets) and one blue cube (home position) was created as shown in Figure 7.1. The experimental subject wore the glove and was asked to perform a task to acquire the red cubes in 3D space. A single red cube (target) will appear in the scene after acquiring the blue cube (home position). The acquisition of the blue cube marks the starting time for the subject to acquire the red cube. The red cubes are placed in 3D positions that are at equal distances from the unique position of the blue cube (origin). The time to acquire each red cube was recorded. The blue cube appeared for the first time after the subject entered the Subject ID, as shown in Figure 7.2.

In order to acquire each of the cubes in this task, the subject has to flex his/her index finger while colliding with the cube. The 3D hand model will change its color to green when the flexion of the index finger is detected, as shown in Figure 7.3.

In every trial, after the blue cube is acquired, a red cube will appear. The subject will try to complete the trial by moving his/her hand in 3D space to reach the red cube and flex the index finger to acquire it as shown in Figure 7.4. After the subject completes a trial by acquiring each of the 5 red cubes, the trial time will be recorded. When all 5 cubes have been acquired, the total experiment time will be calculated as the sum of the 5 trial times.

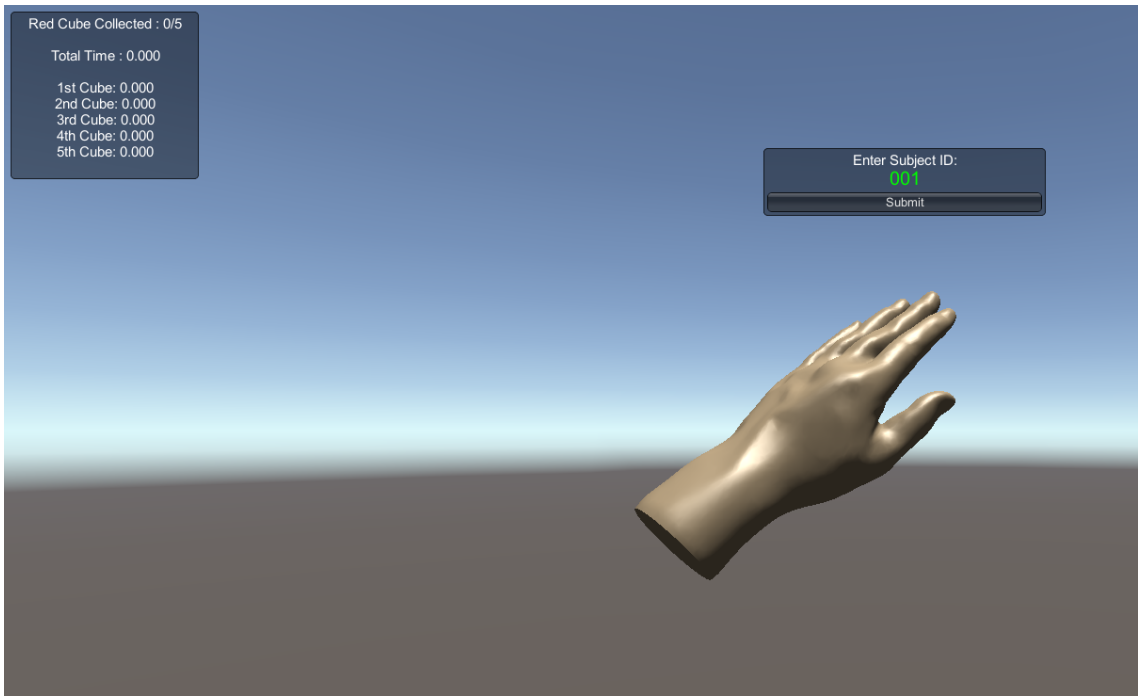


Figure 7.2: Initial stage of the play mode when the subject ID is asked

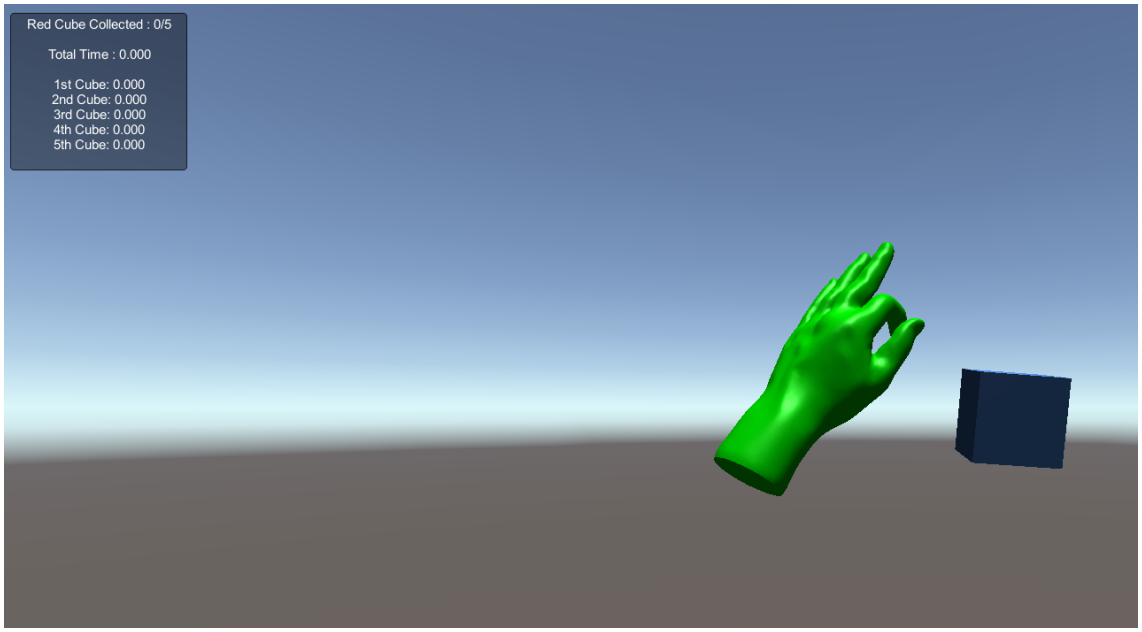


Figure 7.3: The 3D hand model will turn into green indicating the state of flexing

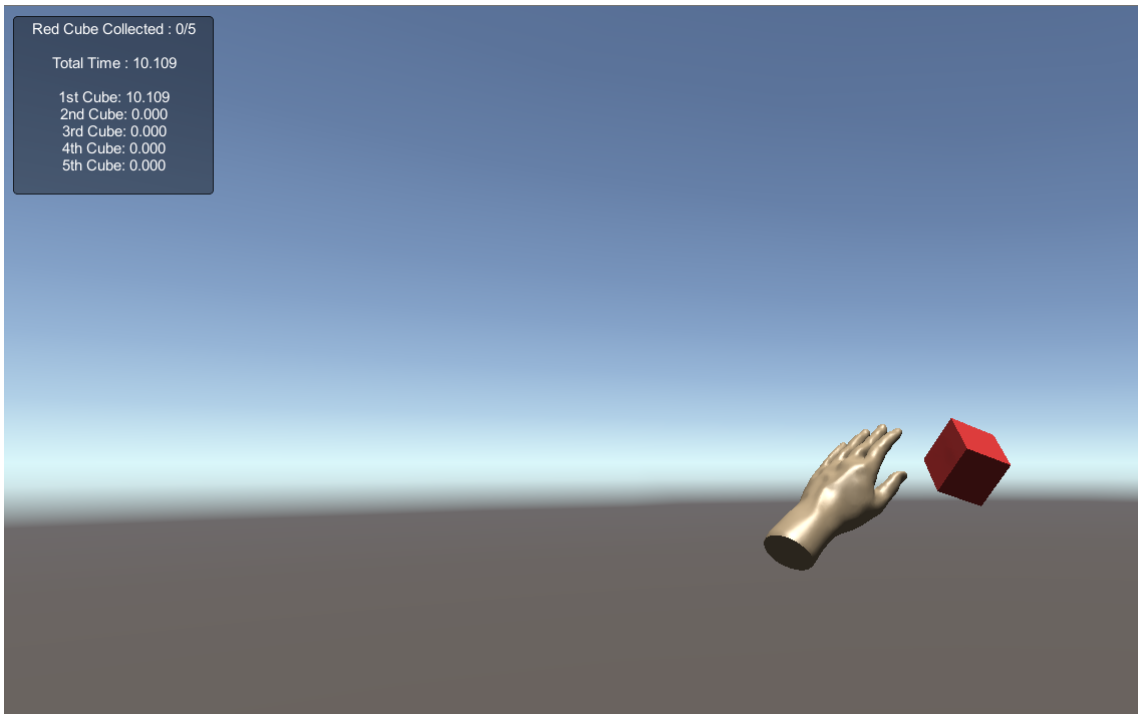


Figure 7.4: The red cube will appear after acquiring the blue cube

7.3 Results

7.3.1 Static Test

To perform the static test, all three inertial measurement units were fixed to a table, to prevent them from moving. The output estimated quaternions for the orientation of hand, index proximal phalange and index middle phalange were recorded for 5 minutes. The output estimated quaternions were recorded when the orientation correction algorithm using gravity vector and magnetic North vector are both disabled and enabled. In Figure 7.5, the output estimated quaternions for hand, index proximal phalange and middle phalange recorded while the orientation correction algorithm was disabled are shown. The result indicates that the bias offset error causes the angular measurement to drift even though the sensors were placed stati-

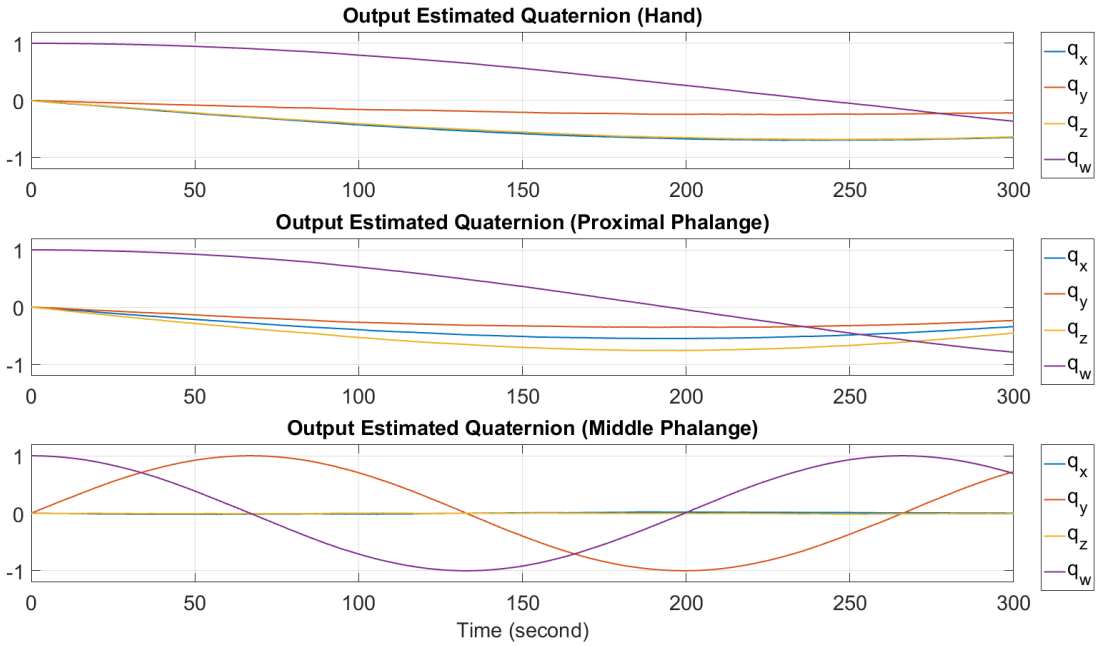


Figure 7.5: Output estimated quaternions without orientation correction algorithm

cally on the table. It was found that the drifts occurred in different rates among the three inertial measurement units. This is because each sensor has a different bias offset error. After enabling the orientation correction algorithm, the plots in Figure 7.6 representing the orientation of 0° angle show that the orientation correction algorithm can improve the estimated quaternion in all three inertial measurement units. This confirmed that in the static mode (when there is no motion applied to the sensors) the orientation correction algorithm using gravity vector and magnetic North vector effectively improve the orientation measurement using the inertial measurement units in real-time.

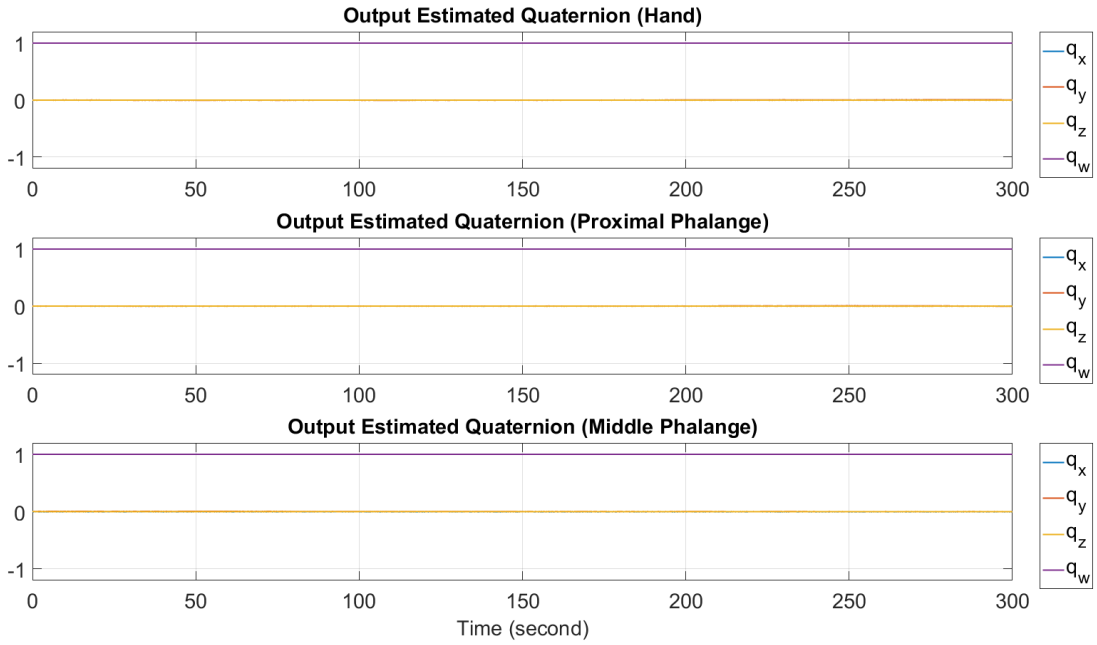


Figure 7.6: Output estimated quaternions with orientation correction algorithm

7.3.2 Results from the dynamic task using the Hand Motion Tracking Interface

To evaluate the performance of the hand motion tracking interface, 30 experiments (each consisting of the acquisition of 5 red cubes) were performed in the 3D environment (described in Section 7.2). During these experiments, the orientation correction algorithm using gravity vector and magnetic North vector was enabled. The time used to acquire each cube was recorded. The statistical characteristics of these acquisition time are shown in Table 7.1.

From the results shown in Table 7.1, it can be seen that the 5 acquisition times in each experiment added to 23.75 seconds on average, with the standard deviation of 6.38 seconds. The minimum total time was 15.27 seconds, whereas the maximum total time was 36.02 seconds. The red cube that took the longest time to acquire

Table 7.1: Statistical data of the time used to acquire red cubes in 3D environment

Statistic Values	Time in seconds					
	1^{st} cube (front)	2^{nd} cube (left)	3^{rd} cube (right)	4^{th} cube (up)	5^{th} cube (down)	Total time
Means	3.34	3.44	5.04	4.60	7.33	23.75
SD	1.04	1.07	1.83	2.01	4.12	6.38
Min	2.05	1.48	2.00	1.38	2.52	15.27
Max	6.45	5.59	8.90	8.77	20.87	36.02

(on average) was the 5^{th} cube, which appeared at the bottom of the screen. It seems, then, that the 5^{th} red cube was more difficult to reach, compared to the other red cubes because its almost out of the OptiTrack V120:Trio field of view. From the experimental data, it can be verified that the real-time implementation of the orientation correction algorithm using gravity vector and magnetic North vector can be effectively applied with the hand motion tracking interface.

As verified by the results, we found that we are able to implement the orientation correction algorithm using gravity vector and magnetic North vector compensation in a real-time manner. Our approach is able to correct the drift in the gyroscope measurements. This method will be one of the effective approaches for the orientation tracking in 3D hand motion tracking interface which can be an alternative way to achieve interactions between a human and a computer. This can also be a significant contribution to improvement in the realism of natural human-computer interactions.

CHAPTER 8

STATISTICAL EVALUATION OF HAND MOTION TRACKING SYSTEM

8.1 Design of Experiment

This chapter describes another experiment that was conducted to comprehensively evaluate the performance and robustness of the hand motion tracking system. Thirty human subjects were asked to participate in the experiment. Each of the subjects wore a glove on his/her left hand and performed hand movement tasks. While the tasks were being performed, the marker coordinate from OptiTrack V120: Trio and the data from an inertial measurement unit attached at the back of the hand, were recorded. The orientation correction algorithm was implemented to calculate the estimated orientation. The orientation estimates obtained with the orientation correction algorithm using the gravity vector and magnetic North vector algorithm are compared with orientation estimates obtained with a fixed bias offset, and the quaternion output from the Kalman-based orientation filtering streamed directly from the Yost Labs 3-Space sensor module.

8.1.1 Testing Environment Setup

For this evaluation, each subject was asked to wear a glove on his/her left hand in which an inertial measurement unit was attached at the back of the hand. The subject was sitting on a chair, having his/her face looking towards a computer screen in which the OptiTrack V120: Trio was installed (sensor bar placed above the screen). Between the subject and the computer screen, a rectangular frame with

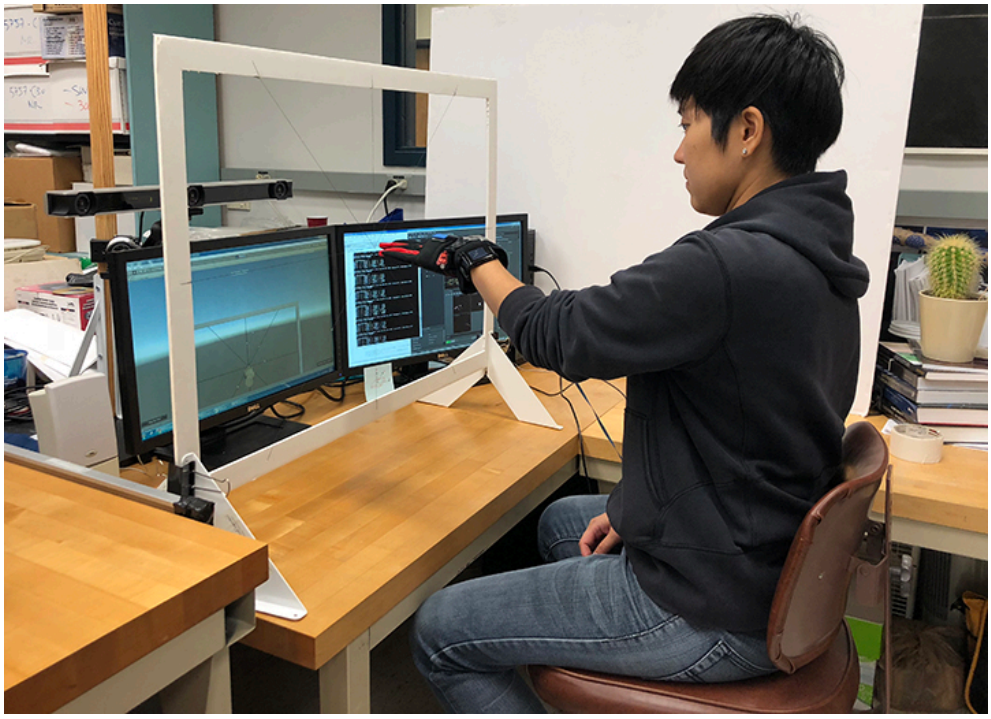


Figure 8.1: Hand motion tracking system testing environment setup

referenced position markers was placed with a fixed distance from the computer screen. The testing environment was set up as shown in Figure 8.1.

8.1.2 Virtual 3D Environment

A virtual 3D environment was created with Unity, for this evaluation. The virtual 3D environment consists of a 3D hand model and the rectangular frame with referenced position markers. The C# script was written and attached to the 3D hand model so that a sequence of 10 pre-defined movements of the 3D hand model were visualized as the hand movement guide for the subjects to perform the tasks. The 10 hand poses involved in the movement sequence for the 3D hand model are depicted as shown in Figure 8.2 and Figure 8.3.

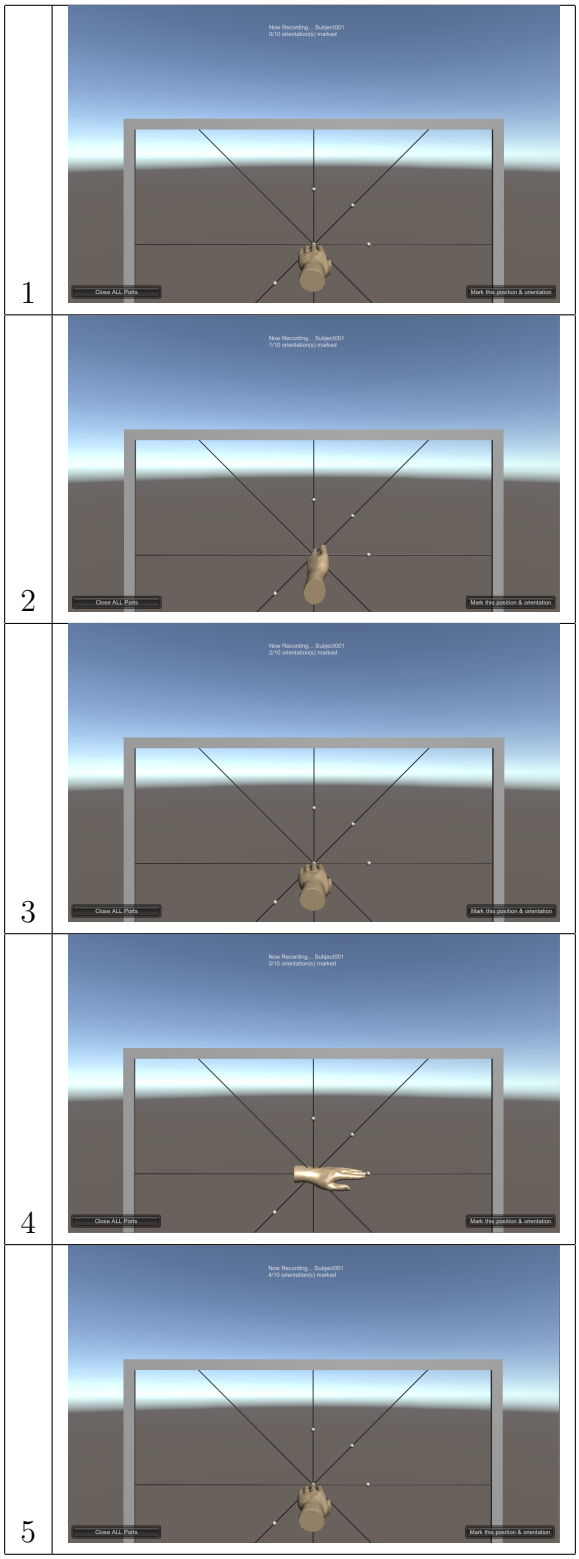


Figure 8.2: The sequence of the 3D hand model movement (poses 1 to 5)

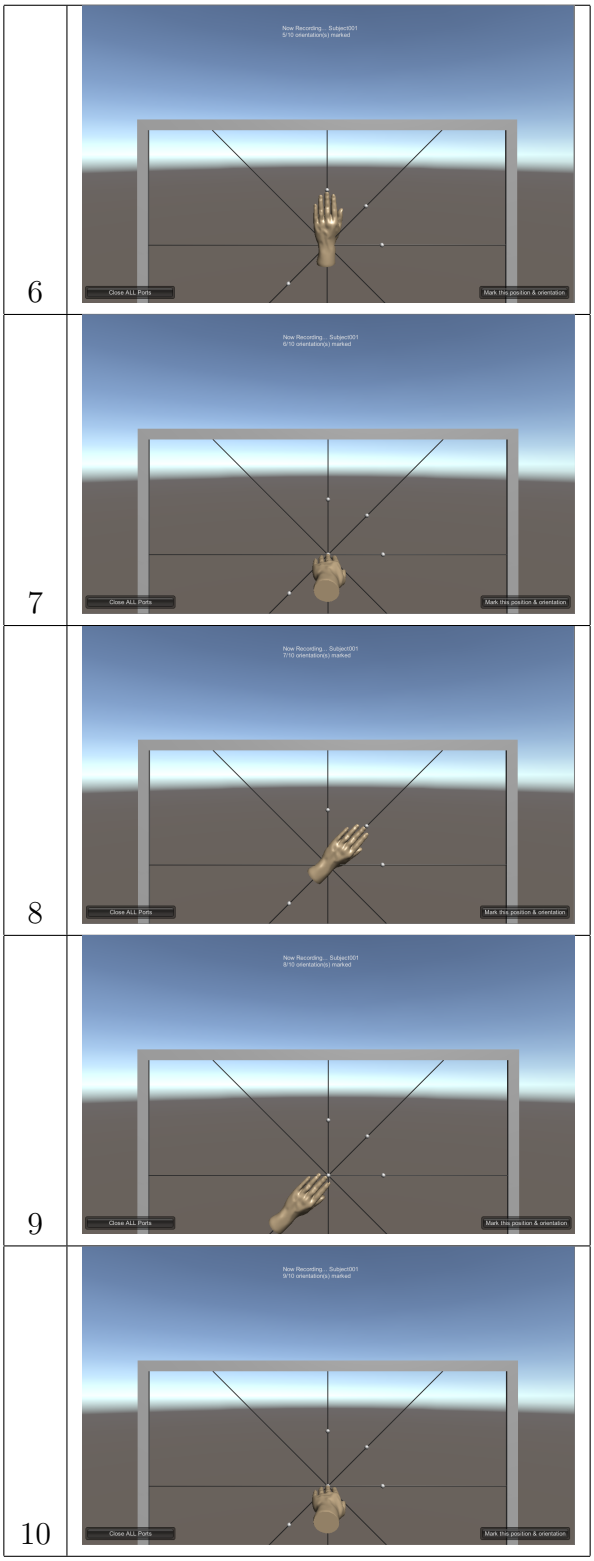


Figure 8.3: The sequence of the 3D hand model movement (poses 6 to 10)

8.2 Experiment Procedure

1. Each subject was asked to wear the glove on his/her left hand and put the hand into the initial position and orientation. (Pose 1 in Figure 8.2)
2. The experimenter clicked on the button labeled as "Mark this position and orientation" to record the initial position and orientation of the hand.
3. The experimenter clicked on the button labeled as "Show hand movement" located on the bottom right corner of the screen, the 3D hand model then rotated and translated to the next state (pose) of the hand movement sequence.
4. The subject moved his/her hand to match the position and orientation as shown by the movement guide on the screen.
5. The experimenter clicked on the button labeled as "Mark this position and orientation" to record the current position and orientation of the subject's hand.
6. Steps 3 to 5 of this procedure were repeated until all 10 states or poses of movement of the 3D hand model had been performed by the subject.
7. The subject was asked to remove the glove and answer the questionnaire about gender, age, and his/her dominant hand.

8.3 Experimental Results

The referenced values of positions and orientations of the 3D hand model have been pre-defined within a C# script written for this evaluation. Once the experimenter clicked on the button labeled as "Mark this position and orientation", instead of printing the position and orientation data to a file, the script automatically calculates the errors of the position and orientation that deviate from referenced data.

Equation 8.1 defines the experimental-quaternion result as the quaternion product of the referenced quaternion and error. By pre-multiplying q^*_{ref} to both sides of the equation as shown in Equation 8.2, we obtain the formula to calculate the error in quaternion form as shown in Equation 8.3. A part of the C# script that performs the position and orientation error calculation is shown in Listing 8.1. The quaternion error for each type of algorithm used to estimate the orientation was calculated by using the formula from Equation 8.3. The errors were converted into Euler angles form using a build-in Unity function. Then, the final results (errors) were stored in a text file.

$$q_{ref} \otimes q_{error} = q_{exp} \tag{8.1}$$

$$q^*_{ref} \otimes q_{ref} \otimes q_{error} = q^*_{ref} \otimes q_{exp} \tag{8.2}$$

$$q_{error} = q^*_{ref} \otimes q_{exp} \tag{8.3}$$

Listing 8.1: Source code to calculate errors in position and orientation

```
// When the button is pressed,
if (GUILayout.Button ("Mark this position & orientation")) {
  // Calculate errors of marker positions
  PosE.x = Mathf.Abs (Pref [rotCount].x - OptitrackRigidBodyManager.
    instance.omPositions [0].x);
  PosE.y = Mathf.Abs (Pref [rotCount].y - OptitrackRigidBodyManager.
    instance.omPositions [0].y);
  PosE.z = Mathf.Abs (Pref [rotCount].z - OptitrackRigidBodyManager.
    instance.omPositions [0].z);
  // Calculate errors in quaternion form
  qGfixed0e = myQuatConj (Qref [rotCount]) * qGfixed0;
```

```
IMUQuat0e = myQuatConj (Qref [rotCount]) * IMUQuat0;
qOUT0e = myQuatConj (Qref [rotCount]) * qOUT0;
// Convert quaternions to Euler angles
EulerFixed0e = qGfixed0e.eulerAngles;
EulerKalman0e = IMUQuat0e.eulerAngles;
EulerGMV0e = qOUT0e.eulerAngles;
//Convert angles exceed 180 degrees to negative angles
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);
// Print the results to file
System.IO.File.AppendAllText(FileStrMark, System.String.Format("
  {0},{1},{2},", Mathf.Abs(EulerFixed0e.x), Mathf.Abs(EulerFixed0e.y)
  , Mathf.Abs(EulerFixed0e.z)));
```

```
38    System.IO.File.AppendAllText(FileStrMark, System.String.Format("
        {0},{1},{2},", Mathf.Abs(EulerKalman0e.x), Mathf.Abs(EulerKalman0e.
        y), Mathf.Abs(EulerKalman0e.z)));
39    System.IO.File.AppendAllText(FileStrMark, System.String.Format("
        {0},{1},{2},", Mathf.Abs(EulerGMV0e.x), Mathf.Abs(EulerGMV0e.y),
        Mathf.Abs(EulerGMV0e.z)));
40    System.IO.File.AppendAllText(FileStrMark, System.String.Format("
        {0},{1},{2}\n", PosE.x, PosE.y, PosE.z));
41  }
```

8.4 Statistical Evaluations

A total of 30 test subjects voluntarily participated in our experiment. There were 10 female participants and 20 male participants with their ages ranging from 19 to 55 years old (26.53 years old on average). All participants were healthy and able to move their hands without any difficulties. Only one participant was left-handed, the remaining 29 participants had the right hands as their dominant hands.

Table 8.1: Descriptive statistics for errors in position tracking

	N	Mean	Std. Deviation
Error in x	300	.01722576	.042859991
Error in y	300	.01031162	.025989994
Error in z	300	.03526925	.095968643

8.4.1 Position Error Analyses

In our experiment, each subject moved his/her hand and placed it at 10 prescribed states or poses. The positions of the subjects' hands in three-dimensional space

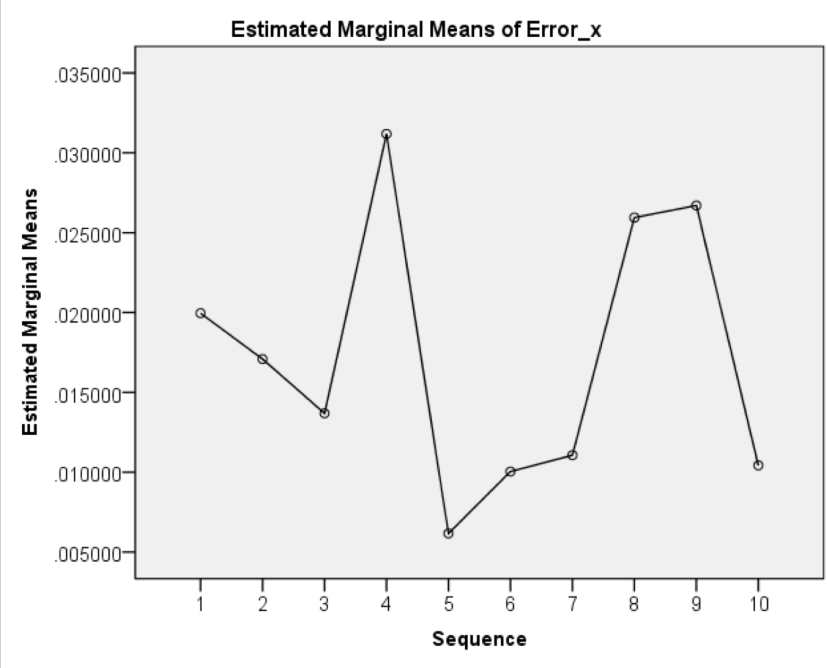


Figure 8.4: Estimated marginal means of the position errors in x

were recorded. A total of 300 rows of data (30 test subjects $\times$ 10 states of hand movement) were statistically analyzed using SPSS. The descriptive statistics for the errors in position tracking are shown in Table 8.1. The position error in the x-axis is 1.7 cm on average, the mean of position error in the y-axis is 1.0 cm, and 3.5 cm for the z-axis. The estimated marginal means of the position errors in x, y and z are shown in Figures 8.4 to 8.6. Notice that the means of position errors for the 4^{th}, 8^{th} and 9^{th} states or poses in all axes are relatively higher than other states in the movement sequence. This could be because of the less natural hand movements required for those states, which caused difficulties for the test subjects in trying to exactly match the pre-defined positions of the hand that were requested.

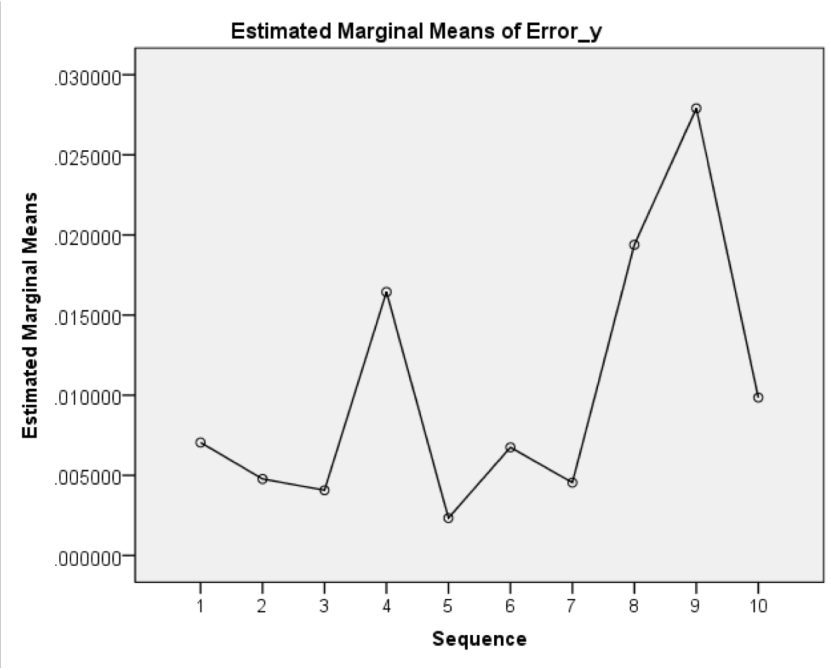


Figure 8.5: Estimated marginal means of the position errors in y

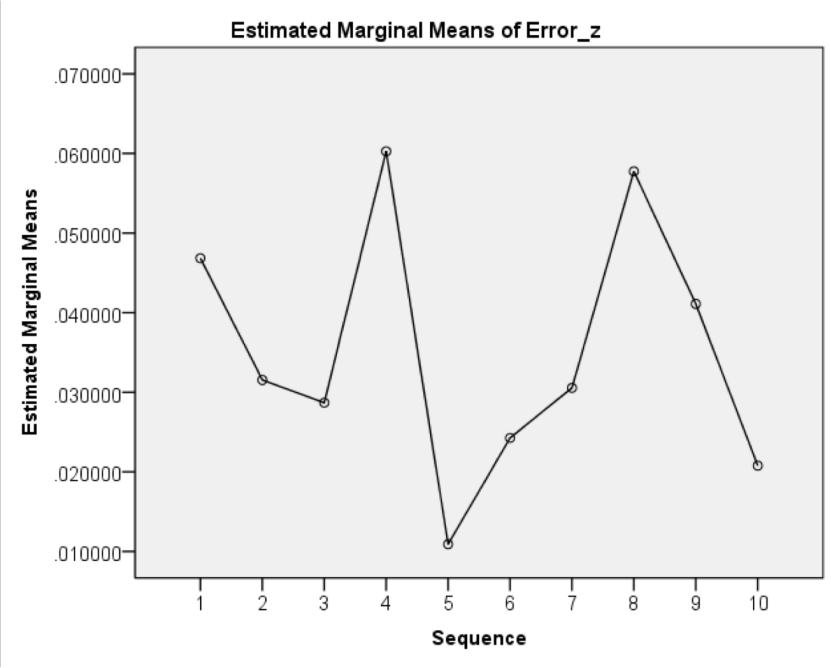


Figure 8.6: Estimated marginal means of the position errors in z

8.4.2 Orientation Error Analyses

In our experiment, each subject moved his/her hand through a sequence of 10 hand poses, or states, while three different orientation correction algorithms were running simultaneously to estimate the orientations. The purpose of this experiment is to evaluate the effects of three different orientation correction algorithms on the orientation output errors. The orientation errors were calculated under the assumption that the subjects oriented their hands exactly as required in each pose, aided by the frame provided. The orientation output errors in the form of Euler Angles (Phi, Theta and Psi) were calculated for the three orientation correction algorithms which are: 1.) the orientation correction using fixed bias offset (FB), 2.) the correction using the Kalman-based orientation filtering streamed directly from the Yost Labs 3-Space sensor module (KF), and 3.) the proposed orientation correction using gravity and magnetic North vector (GMV). A total of 900 rows of data (30 test subjects $\times$ 10 states of hand movement $\times$ 3 algorithms) were recorded and statistically analyzed using SPSS. Table 8.2 shows the estimated means of the orientation errors in all Euler angles. Notice that the means of the orientation errors for GMV and KF are less than that of FB in every Euler angle. The means of the orientation errors for GMV are similar to the means of the orientation errors for KF in every Euler angle. The estimated marginal means of the orientation errors for Phi, Theta and Psi are shown in Figure 8.7 to 8.9, respectively.

To test for the effects of three algorithms on the orientation output errors, a multivariate analysis of variance (MANOVA) was initially suggested. Before performing an analysis of variance, the data has to be validated on two assumptions, which are normality of the error and equal variances across treatments. The normality test is the test for the null hypothesis that the data are normally distributed within each treatment group. Table 8.3 shows the results of the test statistics Kolmogorov-

Table 8.2: Estimated means of the orientation output errors

Dependent Variable	Algorithm	Mean
Phi	FB	11.505
	GMV	5.24
	KF	5.171
Theta	FB	9.288
	GMV	4.646
	KF	4.258
Psi	FB	11.137
	GMV	4.854
	KF	4.799

Table 8.3: Tests of normality

		Kolmogorov-Smirnov			Shapiro-Wilk		
	Algorithm	Statistic	df	Sig.	Statistic	df	Sig.
Phi	FB	.156	300	.000	.836	300	.000
	GMV	.151	300	.000	.876	300	.000
	KF	.182	300	.000	.862	300	.000
Theta	FB	.114	300	.000	.923	300	.000
	GMV	.159	300	.000	.842	300	.000
	KF	.180	300	.000	.817	300	.000
Psi	FB	.139	300	.000	.889	300	.000
	GMV	.182	300	.000	.814	300	.000
	KF	.185	300	.000	.820	300	.000

Table 8.4: Levene's test of equality of error variances

	F	df1	df2	Sig.
Phi	29.352	29	870	.000
Theta	21.847	29	870	.000
Psi	26.455	29	870	.000

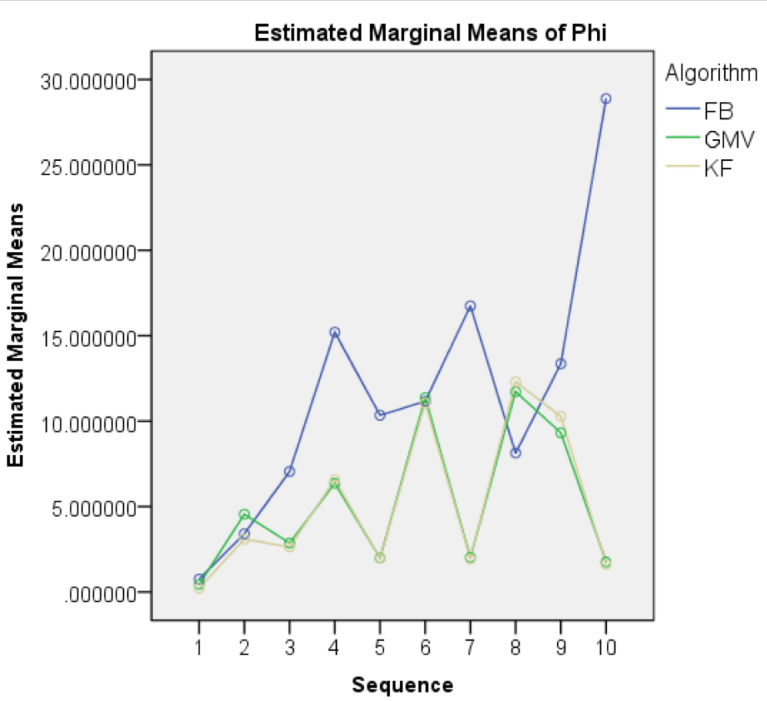


Figure 8.7: Estimated marginal means of the orientation errors for Phi

Smirnov and Shapiro-Wilk, having the p-values for testing normality of 0.000 (the null hypothesis is rejected), indicating strong evidence that the orientation output errors are not normally distributed. Table 8.4 shows the results of the test statistics for the null hypothesis that the error variance of the dependent variable is equal across treatment groups. The p-values of the test of homogeneity of variances are 0.000 (the null hypothesis is rejected) for all three dependent variables (Phi, Theta and Psi), indicating strong evidence that the error variances are not equal among three treatment groups (three algorithms). Since the normality and the homogeneity of variance assumptions are not met, the multivariate analysis of variance (MANOVA) cannot be used as a statistical test model on this data. In this situation, the Kruskal-Wallis H test is suggested as a nonparametric alternative to the usual analysis of variance [53]. The Kruskal-Wallis test is a rank-based nonparametric

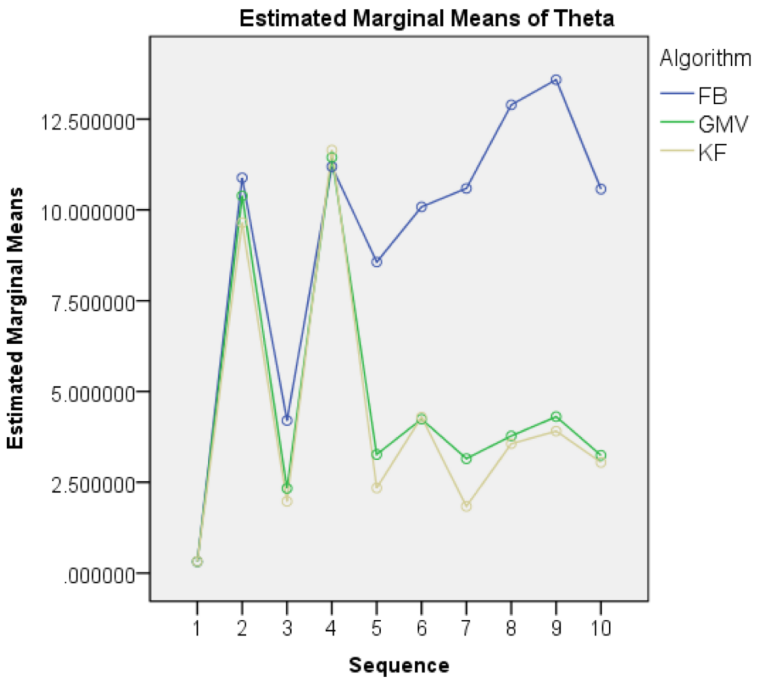


Figure 8.8: Estimated marginal means of the orientation errors for Theta

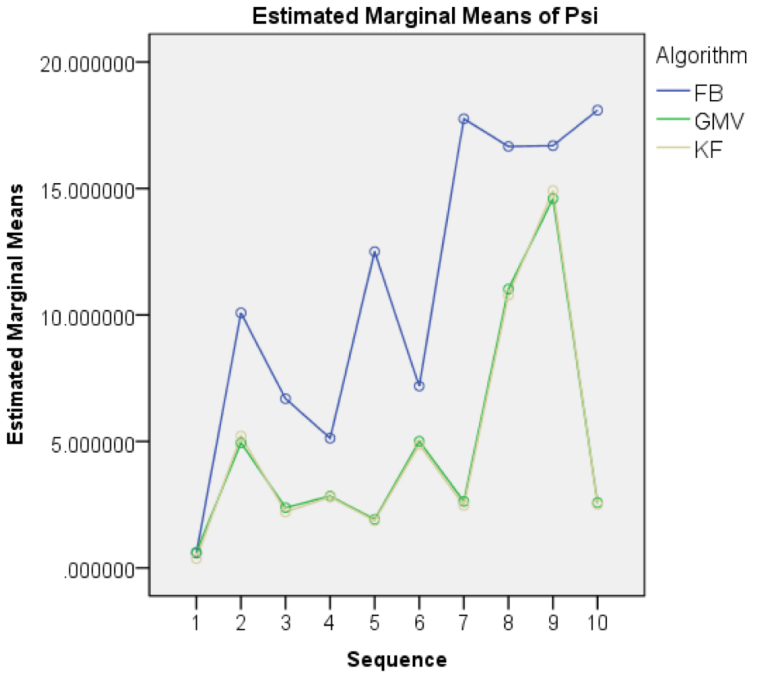


Figure 8.9: Estimated marginal means of the orientation errors for Psi

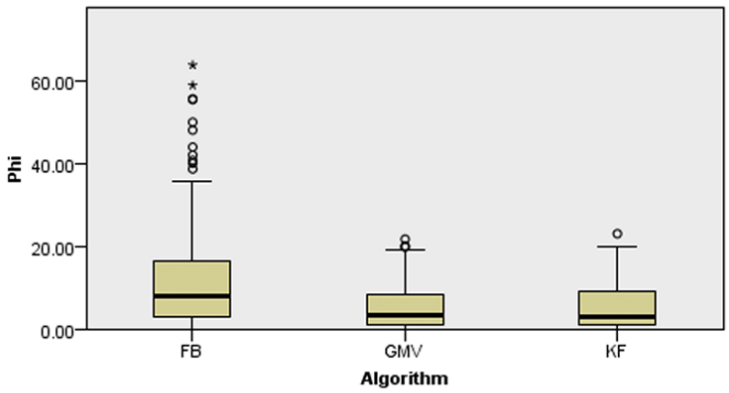
test, which is used to determine the statistical significance of the differences of a dependent variable across two or more treatment groups [54]. Each dependent variable (Phi, Theta and Psi) was tested with Kruskal-Wallis on a 0.05 level of significance to determine if there is a difference in means across the three algorithms. Figure 8.10 shows the test statistics for the orientation output errors in the Euler angle Phi, across three algorithms, in which the null hypothesis is that the distribution of orientation errors in Phi is the same across the three algorithms. The result indicates that there is a statistically significant difference between the orientation errors in Phi produced by different algorithms ($H(2) = 80.773, p = 0.000$), with a mean rank of 560.48 for FB, 400.60 for GMV and 390.43 for KF. For a pairwise comparisons among three algorithms, the results indicate that there are statistically significant differences of the orientation errors in Phi between KF and FB ($p = 0.000$) and between GMV and FB ($p = 0.000$). There is no statistically significant difference of the orientation errors in Phi between KF and GMV ($p = 0.632$). Figure 8.11 shows the test statistics for the orientation output errors in the Euler angle Theta, across the three algorithms, in which the null hypothesis is that the distribution of orientation errors in Theta is the same across the three algorithms. The result indicates that there is a statistically significant difference between the orientation errors in Theta produced by different algorithms ($H(2) = 89.439, p = 0.000$), with a mean rank of 565.57 for FB, 404.86 for GMV and 381.06 for KF. For a pairwise comparisons among three algorithms, the results indicate that there are statistically significant differences of the orientation output errors in Theta between KF and FB ($p = 0.000$) and between GMV and FB ($p = 0.000$). There is no statistically significant difference of the orientation errors in Theta between KF and GMV ($p = 0.262$). Figure 8.12 shows the test statistics for the orientation output errors in the Euler angle Psi, across the three algorithms, in which the null hypothesis is that the distribution of

orientation errors in Psi is the same across the three algorithms. The result indicates that there is a statistically significant difference between the orientation errors in Psi produced by different algorithms ($H(2) = 89.528, p = 0.000$), with a mean rank of 566.37 for FB, 396.26 for GMV and 388.87 for KF. For a pairwise comparisons among three algorithms, the results indicate that there are statistically significant differences of the orientation errors in Psi between KF and FB ($p = 0.000$) and between GMV and FB ($p = 0.000$). There is no statistically significant difference of the orientation errors in Psi between KF and GMV ($p = 0.728$).

Figures 8.7, 8.8 and 8.9 show the estimated marginal means of the orientation errors found using the 3 correction methods. The errors in Euler angles for the orientation correction algorithm using the gravity vector and magnetic North vector (GMV) are similar to the errors in Euler angles for the on-board Kalman-based orientation filtering (KF), for every hand movement in the sequence. The large amount of orientation errors from both GMV and KF in some poses could be caused by the difficulty experienced by the subjects in trying to exactly match the pre-defined orientations of the hand requested on the computer screen. It is possible that the amount of orientation errors for both algorithms could in fact be smaller, and both algorithms could estimate the actual orientation of the hand.

The statistical analyses show that the orientation correction algorithm using gravity vector and magnetic North vector can significantly reduce the errors in orientation tracking when comparing to the orientation correction algorithm using fixed bias offset. The orientation correction algorithm using gravity vector and magnetic North vector is able to estimate the orientation with no significant difference from the Kalman-based orientation filtering.

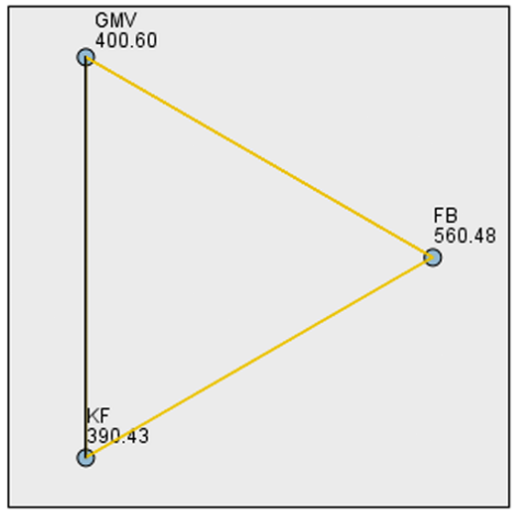
Kruskal-Wallis Test for orientation errors: Phi



Total N	900
Test Statistic	80.773
Degrees of Freedom	2
Asymptotic Sig. (2-sided test)	.000

1. The test statistic is adjusted for ties.

Pairwise Comparisons of Algorithm: Phi

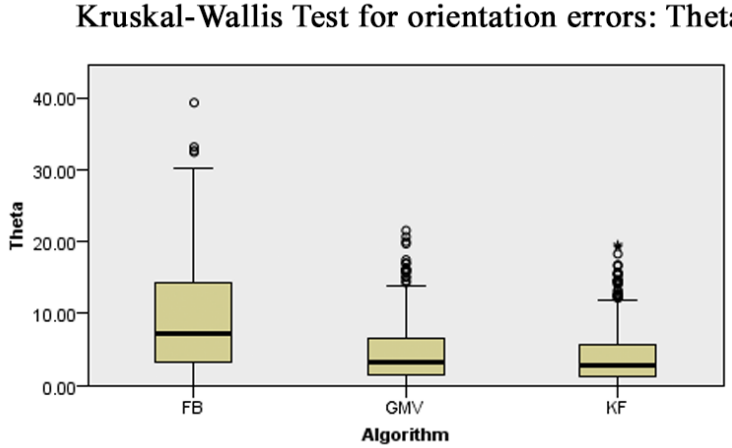


Each node shows the sample average rank of Algorithm.

Sample1-Sample2	Test Statistic	Std. Error	Std. Test Statistic	Sig.	Adj.Sig.
KF-GMV	10.170	21.225	.479	.632	1.000
KF-FB	170.050	21.225	8.012	.000	.000
GMV-FB	159.880	21.225	7.533	.000	.000

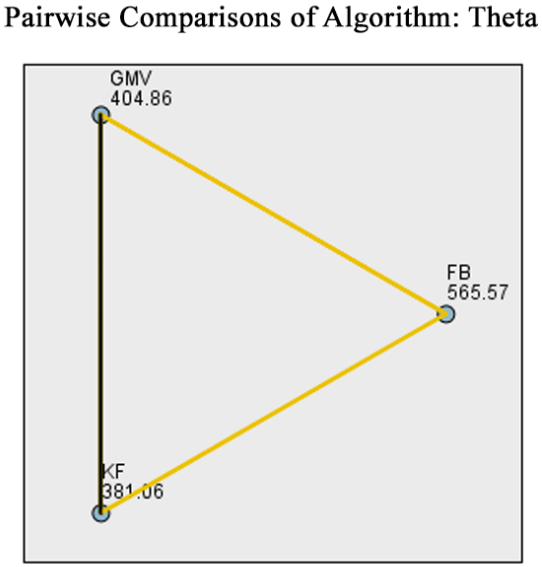
Each row tests the null hypothesis that the Sample 1 and Sample 2 distributions are the same.
Asymptotic significances (2-sided tests) are displayed. The significance level is .05.

Figure 8.10: Results of Kruskal-Wallis test statistics for the orientation errors in the Euler angle Phi, across three different algorithms. (In the box plot, circles are outliers and asterisks are extreme outliers.)



Total N	900
Test Statistic	89.439
Degrees of Freedom	2
Asymptotic Sig. (2-sided test)	.000

1. The test statistic is adjusted for ties.

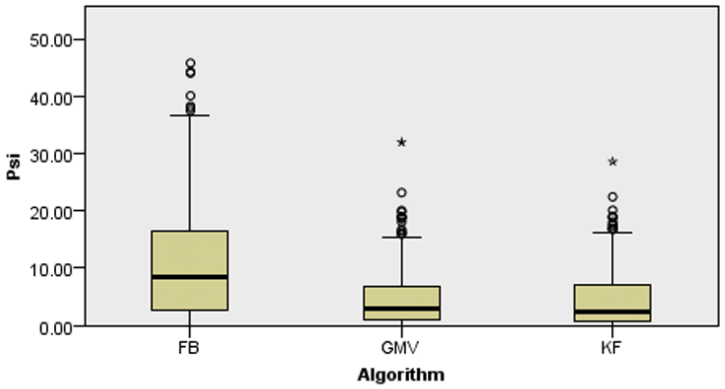


Each node shows the sample average rank of Algorithm.

Sample1-Sample2	Test Statistic	Std. Error	Std. Test Statistic	Sig.	Adj.Sig.
KF-GMV	23.800	21.225	1.121	.262	.786
KF-FB	184.510	21.225	8.693	.000	.000
GMV-FB	160.710	21.225	7.572	.000	.000

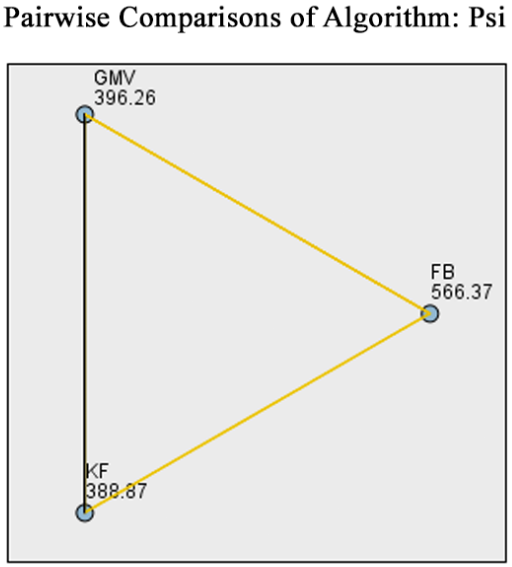
Each row tests the null hypothesis that the Sample 1 and Sample 2 distributions are the same.
Asymptotic significances (2-sided tests) are displayed. The significance level is .05.

Figure 8.11: Results of Kruskal-Wallis test statistics for the orientation errors in the Euler angle Theta, across three different algorithms. (In the box plot, circles are outliers and asterisks are extreme outliers.)



Total N	900
Test Statistic	89.528
Degrees of Freedom	2
Asymptotic Sig. (2-sided test)	.000

1. The test statistic is adjusted for ties.



Each node shows the sample average rank of Algorithm.

Sample1-Sample2	Test Statistic	Std. Error	Std. Test Statistic	Sig.	Adj.Sig.
KF-GMV	7.390	21.225	.348	.728	1.000
KF-FB	177.500	21.225	8.363	.000	.000
GMV-FB	170.110	21.225	8.015	.000	.000

Each row tests the null hypothesis that the Sample 1 and Sample 2 distributions are the same.
Asymptotic significances (2-sided tests) are displayed. The significance level is .05.

Figure 8.12: Results of Kruskal-Wallis test statistics for the orientation errors in the Euler angle Psi, across three different algorithms. (In the box plot, circles are outliers and asterisks are extreme outliers.)

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

This dissertation presented a novel approach in building a system which is capable of determining the position and orientation of the human hand in three-dimensional space, and track the movement of the hand in real-time using MEMS inertial measurement units and infrared cameras. Our research focused on the study and implementation of an algorithm to correct the gyroscope drift, which is a major problem found when using commercial-grade MEMS inertial measurement units for several applications. Gyroscope drift is caused by an artifactual non-zero angular velocity measurement generated while the sensor is static (not in motion). This erroneous gyroscope output produced while there is no input force applied to the inertial measurement unit is called bias offset error. To determine the orientation, the angular velocity is mathematically integrated. Therefore, the bias offset error in the angular velocity reading can be accumulated and produce large amounts of orientation error that grows proportionally to time.

A novel algorithm to improve the IMU orientation estimation was proposed and evaluated. The algorithm begins with the prediction of the bias offset error by using a simple linear regression model to calculate the bias offset error every time when the sensor module is in a static period. When the IMU is not in motion, the gyroscope is supposed to ideally produce the reading of zero. The actual measurement at that time can be used to predict the bias offset error. Results obtained by compensation using the predicted bias offset error show that the algorithm can be used to predict the value while the sensor is not moving and continue to use the same bias offset error while the sensor is not stationary. The unbiased angular velocity was then

calculated by subtracting the gyroscope measurement with the predicted bias offset error previously calculated. This unbiased angular velocity was used to determine the orientation in a quaternion form. This result of the computed quaternion can approximately represent the orientation of the sensor module. However, by only removing the bias offset error, there was still some drift presented in the orientation result.

The computed quaternion was then corrected utilizing the gravity vector and the magnetic North vector. For these correction, the computed quaternion was used to calculate the gravity vector and the magnetic North vector referenced in the sensor's body frame. The calculated gravity vector and calculated magnetic North vector were compared with the measured gravity vector and measured magnetic North vector obtained from accelerometer and magnetometer, respectively. The difference between calculated and measured gravity vectors, and the difference between calculated and measured magnetic North vectors were determined. These differences can be expressed in quaternion form and can be used to correct the original quaternion result to obtain corrected quaternion estimates of the sensor's orientation based on gravity vector and the magnetic North vector corrections, respectively. Then, quaternion interpolation was applied in order to determine the final quaternion estimation by having the control parameter (α) to interpolate between corrected quaternion based on the gravity vector ($\alpha = 1$) and corrected quaternion based on the magnetic North vector ($\alpha = 0$). The control parameter (α) ranging in value from 0 to 1, was derived from the sensor's stillness parameter. The higher value of α (closer to 1) indicates that the sensor is in static period and the final quaternion estimation tends to follow the corrected quaternion based on the gravity vector rather than the corrected quaternion based on the magnetic North vector.

The orientation correction algorithm was preliminarily implemented offline on previously recorded data in order to validate its robustness before applying it to real-time performance. The IMU data was recorded while a subject wearing an instrumented glove performed pre-planned hand movements. By comparing between a sequence of actual hand orientations and 3D visualizations of estimated hand orientations corrected using gravity vector and magnetic North vector correction, confirmed that the proposed correction method provides acceptable results for hand orientation tracking using the IMUs.

The orientation correction algorithm was then adapted and implemented in real-time using Unity. In this implementation, the hand position was estimated from the position of a marker in the wrist of the glove detected by the OptiTrack V120: Trio system, while the orientation was estimated from IMU signals. The real-time results verified that the implementation of the algorithm was able to correct the drift in the gyroscope measurements for the static test and can be effectively applied for hand motion tracking in a real-time manner.

The proposed 3D hand motion tracking system has been verified to be capable of combining two different sources of information (orientation from the IMU and position from the infrared cameras) in order to be aware of the human hand motion in real-time. The validation of the 3D hand motion tracking system was performed by a human subject study in which 30 volunteers completed a hand-movement task. In the experiment, the participants wore the glove with the IMU attached on the back of the hand and performed hand movements according to a hand movement guide shown on the computer screen. For each participant, 10 sets of the orientation and position data corresponding to the 10 pre-defined states of movement (poses) of the 3D hand model were recorded. The statistical analysis shows that the error of position tracking is 1.7 cm on average in the x-axis, 1.0 cm on average in the

y-axis, and 3.5 cm on average in the z-axis. The Kruskal-Wallis tests show that the orientation correction algorithm using gravity vector and magnetic North vector can significantly reduce the errors in orientation tracking when comparing to the orientation correction algorithm using fixed bias offset. The pairwise comparison between the orientation correction algorithm using gravity vector and magnetic North vector and the on-board Kalman-based orientation filtering indicates that there is no statistically significant difference of the orientation output errors for Phi, Theta and Psi with p-values of 0.632, 0.262 and 0.728, respectively. (i.e. The orientation correction algorithm using gravity vector and magnetic North vector is able to estimate the orientation with no significant difference from the Kalman-based orientation filtering.) However, the orientation correction algorithm using gravity vector and magnetic North vector provides the flexibility in selecting the source of reference for the correction (i.e., accelerometer or magnetometer), depending on each different circumstance of measurement. Moreover, unlike Kalman-based orientation filtering, the orientation correction algorithm using gravity vector and magnetic North vector does not require the initializations of noise covariance and uncertainty matrices.

Results from the proposed orientation correction algorithm suggest that this may be one of the effective approaches that will enable orientation tracking in 3D hand motion tracking interfaces, which may provide alternative ways for a human to interact with a computer. The development of hand motion tracking systems using inertial measurement units and infrared cameras can be a significant contribution to the improvement in the realism of natural human-computer interactions within a 3D virtual environment.

It should be noted that one key advantage of the orientation correction approach proposed in this dissertation over other standard approaches (e.g., Kalman Filtering) is that it handles both sources of information used for correction: ac-

celeration measurements and magnetometer measurements on parallel tracks that remain completely distinguishable and modifiable throughout the complete execution of the correction algorithm. This opens up the possibility of dynamically (in real-time) changing the parameters that define the use of each of these sources of information and the weight that each of them is given in defining the final correction of the orientation estimate quaternion in each iteration. In the future, this property may pave the way to the development of orientation correction algorithms that change their operating parameters during use, to improve their performance after they are used in a specific location.

9.2 Future Work

In this work, the α as the parameter used to determine the final quaternion output was calculated from the Yost Labs 3-Space sensor's internal parameter called "confidence value" which indicates the sensor's stillness. In the future, it is expected that our orientation correction algorithm could be improved to be generalized that all control parameters will be determined from the inertial measurements (accelerations, angular velocity) themselves. When the IMU is in motion and the accelerometer measures not only the acceleration due to gravity but also the acceleration due to linear motion, the measurements from accelerometer can no longer be used as reliable references to correct the orientation.

In those cases, a low value of α interpolates the final quaternion to rely more on the quaternion correction using the magnetic North vector because it is not affected by the motion of the IMU. However, in some circumstances, the magnetic field around the user might not be uniform, due to the presence of ferromagnetic objects. This may degrade the overall orientation estimation when the hand is

located in certain space regions. This situation could be improved if the directions of the magnetic field in any particular positions around the user are known. The continued study of the orientation correction algorithm could focus on the modeling of the magnetic field around the user and the additional orientation correction when the IMU is in motion at any different positions in 3D space due to non-uniform magnetic field.

The full hand motion tracking system could further be improved by adding more inertial measurement units at the remaining fingers and thumb so that the system can be utilized for more natural 3D user interface in virtual environment such as grabbing, holding objects or performing more diverse hand gestures.

BIBLIOGRAPHY

[1] X. Zhang, X. Liu, S.-M. Yuan, and S.-F. Lin, "Eye tracking based control system for natural human-computer interaction," *Computational Intelligence and Neuroscience*, vol. 2017, 2017.

[2] M.-C. Roh, D. Kang, S. Huh, and S.-W. Lee, "A virtual mouse interface with a two-layered bayesian network," *Multimedia Tools and Applications*, vol. 76, no. 2, pp. 1615–1638, 2017. ID: Roh2017.

[3] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, 1997.

[4] R. Mccall, S. O'Neil, and F. Carroll, *Measuring presence in virtual environments*. ACM, 2004. ID: acm985934; ACM Digital Library; ACM Digital Library (Association for Computing Machinery); KESLI (ACM Digital Library).

[5] M. Slater, M. Usoh, and A. Steed, "Depth of presence in virtual environments," *Presence: Teleoperators & Virtual Environments*, vol. 3, no. 2, pp. 130–144, 1994.

[6] M. Slater and S. Wilbur, "A framework for immersive virtual environments (five)," *Presence: Teleoperators & Virtual Environments*, vol. 6, no. 6, p. 603, 1997.

[7] C. Heeter, "Being there: The subjective experience of presence," *Presence: Teleoperators and Virtual Environments*, vol. 1, no. 2, pp. 262–271, 1992. doi: 10.1162/pres.1992.1.2.262; 07.

[8] S. Sukkarieh and E. M. Nebot, "A high integrity imu/gps navigation loop for autonomous land vehicle applications," *IEEE Transactions on Robotics & Automation*, vol. 15, no. 3, p. 572, 1999.

[9] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, "Mobile robot positioning sensors and techniques," *NAVAL COMMAND CONTROL AND OCEAN SURVEILLANCE CENTER RDT AND E DIV SAN DIEGO CA*, 1997.

[10] J. L. Marins, X. Yun, E. R. Bachmann, R. B. McGhee, and M. J. Zyda, "An extended kalman filter for quaternion-based orientation estimation using marg sensors," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4, pp. 2003–2011, IEEE, 2001.

[11] X. Yun and E. R. Bachmann, "Design, implementation, and experimental results of a quaternion-based kalman filter for human body motion tracking," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1216–1227, 2006.

[12] X. Yun, M. Lizarraga, E. R. Bachmann, and R. B. McGhee, "An improved quaternion-based kalman filter for real-time tracking of rigid body orientation," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, pp. 1074–1079, IEEE, 2003.

[13] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pp. 1–7, IEEE, 2011.

[14] S. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," *Report x-io and University of Bristol (UK)*, vol. 25, 2010.

[15] E. R. Bachmann, I. Duman, U. Y. Usta, R. B. McGhee, X. P. Yun, and M. J. Zyda, "Orientation tracking for humans and robots using inertial sensors," in *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on*, pp. 187–194, IEEE, 1999.

[16] X. Kong, "Ins algorithm using quaternion model for low cost imu," *Robotics and Autonomous Systems*, vol. 46, no. 4, pp. 221–246, 2004.

[17] J. L. Hernandez-Rebollar, R. W. Lindeman, and N. Kyriakopoulos, "A multi-class pattern recognition system for practical finger spelling translation," in *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, pp. 185–190, IEEE, 2002.

[18] T. D. Bui and L. T. Nguyen, "Recognizing postures in vietnamese sign language with mems accelerometers," *IEEE sensors journal*, vol. 7, no. 5, pp. 707–712, 2007.

[19] J.-H. Kim, N. D. Thang, and T.-S. Kim, "3-d hand motion tracking and gesture recognition using a data glove," in *Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on*, pp. 1013–1018, IEEE, 2009.

[20] T. S. Saponas, D. S. Tan, D. Morris, R. Balakrishnan, J. Turner, and J. A. Landay, "Enabling always-available input with muscle-computer interfaces," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pp. 167–176, ACM, 2009.

[21] X. Zhang, X. Chen, W.-h. Wang, J.-h. Yang, V. Lantz, and K.-q. Wang, "Hand gesture recognition and virtual game control based on 3d accelerometer and emg sensors," in *Proceedings of the 14th international conference on Intelligent user interfaces*, pp. 401–406, ACM, 2009.

[22] K.-Y. Chen, K. Lyons, S. White, and S. Patel, "utrack: 3d input using two magnetic sensors," in *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pp. 237–244, ACM, 2013.

[23] C.-S. Fahn and H. Sun, "Development of a fingertip glove equipped with magnetic tracking sensors," *Sensors*, vol. 10, no. 2, pp. 1119–1140, 2010.

[24] K. N. Tarchanidis and J. N. Lygouras, "Data glove with a force sensor," *IEEE Transactions on Instrumentation and measurement*, vol. 52, no. 3, pp. 984–989, 2003.

[25] G. Saggio, F. Giannini, M. Todisco, and G. Costantini, "A data glove based sensor interface to expressively control musical processes," in *Advances in Sensors and Interfaces (IWASI), 2011 4th IEEE International Workshop on*, pp. 192–195, IEEE, 2011.

[26] A. Tognetti, N. Carbonaro, G. Zupone, and D. De Rossi, "Characterization of a novel data glove based on textile integrated sensors," in *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pp. 2510–2513, IEEE, 2006.

[27] K. Kurita, "Non-contact and non-attached human hand motion sensing technique for application to the human machine interface," in *SICE Annual Conference 2010, Proceedings of*, pp. 3536–3539, IEEE, 2010.

[28] C. Nölker and H. Ritter, "Detection of fingertips in human hand movement sequences," in *International Gesture Workshop*, pp. 209–218, Springer, 1997.

[29] O. Rumyantsev, M. Merati, and V. Ramachandran, "Hand sign recognition through palm gesture and movement," *Image Processing*, 2012.

[30] E.-J. Ong and R. Bowden, "A boosted classifier tree for hand shape detection," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pp. 889–894, IEEE, 2004.

[31] M. Alsheakhali, A. Skaik, M. Aldahdouh, and M. Alhelou, "Hand gesture recognition system," *Information & Communication Systems*, vol. 132, 2011.

[32] S. Park, S. Yu, J. Kim, S. Kim, and S. Lee, "3d hand tracking using kalman filter in depth space," *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, p. 36, 2012.

[33] M. Elgendi, F. Picon, and N. Magenant-Thalmann, "Real-time speed detection of hand gesture using, kinect," in *Proc. Workshop on Autonomous Social Robots and Virtual Humans, The 25th Annual Conference on Computer Animation and Social Agents (CASA 2012)*, 2012.

[34] D. Titterton, J. L. Weston, and J. Weston, *Strapdown inertial navigation technology*, vol. 17. IET, 2004.

[35] J. B. Kuipers, *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton, N.J: Princeton University Press, 1999. ID: 024971770; Includes bibliographical references (p. 365-366) and index.

[36] J.-L. Blanco, "A tutorial on se (3) transformation parameterizations and on-manifold optimization," *University of Malaga, Tech. Rep*, vol. 3, 2010.

[37] P. Aggarwal, *MEMS-based integrated navigation*. Artech House, 2010.

[38] L. Lin and A. P. Pisano, "Silicon-processed microneedles," *Journal of Microelectromechanical Systems*, vol. 8, no. 1, pp. 78–84, 1999.

[39] M. Gad-el Hak, *The MEMS handbook*. CRC press, 2001.

[40] H. Hou, *Modeling inertial sensors errors using Allan variance*. University of Calgary, Department of Geomatics Engineering, 2004.

[41] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using allan variance," *IEEE Transactions on instrumentation and measurement*, vol. 57, no. 1, pp. 140–149, 2008.

[42] P. Aggarwal, Z. Syed, X. Niu, and N. El-Sheimy, "A standard testing and calibration procedure for low cost mems inertial sensors and units," *The Journal of Navigation*, vol. 61, no. 2, pp. 323–336, 2008.

[43] N. El-Sheimy, "Inertial techniques and ins/dgps integration," *Engo 623-Course Notes*, pp. 170–182, 2006.

[44] R. G. Brown, P. Y. Hwang, *et al.*, *Introduction to random signals and applied Kalman filtering*, vol. 3. Wiley New York, 1992.

[45] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*, vol. 821. John Wiley & Sons, 2012.

[46] R. G. Valenti, I. Dryanovski, and J. Xiao, "Keeping a good attitude: A quaternion-based orientation filter for imus and margs," *Sensors*, vol. 15, no. 8, pp. 19302–19330, 2015.

[47] N. O-larnnithipong and A. Barreto, "Gyroscope drift correction algorithm for inertial measurement unit used in hand motion tracking," in *2016 IEEE SENSORS*, pp. 1–3, 2016.

[48] E. B. Dam, M. Koch, and M. Lillholm, *Quaternions, interpolation and animation*, vol. 2. Datalogisk Institut, Kbenhavns Universitet, 1998.

[49] J. C. Principe, J.-M. Kuo, and S. Celebi, "An analysis of the gamma memory in dynamic neural networks," *IEEE transactions on Neural Networks*, vol. 5, no. 2, pp. 331–337, 1994.

[50] J. C. Principe, B. De Vries, and P. G. De Oliveira, "The gamma-filter-a new class of adaptive iir filters with restricted feedback," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 649–656, 1993.

[51] H. H. Ip and C. S. Chan, "Dynamic simulation of human hand motion using an anatomically correct hierarchical approach," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 2, pp. 1307–1312, IEEE, 1997.

[52] O. Nonnarit, A. Barreto, S. Tangnimitchok, N. Ratchatanantakit, *et al.*, "Orientation correction for a 3d hand motion tracking interface using inertial measurement units," in *International Conference on Human-Computer Interaction*, pp. 321–333, Springer, 2018.

[53] D. C. Montgomery, *Design and analysis of experiments*. John wiley & sons, 2017.

[54] A. Field, *Discovering statistics using SPSS*. Sage publications, 2009.

APPENDICES

Appendix A

Source Codes of The Offline Implementation of Orientation Correction Algorithm using Gravity Vector and Magnetic North Vector

Listing A.1: Source Code to Implement The Orientation Correction Algorithm

```
% Read recording data from file
[label,t,Stillness,GyroXYZ,AcceleroXYZ,IMUquat,MagnetoXYZ] =
    readRecordingFile(FILENAME);
% Setting of Parameters
GYRO_THRSHLD = 0.2;
buffSize = 50;
B_SCALING = 1.00;
trigcount = 0;
% Initilization of Arrays
N = length(t); % Number of Samples
SR = N / t(end); % Sampling Rate
gyroBuff = zeros(buffSize,3);
gyroMaxAbsAvg = zeros(N,1);
acceleroBuff = zeros(buffSize,3);
acceleroAvg = zeros(N,3);
magnetoBuff = zeros(buffSize,3);
magnetoAvg = zeros(N,3);
Bias = zeros(N,3);
UnbiasedXYZ = zeros(N,3);
alpha = ones(size(Stillness));
dt = 1/SR; % Sampling Time
qG = zeros(N,4); qG(1,:) = [0 0 0 1];
dqG = zeros(N,4);
qGA = zeros(N,4); qGA(1,:) = [0 0 0 1];
dqGA = zeros(N,4);
qGM = zeros(N,4); qGM(1,:) = [0 0 0 1];
dqGM = zeros(N,4);
qOUT = zeros(N,4);
A_int = [AcceleroXYZ(1,:) 0]; % Measured gravity vector
a4 = zeros(N,4); % Computed Gravity Vector (Pure Quaternion)
a3 = zeros(N,3); % Computed Gravity Vector (3D-vector)
```

```
M_int = [MagnetoXYZ(1,:) 0]; % Measured magnetic North vector
m4 = zeros(N,4); % Computed Magnetic North Vector (Pure Quaternion)
m3 = zeros(N,3); % Computed Magnetic North Vector (3D-vector)
% For every single sample put into the buffer
for i=1:1:N-buffSize
gyroBuff = GyroXYZ(i:i+buffSize-1,:);
gyroMaxAbsAvg(i) = max(abs(mean(gyroBuff)));
acceleroBuff = AcceleroXYZ(i:i+buffSize-1,:);
acceleroAvg(i,:) = mean(acceleroBuff);
magnetoBuff = MagnetoXYZ(i:i+buffSize-1,:);
magnetoAvg(i,:) = mean(magnetoBuff);
if(i~=1)
  Bias(i,:) = Bias(i-1,:);
end
if(gyroMaxAbsAvg(i) < GYRO_THRSHLD)
  trigcount = trigcount+1;
else
  trigcount = 0;
  Bias(i,:) = mean(Bias(1:i-1,:));
end
% Recalculate Bias if gyroMaxAbsAvg(i) < GYRO_THRSHLD for 25 samples
if(trigcount == 25)
  [LL,UL] = deal(i,i+buffSize-1);
  tt=LL:UL;
  b1x = ((tt-mean(tt))*(GyroXYZ(LL:UL,1)-mean(GyroXYZ(LL:UL,1))))/sum((
    tt-mean(tt)).^2);
  b0x = mean(GyroXYZ(LL:UL,1))-b1x*mean(tt);
  Bias(i,1) = b0x+b1x*i;
  b1y = ((tt-mean(tt))*(GyroXYZ(LL:UL,2)-mean(GyroXYZ(LL:UL,2))))/sum((
    tt-mean(tt)).^2);
  b0y = mean(GyroXYZ(LL:UL,2))-b1y*mean(tt);
  Bias(i,2) = b0y+b1y*i;
```

```
61    b1z = ((tt-mean(tt))*(GyroXYZ(LL:UL,3)-mean(GyroXYZ(LL:UL,3))))/sum((
       tt-mean(tt)).^2);
62    b0z = mean(GyroXYZ(LL:UL,3))-b1z*mean(tt);
63    Bias(i,3) = b0z+b1z*i;
64    trigcount = 0;
65  end
66  % Removing Gyroscope Bias
67  UnbiasedXYZ(i,:) = GyroXYZ(i,:) - (B_SCALING * Bias(i,:));
68  % Compute Quaternions
69  if(i~=1)
70    % Augment 0 to Angular Velocity
71    w = [UnbiasedXYZ(i,1),UnbiasedXYZ(i,2),UnbiasedXYZ(i,3),0];
72    % Quaternion Calculations
73    dqG(i,:) = 0.5 * myQuatProd(qG(i-1,:),w);
74    qG(i,:) = myQuatIntegrate(dqG(i,:),qG(i-1,:),dt);
75    qG(i,:) = myQuatNormalize(qG(i,:));
76    dqGA(i,:) = 0.5 * myQuatProd(qGA(i-1,:),w);
77    qGA(i,:) = myQuatIntegrate(dqGA(i,:),qGA(i-1,:),dt);
78    qGA(i,:) = myQuatNormalize(qGA(i,:));
79    dqGM(i,:) = 0.5 * myQuatProd(qGM(i-1,:),w);
80    qGM(i,:) = myQuatIntegrate(dqGM(i,:),qGM(i-1,:),dt);
81    qGM(i,:) = myQuatNormalize(qGM(i,:));
82  end
83  % Compute Gravity Vector (y = q' * m * q)
84  a4(i,:) = myQuatProd(myQuatConj(qGA(i,:)),myQuatProd(A_int,qGA(i,:)));
85  a3(i,:) = a4(i,1:3); % Convert pure quaternion to 3D Vector
86  % Compute Magnetic North Vector (y = q' * m * q)
87  m4(i,:) = myQuatProd(myQuatConj(qGM(i,:)),myQuatProd(M_int,qGM(i,:)));
88  m3(i,:) = m4(i,1:3); % Convert pure quaternion to 3D Vector
89  % Compute Difference in Quaternion
90  v2 = a3(i,:); % Computed Gravity Vector
91  v1 = acceleroAvg(i,:); % Measured Gravity Vector
```

```
92 qv = cross(v1,v2);
93 qw = sqrt( (v1(1)^2+v1(2)^2+v1(3)^2) * (v2(1)^2+v2(2)^2+v2(3)^2) ) +
       dot(v1,v2);
94 deltaQa = myQuatNormalize([qv,qw]);
95 v2 = m3(i,:); % Computed Magnetic North Vector
96 v1 = magnetoAvg(i,:); % Measured Magnetic North Vector
97 qv = cross(v1,v2);
98 qw = sqrt( (v1(1)^2+v1(2)^2+v1(3)^2) * (v2(1)^2+v2(2)^2+v2(3)^2) ) +
       dot(v1,v2);
99 deltaQm = myQuatNormalize([qv,qw]);
100 % Quaternion Interpolation
101 alpha(i) = mean(Stillness(i:i+buffSize-1))^2;
102 qGM(i,:) = myQuatNormalize(myQuatProd(qGM(i,:),deltaQm));
103 qGA(i,:) = myQuatNormalize(myQuatProd(qGA(i,:),deltaQa));
104 qOUT(i,:) = QSLERP(qGM(i,:),qGA(i,:),alpha(i));
105 end
```

Listing A.2: Function to Compute Quaternion Product (myQuatProd.m)

```
1 % syntax: r = myQuatProd(q,w)
2 % where r, q and w are 4-element row vectors
3 % representing quaternion with its real part located at the last
     component.
4 function r = myQuatProd(q,w)
5
6   r = [ w(4), w(3),-w(2),w(1);
7        -w(3), w(4), w(1),w(2);
8         w(2),-w(1), w(4),w(3);
9        -w(1),-w(2),-w(3),w(4)] * transpose(q);
10
11  r = transpose(r);
12 end
```

Listing A.3: Function to Integrate Quaternion Rate (myQuatIntegrate.m)

```
% dq = quaternion rate
% q0 = quaternion from the previous state
% dt = sampling time
% q1 = resultant quaternion
function q1 = myQuatIntegrate(dq,q0,dt)
  w = 2 * myQuatProd(dq,myQuatConj(q0));
  exp_q = myQuatExp((w * dt)/2);
  q1 = myQuatProd(exp_q,q0);
end
```

Listing A.4: Function to Calculate Quaternionic Exponential (myQuatExp.m)

```
function exp_q = myQuatExp(q)
  qvnorm2 = sqrt(q(1)^2 + q(2)^2 + q(3)^2);
  if(qvnorm2 ~= 0)
    exp_q = exp(q(4))*[(sin(qvnorm2)/qvnorm2)*q(1:3), cos(qvnorm2)];
  else
    exp_q = exp(q(4))*[q(1:3), cos(qvnorm2)];
  end
end
```

Listing A.5: Function to Normalize Quaternion (myQuatNormalize.m)

```
function normalized_q = myQuatNormalize(q)
  normalized_q = q/(sqrt(q(1)^2+q(2)^2+q(3)^2+q(4)^2));
end
```

Listing A.6: Function to Conjugate Quaternion (myQuatConj.m)

```
function q_star = myQuatConj(q)
  q_star = [-q(1),-q(2),-q(3),q(4)];
end
```

Appendix B

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

FLORIDA INTERNATIONAL UNIVERSITY

Office of Research Integrity
Research Compliance, MARC 414

MEMORANDUM

To: Dr. Armando Barreto

CC: Nonnarit O-larnnithipong

From: Maria Melendez-Vargas, MIBA, IRB Coordinator

Date: April 23, 2018

Protocol Title: "Hand Motion Tracking using Inertial Measurement Unit and Infrared Cameras"

The Health Sciences Institutional Review Board of Florida International University has re-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-16-0188-CR02	**IRB Approval Date:**	04/18/18
TOPAZ Reference #:	104800	**IRB Expiration Date:**	05/10/19

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.

MMV/em

Appendix C

Adult Consent To Participate In a Research Study

FIU IRB Approval:	04/18/2018
FIU IRB Expiration:	05/10/2019
FIU IRB Number:	IRB-16-0188

FLORIDA INTERNATIONAL UNIVERSITY

ADULT CONSENT TO PARTICIPATE IN A RESEARCH STUDY
Hand Motion Tracking in 3D Space Using Inertial Measurement Unit and Infrared Cameras

PURPOSE OF THE STUDY
You are being asked to be in a research 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 require 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 has the inertial measurement units attached on it. It has a light weight and it is as 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.

FIU IRB Approval:	04/18/2018
FIU IRB Expiration:	05/10/2019
FIU IRB Number:	IRB-16-0188

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 a subject. Research records will be stored securely and only the researcher team will have access to the records. However, your records may be reviewed for audit purposes by authorized University or other agents who will be bound by the same provisions of confidentiality.

COMPENSATION & COSTS
You will not be provided any compensation for your participation.
You will not be responsible for any costs to participate 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. Your withdrawal or lack of participation will not affect any benefits to which you are otherwise entitled. The investigator reserves the right to remove you without your consent at such time that they feel 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 Nonnarit O-larnnithipong at EC 3970, Tel. (305) 348-6072, Email Address: nolar002@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 Date

_____
Printed Name of Participant

_____ _____
Signature of Person Obtaining Consent Date

VITA

NONNARIT O-LARNNITHIPONG

1990	Born, Chonburi, Thailand
2008 - 2011	B.Eng., Mechatronics Engineering Assumption University Samut Prakan, Thailand
2012 - 2013	Assistant Lecturer Assumption University Samut Prakan, Thailand
2013 - 2018	Ph.D., Electrical Engineering Florida International University Miami, Florida
	Graduate Teaching Assistant Florida International University Miami, Florida

PUBLICATIONS AND PRESENTATIONS

Abyarjoo, F., O-larnnithipong, N., Tangnimitchok, S., Ortega, F., Barreto A. (2015). *PostureMonitor: Real-Time IMU Wearable Technology to Foster Poise and Health.* Design, User Experience, and Usability: Interactive Experience Design. Lecture Notes in Computer Science book series. Vol. 9188. pp. 543-552. Aaron Marcus. Springer International Publishing Switzerland.

O-larnnithipong, N., Barreto, A., Ratchatanantakit, N., Tangnimitchok, S., Ortega, F. R. (2018). *Real-Time Implementation of Orientation Correction Algorithm for 3D Hand Motion Tracking Interface.* Universal Access in Human-Computer Interaction. Methods, Technologies, and Users. Lecture Notes in Computer Science book series. Vol. 10907. pp. 228-242. Springer International Publishing Switzerland.

O-larnnithipong, N., Barreto, A., Tangnimitchok, S., Ratchatanantakit, N. (2018). *Orientation Correction for a 3D Hand Motion Tracking Interface Using Inertial Measurement Units.* Human-Computer Interaction. Interaction Technologies. Lecture Notes in Computer Science book series. Vol. 10903. pp. 321-333. Springer International Publishing Switzerland.

O-larnnithipong, N., Barreto, A. (2016). *Gyroscope Drift Correction Algorithm for Inertial Measurement Unit Used in Hand Motion Tracking.* Sensors, 2016 IEEE.

O-larnnithipong, N., Tangnimitchok, S., Barreto A. (2016). *Gyroscope Drift Correction Algorithm for Inertial Measurement Unit Applications in Robotics.* The 29th Florida Conference on Recent Advances in Robotics and Robot Showcase, May 12-13, 2016, Miami, FL.

O-larnnithipong, Nonnarit. Simple Hands-on Project with Unity3D and Oculus Rift. Interaction Design for 3D User Interfaces: The World of Modern Input Devices for Research, Applications, and Game Development. pp. 483-509. Francisco R. Ortega, Fatemeh Abyarjoo, Armando Barreto, Naphtali Rishe, Malek Adjouadi. CRC Press, 2015.

O-larnnithipong, N., Barreto, A., Abyarjoo F. (2014). *Impact of Binaural 3D Sound on Navigation Within a Virtual Environment.* Conferences on Computer, Information, Systems Sciences, and Engineering.

Ortega, F., Barreto A., Rishe N., O-larnnithipong, N., Adjouadi M., Abyarjoo, F. (2015). *GyroTouch: Wrist Gyroscope with a Multi-Touch Display.* Human-Computer Interaction: Interaction Technologies. Lecture Notes in Computer Science book series. Vol. 9170. pp. 262-270. Aaron Marcus. Springer International Publishing Switzerland.

Tangnimitchok, S., O-larnnithipong, N., Ratchatanantakit, N., Barreto, A., Ortega, F. R., Rishe, N. D. (2018). *A System for Non-intrusive Affective Assessment in the Circumplex Model from Pupil Diameter and Facial Expression Monitoring.* Human-Computer Interaction. Theories, Methods, and Human Issues. Lecture Notes in Computer Science book series. Vol. 10901. pp. 465-477. Springer International Publishing Switzerland.

Tangnimitchok, S., O-larnnithipong, N., Barreto, A., Ortega, F. R., Rishe, N. D. (2016). *Finding an Efficient Threshold for Fixation Detection in Eye Gaze Tracking.* Human-Computer Interaction. Interaction Platforms and Techniques. Lecture Notes in Computer Science book series. Vol. 9732. pp. 93-103. Masaaki Kurosu. Springer International Publishing Switzerland.