Upper Body Joint Angle Calculation and Analysis Using Multiple inertial Measurement Units

Aaron S. Freedkin
asf0705@gmail.com

Follow this and additional works at: https://huskiecommons.lib.niu.edu/allgraduate-thesesdissertations

Recommended Citation
https://huskiecommons.lib.niu.edu/allgraduate-thesesdissertations/7041
ABSTRACT

UPPER BODY JOINT ANGLE CALCULATION AND ANALYSIS USING MULTIPLE INERTIAL MEASUREMENT UNITS

Aaron Freedkin, M.S.
Department of Mechanical Engineering
Northern Illinois University, 2022
Dr. Ji-Chul Ryu, Director

A prominent area in biomechanics revolves around finding solutions to common issues like work-related musculoskeletal disorders (WMSDs). These are common in professions associated with unnatural postures like commercial fishing and farming. Understanding how these professions move on a day-to-day basis can help find solutions to WMSDs. While a motion capture system might be the most common equipment for body posture measurement, it lacks the portability to work in remote environments like commercial fishing and farming. Therefore, joint angle estimation using IMUs (Inertial Measurement Units) could provide a potential alternative.

An IMU consists of a triaxial accelerometer, gyroscope, and magnetometer, which can each be used to build an estimation of the sensor orientation with respect to the world/fixed frame of reference. To estimate orientations, the gyroscope data of angular velocity can be numerically integrated, or the accelerometer together with the magnetometer can be used. However, these sensors are susceptible to gyroscope drift due to the integration process, accelerometer noise during dynamic conditions, and magnetic hard and soft iron distortions, respectively.
In this thesis, three sensor fusion algorithms that have been developed to account for the issues while combining the benefits associated with each sensor were applied to estimate joint angles of the shoulder and arm effectively. Madgwick’s filter utilizes the gradient descent algorithm to quickly narrow in on an orientation estimation. Kalman filters are an iterative two-step process involving a prediction followed by an update, using a knowledge of measurement uncertainties and their Gaussian distributions in each iteration. Complementary filters use a weighted average to combine the sensor data one at a time and find an angle and axis of rotation to tilt the orientation closer to the actual value.

To demonstrate the application of the three algorithms for arm motion tracking, three IMUs were placed on each part of an arm (lower, upper, and shoulder). The data was then post-processed to estimate the elbow and shoulder joint angles. The estimations were compared with the reference data collected using a motion capture system. Furthermore, parameters for each sensor fusion algorithm were determined to minimize error and compare algorithm accuracy.
NORTHERN ILLINOIS UNIVERSITY
DE KALB, ILLINOIS

DECEMBER 2022

UPPER BODY JOINT ANGLE CALCULATION AND ANALYSIS USING
MULTIPLE INERTIAL MEASUREMENT UNITS

BY

AARON FREEDKIN
© 2022 Aaron Freedkin

A THESIS SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE
MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL ENGINEERING

Thesis Director:
Dr. Ji-Chul Ryu
ACKNOWLEDGEMENTS

I’d like to acknowledge my advisor, Dr. Ji-Chul Ryu, for guiding me through complex topics and confusing procedures, the mechanical engineering staff of Northern Illinois University, who pushed me to be a well-versed and comprehensive engineer, Dr. Jaejin Hwang for allowing me access to the motion capture system used for data validation in this masters thesis, Vishal Garela for assisting in the operation of the motion capture system, and Dr. Brianno Coller for his input on my thesis and overall participation in the thesis committee.
DEDICATION

To my parents, who pushed me to pursue my passions, inspired me to be the best that I can be, and instilled a motivation for success.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation and Objective</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Literature Review</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Outline</td>
<td>4</td>
</tr>
<tr>
<td>2 BASIC THEORETICAL BACKGROUND</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Inertial Measurement Unit</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Rotation Representation using a Rotation Matrix</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Parameterization of Rotations</td>
<td>6</td>
</tr>
<tr>
<td>2.3.1 Euler Angles &amp; Pitch, Roll, Yaw</td>
<td>7</td>
</tr>
<tr>
<td>2.3.2 Angle-Axis Notation</td>
<td>8</td>
</tr>
<tr>
<td>2.3.3 Quaternions</td>
<td>8</td>
</tr>
<tr>
<td>3 FILTERS FOR OPTIMAL RESULTS</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Madgwick’s Filter</td>
<td>11</td>
</tr>
<tr>
<td>3.1.1 Calculating orientation from gyroscope measurements</td>
<td>13</td>
</tr>
<tr>
<td>3.1.2 Calculating orientation from a homogeneous field</td>
<td>13</td>
</tr>
<tr>
<td>3.1.3 Sensor Fusion</td>
<td>17</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>3.2 Kalman Filter</td>
<td>18</td>
</tr>
<tr>
<td>3.2.1 General Kalman Filter Construction</td>
<td>18</td>
</tr>
<tr>
<td>3.2.2 System Variables and Parameters</td>
<td>19</td>
</tr>
<tr>
<td>3.2.3 System Model Construction</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Complementary Filter</td>
<td>23</td>
</tr>
<tr>
<td>3.3.1 Frame Transformation of Accelerometer and Magnetometer Data</td>
<td>23</td>
</tr>
<tr>
<td>3.3.2 Correction Quaternion Calculation</td>
<td>24</td>
</tr>
<tr>
<td>4 EXPERIMENTAL RESULT</td>
<td>28</td>
</tr>
<tr>
<td>4.1 Materials</td>
<td>28</td>
</tr>
<tr>
<td>4.1.1 Inertial Measurement Unit</td>
<td>28</td>
</tr>
<tr>
<td>4.1.2 Motion Capture System</td>
<td>28</td>
</tr>
<tr>
<td>4.2 Experimental Setup and Procedure</td>
<td>31</td>
</tr>
<tr>
<td>4.3 Similarity Transformation for the IMU and the Motion Capture System</td>
<td>33</td>
</tr>
<tr>
<td>4.4 Experimental Results of Three Filters</td>
<td>34</td>
</tr>
<tr>
<td>4.4.1 Madgwick’s Gradient Descent Algorithm</td>
<td>35</td>
</tr>
<tr>
<td>4.4.2 Kalman Filter</td>
<td>36</td>
</tr>
<tr>
<td>4.4.3 Complementary Filter</td>
<td>39</td>
</tr>
<tr>
<td>4.5 Joint Angles Estimation</td>
<td>41</td>
</tr>
<tr>
<td>5 DISCUSSIONS AND CONCLUSIONS</td>
<td>50</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>51</td>
</tr>
<tr>
<td>APPENDIX: CODE</td>
<td>55</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>44.</td>
</tr>
<tr>
<td>4.2</td>
<td>44.</td>
</tr>
<tr>
<td>4.3</td>
<td>45.</td>
</tr>
<tr>
<td>4.4</td>
<td>46.</td>
</tr>
</tbody>
</table>
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Tilt quaternion visualization</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>LPMS-B2 Inertial Measurement Unit</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>NIU Wellness &amp; Ergonomic Laboratory</td>
<td>30</td>
</tr>
<tr>
<td>4.3</td>
<td>OptiTrack Motion Capture Origin Plate</td>
<td>31</td>
</tr>
<tr>
<td>4.4</td>
<td>3D Printed marker plate</td>
<td>32</td>
</tr>
<tr>
<td>4.5</td>
<td>Reaching up dataset pictures from left to right: Initial position from the</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>back, initial position from the side, final position from the back, final</td>
<td></td>
</tr>
<tr>
<td></td>
<td>position from the side</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>Desk reach dataset pictures from left to right: Initial position from the</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>back, initial position from the side, final position from the back, final</td>
<td></td>
</tr>
<tr>
<td></td>
<td>position from the side</td>
<td></td>
</tr>
<tr>
<td>4.7</td>
<td>Marker plates during coordinate frame alignment</td>
<td>34</td>
</tr>
<tr>
<td>4.8</td>
<td>Reaching up dataset - Madgwick filter error for each IMU while varying β</td>
<td>35</td>
</tr>
<tr>
<td>4.9</td>
<td>Desk reach dataset - Madgwick filter error for each IMU while varying β</td>
<td>36</td>
</tr>
<tr>
<td>4.10</td>
<td>Reaching up dataset - Madgwick filter pitch, roll, and yaw for each sensor</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>using optimal β values</td>
<td></td>
</tr>
<tr>
<td>4.11</td>
<td>Desk reach dataset - Madgwick filter pitch, roll, and yaw for each sensor</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>using optimal β values</td>
<td></td>
</tr>
<tr>
<td>4.12</td>
<td>Reaching up dataset - Kalman filter pitch, roll, and yaw for each sensor</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>using optimal covariance matrices</td>
<td></td>
</tr>
<tr>
<td>4.13</td>
<td>Desk reach dataset - Kalman filter pitch, roll, and yaw for each sensor</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>using optimal covariance matrices</td>
<td></td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4.14</td>
<td>Reaching up dataset - Complimentary filter error for each IMU with varied α values</td>
<td>42</td>
</tr>
<tr>
<td>4.15</td>
<td>Desk reach dataset - Complimentary filter error for each IMU with varied α values</td>
<td>43</td>
</tr>
<tr>
<td>4.16</td>
<td>Reaching up dataset - Complimentary filter pitch, roll, and yaw for each sensor using optimal α values</td>
<td>44</td>
</tr>
<tr>
<td>4.17</td>
<td>Desk reach dataset - Complimentary filter pitch, roll, and yaw for each sensor using optimal α values</td>
<td>45</td>
</tr>
<tr>
<td>4.18</td>
<td>Reaching up dataset - All sensor fusion methods for each sensor using optimal parameter values</td>
<td>46</td>
</tr>
<tr>
<td>4.19</td>
<td>Desk reach dataset - All sensor fusion methods for each sensor using optimal parameter values</td>
<td>47</td>
</tr>
<tr>
<td>4.20</td>
<td>Reaching up dataset - All sensor fusion method elbow joint angles using optimal parameter values</td>
<td>48</td>
</tr>
<tr>
<td>4.21</td>
<td>Desk reach dataset - All sensor fusion method elbow joint angles using optimal parameter values</td>
<td>48</td>
</tr>
<tr>
<td>4.22</td>
<td>Reaching up dataset - All sensor fusion method shoulder joint angles using optimal parameter values</td>
<td>49</td>
</tr>
<tr>
<td>4.23</td>
<td>Desk reach dataset - All sensor fusion method shoulder joint angles using optimal parameter values</td>
<td>49</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

1.1 Motivation and Objective

Workplace musculoskeletal disorders (WMSDs) are a common and serious problem that can affect almost any workplace that puts workers in unnatural postures [1]. These disorders can present themselves as injuries to muscles, nerves, tendons, joints, cartilage and spinal discs [2]. Workers in environments that encounter these injuries are often trained to identify and isolate these WMSDs. Unfortunately, while WMSDs are a fairly prominent issue in workplaces around the world, the main method for identifying them has been simple self-report and observational methods. With more focus on body posture estimation prior to the development of WMSDs, advances may be made towards prevention of common WMSDs.

Optical motion capture systems are the most common method for body posture estimation, however these systems are large and bulky, requiring a contained environment with multiple calibrated cameras directed at the same region. For remote or isolated environments such as commercial fishing and farming, these systems are not an ideal option for body posture estimation. Therefore, inertial measurement units (IMUs) provide a portable alternative to body posture estimation in order to help prevent common WMSDs. An IMU consists of a triaxial accelerometer, gyroscope, and magnetometer. Through sensor fusion algorithms, these sensors can be used to determine the orientation of the sensor with respect to a fixed reference frame.
While one IMU is able to track the orientation of a rigid body, full body posture estimation requires multiple IMUs. For accurate body posture estimation, each IMU will need to accurately estimate orientation. This necessitates more in-depth analysis of sensor fusion methods used to fuse accelerometer, magnetometer, and gyroscope data.

Therefore, the main objective of this thesis is to apply three sensor fusion methods (Madgwick’s gradient descent algorithm, a complimentary filter, and a Kalman filter) for joint angle estimations, and determine the accuracies of each method. All methods will utilize a quaternion orientation representation to reduce the effect of singularities and each sensor fusion method is covered in depth. Parameters for each method are determined to minimize total error, and all data will be verified with an optical motion capture system.

1.2 Literature Review

There are many sensor fusion algorithms for IMUs, and because of this, many comparisons have also been made. Bergamini et al. [3] compared the effectiveness of an extended Kalman filter (EKF) and a nonlinear observer sensor fusion methods under different experimental conditions (time duration, measurement volume, out-of-plane movement, static/dynamic conditions) with one IMU. Bleser et al. [4] compared four methods for fusing IMU data including a Kalman filter, unscented Kalman filter, and an extended Kalman filter. Each method was then used to reconstruct arm motion and results compared. Madgwick et al. [5] developed a quaternion-based gradient descent approach for fusing IMU accelerometer, magnetometer, and gyroscope measurements to produce an accurate orientation estimation. Luinge et al. [6] developed a sensor fusion algorithm using accelerometer and gyroscope measurements in a Kalman filter. In this method, gyroscope offset is continuously recalibrated and gyroscope drift is continuously corrected using an inclination estimate repeatedly updated.
from accelerometer readings. Deep et al. [7] proposes a novel approach for utilizing smartphone based IMU orientation and displacement estimations to compliment GPS data when GPS data is momentarily unavailable. Huang et al. [8] developed a quaternion based complimentary filter for accelerometer and gyroscope data, designed for attitude estimation of agricultural implements. Antonov et al. [9] presents a novel method for predictive orientation tracking to improve time lag for a more user friendly experience with the Oculus Rift. Lovell et al. [10] developed a geometrically intuitive quaternion complimentary filter that is computationally inexpensive compared to Madgwick’s algorithm. Sola et al. [11] covers in depth theoretical background regarding quaternion representation and operations as well as applications for IMUs through error-state Kalman filtering. Adjouadi et al. [12] created this book to provide a more intuitive understanding of Kalman filtering with examples and source code dealing with processing IMU data to create a quaternion output. 

IMUs have been used very often in recent years to calculate joint angles. Aissaoui et al. [13] used accelerometer and gyroscope measurements from two IMUs to implement a calibration sequence that improves knee joint angle calculation accuracy. Cutti et al. [14] developed a protocol using four IMUs for assessing scapulothoracic (scapula), humerothoracic (position of the arm relative to the chest) and elbow 3D kinematics. Kim et al. [15] used two IMUs to assess the time-domain characteristics of cervical motion in real time and determine the cervical range of motion to help clinical evaluation of spines. Chen et al. [16] created a data glove with 18 IMUs for real time hand function evaluation. This work was specifically designed to evaluate hand functions for the medical field and utilizes madgwick’s gradient descent algorithm for sensor fusion. Barim et al. [17] developed a five IMU full body system for automatically identifying lifting task duration. Barim et al. also utilized subject joint lengths to improve result accuracy. Hu et al. [18] develops a technique that combines sensor fusion and an annealing optimization technique to estimate elbow and shoulder joint angles for stroke patients. Cappozzo et al. [19] developed an anatomical calibration technique
to assist in lower limb kinematic estimation. Hu et al. [20] also used accelerometer and gyroscope measurements from two IMUs combined with joint lengths in a Kalman filter to improve joint angle estimation accuracy. Harris et al. [21] uses two IMUs combined with upper arm kinematics and a Lagrangian based optimization approach to accurately estimate upper arm positional data.

It has also been fairly common to put multiple IMUs under identical conditions (on the same joint) to improve the result by considering them to be one IMU. Cartiaux et al. [22] used 7 IMUs to compare four functional (sensor-to-segment) calibration methods in order to improve accuracy of lower limb joint angle calculations. Basset et al. [23] uses three IMUs in a triad structure to improve overall orientation estimation through an unscented Kalman filter (UKF). Handel et al. [24] created an array of 18 ultra-low-cost IMUs to improve displacement estimation using a weighted average. Coopmans et al. [25] developed an estimation-domain fusion strategy using an extended Kalman filter (EKF) to improve orientation estimation accuracy by fusing data from two IMUs on a small unmanned aerial system (sUAS). Rasoulzadeh et al. [26] uses a discrete Kalman filter and four planar-mounted low cost IMUs to improve angular velocity (gyroscope) accuracy.

1.3 Outline

The rest of the thesis is organized as follows. The in-depth explanation of IMUs, and various rotation representations including quaternions are discussed in Chapter 2. The processes of the three sensor fusion methods used in this thesis are discussed in Chapter 3. Experimental analysis methodology, coefficient values analyzed for each method, and experimental results are discussed in Chapter 4. Finally, concluding remarks are stated in Chapter 5.
CHAPTER 2
BASIC THEORETICAL BACKGROUND

2.1 Inertial Measurement Unit

An inertial measurement unit (IMU) is a small sensor typically comprising of a triaxial accelerometer, a triaxial gyroscope, and a triaxial magnetometer. These measurements can be fused in different manners to determine the orientation of the sensor. These sensors are widely used in industry wherever knowledge of orientation is necessary, such as the robotics, aerospace, and automotive industries.

With an IMU, orientation estimates can be created using either the accelerometer combined with the magnetometer, or by using solely the gyroscope. For the accelerometer and magnetometer based orientation, we essentially calculate the orientation based on the knowledge of two vectors in 3D space. The first vector is Earth’s gravity vector, which is sensed using the accelerometer measurements. However, while the gravity vector may work for calculating orientation with respect to vectors perpendicular to gravity, it is unable to determine orientation about the up/down axis, as this is always collinear to the gravity vector. For this, we require the magnetometer measurements. Additionally, orientation can be calculated using the gyroscope measurements. Since the gyroscope measurements output angular velocity, by integrating these values, one can detect angular position. This is covered in more depth in Chapter 3.
2.2 Rotation Representation using a Rotation Matrix

Rotations can be generally defined as the relationship between two coordinate frames. With an IMU, the two frames in question are the body frame, or the frame that corresponds to the IMU sensor that rotates, and the reference frame, which is fixed or inertial such as the Earth (World) frame. In this thesis, the fixed frame is assigned through North-East-Down (NED) representation.

We can quantify the rotation between these frames using a matrix known as a rotation matrix. This matrix is essentially a 3-dimensional projection of the body frame onto the fixed frame which is given by

\[ R = \begin{bmatrix}
    x_1 \cdot x_0, x_1 \cdot y_0, x_1 \cdot z_0 \\
    y_1 \cdot x_0, y_1 \cdot y_0, y_1 \cdot z_0 \\
    z_1 \cdot x_0, z_1 \cdot y_0, z_1 \cdot z_0
\end{bmatrix} \]  

(2.1)

where \( x_0, y_0, z_0 \), and \( x_1, y_1, z_1 \), denote the basis vectors of frame 0 and frame 1 respectively.

While rotation matrices are a common orientation representation due to the ease of use, 9 values to represent orientation adds a level of computation complexity that can be avoided with other methods.

2.3 Parameterization of Rotations

Alternatively, all rotations can be defined using only three independent quantities. Consequently, there are several rotation representations available other than a rotation matrix. These methods are Euler Angles, Roll-Pitch-Yaw, and Angle-Axis notation. Since all of these
methods face singularities in the real world, we discuss another representation in Chapter 2.3.3 and use it in this thesis: quaternions.

### 2.3.1 Euler Angles & Pitch, Roll, Yaw

Euler angles [27] describe an orientation through three separate rotations about individual principal axes. By forming individual rotation matrices for each axis rotation, these can be multiplied together to create a rotation matrix. This overall rotation matrix can then be solved for angles of $\phi$, $\theta$, and $\psi$. For example, in the case of the ZYX convention

\[
R = R_z(\psi)R_y(\theta)R_x(\phi) =
\begin{bmatrix}
    c\theta c\psi & s\theta s\psi c\phi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\
    c\theta s\psi & s\theta s\psi c\phi + c\phi s\psi & c\phi s\theta s\psi - s\phi c\psi \\
    -s\theta & s\phi c\theta & c\phi c\theta
\end{bmatrix}
\] (2.2)

The difference between Euler angles and the Roll-Pitch-Yaw representations is about whether the individual subsequent rotations are with respect to the fixed frame or the current frame. This makes a difference in the multiplication of individual rotation matrices to compute the overall rotation matrix. Euler angles are with respect to the current frame, while roll, pitch, and yaw are with respect to the fixed frame. Euler angles use post multiplication, while Roll, Pitch, and Yaw use pre-multiplication.

Both methods are however, not ideal as they are susceptible to singularities at certain angles. A common example of the singularity one may face is called gimbal lock [27].
2.3.2 Angle-Axis Notation

Another method commonly used to parameterize rotations is the angle-axis representation. The idea behind this method is that any rotation can be considered a pure rotation about one axis. By defining the position of the axis using two angles as well as the rotation about the axis, we have three independent parameters that completely define the rotation.

However, this method is also susceptible to singularities if the rotation is at 180° about the axis of rotation.

2.3.3 Quaternions

To avoid the singularities in the methods mentioned in the previous sub-chapters, quaternions can be used. While the previous methods define a rotation using three parameters, quaternions use four parameters. Similar to angle-axis representation, these four parameters define an axis of pure rotation and a rotation about this axis. However, unlike angle-axis representation, quaternions use three parameters (instead of two) to define the axis. This prevents singularities universally. A quaternion is defined as

\[ A_B^q = \begin{bmatrix} \cos(\frac{\theta}{2}), n_x \sin(\frac{\theta}{2}), n_y \sin(\frac{\theta}{2}), n_z \sin(\frac{\theta}{2}) \end{bmatrix} \]

(2.3)

where \( n_x, n_y, \) and \( n_z \) defines the vector of axis of rotation and \( \theta \) is the rotation about that axis. Additionally, by flipping the axis to its negative, we get the quaternion conjugate, \( A_B^q \) defined as

\[ A_B^{q^*} = B_A^q = \begin{bmatrix} \cos(\frac{\theta}{2}), -n_x \sin(\frac{\theta}{2}), -n_y \sin(\frac{\theta}{2}), -n_z \sin(\frac{\theta}{2}) \end{bmatrix} \]

(2.4)
where frame A represents the inertial frame, and frame B represents the body frame.

Quaternions are defined as a 4-dimensional complex number, where Eq. (2.3) can be simplified to

\[ q_w + q_x i + q_y j + q_z k \]  

(2.5)

where \( i^2 = j^2 = k^2 = -1 \). Using the nature of complex numbers and the multiplications among \( i,j \), and \( k \) defined as in Eqs. (2.6)-(2.8), quaternion multiplication (which is denoted by \( \otimes \)), can be defined.

\[ i = jk = -kj \]  

(2.6)

\[ j = ki = -ik \]  

(2.7)

\[ k = ij = -ji \]  

(2.8)

For a more detailed understanding of quaternion multiplication, refer to Sola et al. [11].

Using quaternions and their conjugates, coordinate transformations can be performed relatively easily. For example, let \( F_A \) represent the world frame, and \( F_B \) represent the body frame. \( A_B^q \) represents the orientation of \( F_B \) relative to \( F_A \). If the coordinates of a point \( p \) expressed in \( F_A \) and the coordinates of the point expressed in \( F_B \) is desired, this can be calculated according to

\[ p^B = A_B^q \otimes p^A \otimes A_B^{q^*} \]  

(2.9)

This methodology is also applied for vector transformations, which is commonly used in Chapter 3.
Additionally, the composition of two rotations can be simply done. Consider three separate coordinate frames, A, B, and C. With knowledge of $A_B^q$ and $B_C^q$, $A_C^q$ is given by

$$A_C^q = A_B^q \otimes B_C^q \quad (2.10)$$

Compared to rotation matrices, the computational complexity of quaternions reduces to less than a half.

As used in Chapter 3, it is also important to know the conversion between quaternions and Euler angles. The conversion with the ZYX Euler angle convention is given by

\[
\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4 \\
\end{bmatrix} =
\begin{bmatrix}
\cos \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2} \\
\sin \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} - \cos \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2} \\
\cos \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} \\
\cos \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} - \sin \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} \\
\end{bmatrix}
\quad (2.11)
\]

\[
\begin{bmatrix}
\phi \\
\theta \\
\psi \\
\end{bmatrix} =
\begin{bmatrix}
\text{atan2}(2(q_2q_3 + q_1q_4), 1 - 2(q_3^2 + q_4^2)) \\
- \sin^{-1}(2(q_2q_4 + q_1q_3)) \\
\text{atan2}(2(q_3q_4 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \\
\end{bmatrix}
\quad (2.12)
\]

where $\text{atan2}()$ is the two-argument arctangent function.

Quaternions have some additional advantages regarding gyroscope readings. The rate of change of quaternions can be readily available from the angular velocity of the system. This is discussed in Chapter 3.
CHAPTER 3
FILTERS FOR OPTIMAL RESULTS

Orientation can be calculated using either gyroscope measurements alone, or as a combination of the accelerometer and magnetometer measurements. However, a problem lies with each method. Gyroscope measurements are prone to a natural phenomena called gyroscope drift, where values slowly deviate from the true values and therefore lose accuracy over time [20]. Similarly, magnetometer measurements are prone to noise due to magnetic interference from hard iron and soft iron sources [5]. Accelerometer measurements are also unreliable on their own, because for an orientation to be estimated from these measurements, gravity is assumed to be the only acceleration acting on the IMU. This means that error is encountered whenever the IMU is in motion.

Therefore, there are filtering methods which take the good parts of each sensor and combine them to remove the negative effects. In this chapter, three sensor fusion methods including Madgwick’s gradient descent algorithm, a Kalman filter, and a complementary filter are discussed and compared.

3.1 Madgwick’s Filter

This sensor fusion method is directly adopted and summarized from Sebastian Madgwick’s work [5]. As a general guide for understanding the information provided in this chapter, any variable with a superscript E: (e.g., $E_x$) corresponds to the Earth/fixed frame. Similarly, any variable with a superscript S: (e.g., $S_x$) corresponds to the sensor/body frame. Additionally,
any variable with a hat on top: (e.g., $\hat{x}$) means it is an estimate and any variable with a dot on top: (e.g., $\dot{x}$) represents the first derivative of the variable.

Gradient descent is a commonly used optimization technique where a calculated gradient is used to converge on the true solution. This method works through the development of a cost function meant to converge to zero. This convergence is typically achieved through a calculated gradient continuously pushing the data closer and closer to the true value.

Madgwick’s gradient descent algorithm shows the quick convergence of the estimated data to the real data in general. This is accomplished through the development of a cost function that uses the previously estimated orientation (calculated from gyroscope, accelerometer, and magnetometer data) to guarantee convergence of accelerometer data and magnetometer data to their corresponding reference vectors (gravity and Earth’s magnetic field respectively).

Using quaternions, the cost function can be defined as

$$f(\hat{q}, \dot{E}_d, S_s) = \hat{q}^* \otimes ^E d \otimes \hat{q} - S_s$$

where $q$ and $q^*$ are a quaternion and its conjugate, derived according to Eqs. (2.3) and (2.4), respectively.

As seen here, the cost function is calculated using sensor measurements $S_s$ in the sensor/body frame, corresponding measurements in the fixed/world frame based on knowledge of the homogeneous field, $E_d$, and the previous quaternion-based orientation estimate. Because $S_s$ and $E_d$ are meant to be equivalent values in different frames, bringing the value of $f(\hat{q}, \dot{E}_d, S_s)$ close to zero will guarantee convergence of the orientation estimation, $\hat{q}$. 

3.1.1 Calculating orientation from gyroscope measurements

An estimate in orientation using quaternions can be found from gyroscope data as follows [11]. First, the gyroscope values are converted to a quaternion form given by

\[ S_\omega = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix} \] (3.2)

Then, the rate of the quaternion is given by

\[ \dot{\hat{q}}_k = \frac{1}{2} \hat{q}_{k-1} \otimes S_\omega \] (3.3)

where \( \hat{q}_{k-1} \) denotes the estimated quaternion at the previous time step.

Finally, the updated quaternion estimate based on gyroscope data can be found through numerical integration.

\[ \hat{q}_k = \hat{q}_{k-1} + \dot{\hat{q}}_k \Delta t \] (3.4)

3.1.2 Calculating orientation from a homogeneous field

Calculating orientation using data from a homogeneous field is the core part in this algorithm. A homogeneous field is a field vector that doesn’t change over time. In our case, Earth’s gravitational and electromagnetic fields stay constant (in the same geographical region), allowing us to sense them using accelerometer and magnetometer sensors. By minimizing the cost function \( f(\hat{q}, E_d, S_s) \) through an iterative process, we can calculate the next quaternion estimate. The general equation for updating the orientation estimate can be
described as the previous orientation plus a step element. This step element incorporates the normalized cost function gradient multiplied by an adjustable parameter $\beta$ added to the gyroscope-based quaternion derivative. The gyroscope-based quaternion derivative sets the direction for the step element while the cost function gradient serves as a trust value that pushes the direction away from the gyroscopic drift. This equation is written as

$$\hat{q}_k = \hat{q}_{k-1} + (\hat{q}_{k-1} - \beta \frac{\nabla f}{||\nabla f||}) \Delta t, \ k = 1, 2...n$$ (3.5)

where $\nabla f$ is the gradient of the cost function, and is computed using the cost function and its corresponding Jacobian.

$$\nabla f(\hat{q}_k, E^d, S_s) = J^T(\hat{q}_k, E^d)f(\hat{q}_k, E^d, S_s)$$ (3.6)

More specifically, the cost function and subsequent equations can be applied to any measurements stemming from a homogeneous field, which includes both accelerometer and magnetometer measurements. To build a cost function with accelerometer measurements defined in Eq. (3.1), we input $E_g$ for $E^d$ and $S_a$ for $S_s$ as

$$E_g = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$ (3.7)

$$S_a = \begin{bmatrix} 0 & a_x & a_y & a_z \end{bmatrix}$$ (3.8)
where $Eg$ denotes the gravity vector in the Earth/fixed frame, and $Sa$ denotes accelerometer measurements in the sensor frame. This yields the cost function and corresponding Jacobian.

$$f_g(\hat{q}, Eg, Sa) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - a_x \\ 2(q_1q_2 + q_3q_4) - a_y \\ 2(\frac{1}{2} - q_2^2 - q_3^2) - a_z \end{bmatrix} \tag{3.9}$$

$$J_g(\hat{q}) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{3.10}$$

Similarly, for magnetometer measurements we can form a separate cost function and Jacobian. This can be done using knowledge of Earth’s magnetic field measurements from a set position combined with magnetometer measurements. However, magnetic distortion is a common problem affecting magnetometer measurements. There are magnetic interferences from nearby electromagnetic fields, typically called hard iron sources, that add/subtract to the magnetic field, and interferences caused by nearby metals like nickel and iron, called soft iron sources that distort/deflect the magnetic field. Madgwick used this as an opportunity to improve the accuracy of the detected magnetic field of Earth using the accelerometer. By converting the magnetometer measurements into the world frame, and normalizing the data along Earth’s x and z axes (the magnetic world frame is aligned such that Earth’s magnetic field acts along the sensors x and z plane), the magnetic distortion is being compensated for. To accomplish this, we use the estimated quaternion at the previous time step, $\hat{q}_{k-1}$ to convert the magnetometer measurements to the world frame.
\[ E_{h_k} = \begin{bmatrix} 0 & h_x & h_y & h_z \end{bmatrix} = \hat{q}_{k-1} \otimes S_m \otimes \hat{q}^*_k \]  

(3.11)

where \( S_m \) is the raw magnetometer data in the sensor frame and \( E_{h_k} \) is the raw magnetometer data oriented in the world frame.

Then we normalize this world frame representation of the magnetometer data along the x and z axes to give us our distortion-compensated reference vector in the world frame for the magnetometer cost function, \( E_b \).

\[ E_b = \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & 0 & h_z \end{bmatrix} \]  

(3.12)

Now we substitute \( E_b \) for \( E_d \) and \( S_m \) for \( S_s \) for the cost function defined in Eq. (3.1) as

\[ S_m = \begin{bmatrix} 0 & m_x & m_y & m_z \end{bmatrix} \]  

(3.13)

resulting in the cost function and corresponding Jacobian given by

\[
\begin{align*}
    f_b(\hat{q}, E_b, S_m) &= \begin{bmatrix} 2b_x(\frac{1}{2} - q_3^2 - q_4^2) + 2b_z(q_2q_4 - q_1q_3) - m_x \\
    2b_x(q_2q_3 - q_1q_4) + 2b_z(q_1q_2 + q_3q_4) - m_y \\
    2b_x(q_1q_3 + q_2q_4) + 2b_z(\frac{1}{2} - q_2^2 - q_3^2) - m_z 
\end{bmatrix} \\
    J_b(\hat{q}, E_b) &= \begin{bmatrix} -2b_xq_3 & 2b_xq_4 & -4b_xq_3 - 2b_zq_1 & -4b_xq_4 + 2b_zq_2 \\
    -2b_xq_4 + 2b_zq_2 & 2b_xq_3 + 2b_zq_1 & 2b_xq_2 + 2b_zq_4 & -2b_xq_1 + 2b_zq_3 \\
    2b_xq_3 & -2b_xq_4 + 4b_zq_2 & 2b_xq_1 - 4b_zq_3 & 2b_xq_2 \end{bmatrix}
\end{align*}
\]

(3.14)  

(3.15)
In order to use both sets of measurements simultaneously, the cost functions and their Jacobians can be combined by simply stacking them into new matrices such that

\[
f_{g,b}(\hat{\mathbf{q}}, E_g, E_b, S_a, S_m) = \begin{bmatrix} f_g \\ f_b \end{bmatrix} \tag{3.16}
\]

\[
J_{g,b}(\hat{\mathbf{q}}, E_b) = \begin{bmatrix} J^T_g \\ J^T_b \end{bmatrix} \tag{3.17}
\]

### 3.1.3 Sensor Fusion

Finally, we can calculate the updated quaternion estimates from the homogeneous fields using the following equations.

\[
\hat{\mathbf{q}}_k = \hat{\mathbf{q}}_{k-1} + \dot{\mathbf{q}}_k \Delta t, \tag{3.18}
\]

\[
\dot{\mathbf{q}}_k = \dot{\mathbf{q}}^w_{k-1} - \beta \frac{\nabla f}{\| \nabla f \|} \tag{3.19}
\]

where the gradient of the cost function is calculated as

\[
\nabla f = J^T_{g,b} f_{g,b} \tag{3.20}
\]
Where, in Eq. (3.19) $\beta$ can be considered a step size for the iteration. An ideal value for $\beta$ that ensures the convergence rate of $\dot{q}$ is limited to the maximum physical orientation rate and is given by

$$\beta = \sqrt{\frac{3}{4}} \dot{\omega}_{max}$$  \hspace{1cm} (3.21)

### 3.2 Kalman Filter

The Kalman filter is a two step process, where a state and corresponding error is initially predicted, and then updated after a Kalman gain is calculated using these predictions. This is considered to be an optimal estimation algorithm and is therefore widely used in industry.

#### 3.2.1 General Kalman Filter Construction

First, a prediction is made for the new estimate based on a state-transition matrix, $A$. This state-transition matrix varies by system, as it’s from the system model. Using the state-transition matrix, we can have the prediction of the state (a priori state estimate) at the next time-step before a more accurate (a posteriori) estimate can be created.

$$\hat{x}_{k} = A\hat{x}_{k-1}$$  \hspace{1cm} (3.22)

The error covariance can then be predicted using the previous error covariance matrix, the state-transition matrix, and a process noise covariance matrix, $Q$. 
\[ P_k^- = AP_{k-1}A^T + Q \] (3.23)

The Kalman gain can be calculated as

\[ K_k = P_k^- H^T \cdot (HP_k^- H^T + R)^{-1} \] (3.24)

where \( H \) denotes the state-to-measurement matrix and \( R \) denotes the measurement noise covariance matrix.

Basically, the Kalman gain determines how heavily the a priori estimate and the measurement contribute to the calculation of the estimate, \( \hat{x}_k \).

Finally, the estimated state and error covariance can be updated using Eqs. (3.22) and (3.24) and the measurement \( z_k \) at the current time step such that

\[ \hat{x}_k = \hat{x}_k^- + K_k [z_k - H\hat{x}_k^-] \] (3.25)

\[ P_k = P_k^- - K_k H P_k^- \] (3.26)

### 3.2.2 System Variables and Parameters

In this subchapter and the next, we discuss how specially to design the Kalman filter for the thesis work. First, the filter requires inputs of:

- \( x_0 \): an state
• $P_0$: an initialized process error matrix

• $z_k$: an accelerometer and magnetometer measurement at every time step

• $R$: a measurement uncertainty covariance matrix

• $Q$: a process noise covariance matrix

• $S^\omega$: Gyroscope measurements to be used for the state transition matrix, $A$

In the designed Kalman filter, a quaternion orientation estimate serves as the state, $x$.

$$x = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix}$$ (3.27)

The measurement $z_k$ is a quaternion estimate created from accelerometer and magnetometer data. The raw sensor data is first converted to corresponding Euler angles based on the equations in Chapter 3 in [28]. These Euler angles are then converted to a quaternion using Eq. (2.12).

$Q$ and $R$ are calculated using the variances and covariances for all variables in the state of the system. The covariance matrix associated with a quaternion can be calculated as

$$
\begin{bmatrix}
\sigma_w^2 & \sigma_w \sigma_x & \sigma_w \sigma_y & \sigma_w \sigma_z \\
\sigma_w \sigma_x & \sigma_x^2 & \sigma_x \sigma_y & \sigma_x \sigma_z \\
\sigma_w \sigma_y & \sigma_x \sigma_y & \sigma_y^2 & \sigma_y \sigma_z \\
\sigma_w \sigma_z & \sigma_x \sigma_z & \sigma_y \sigma_z & \sigma_z^2
\end{bmatrix}
$$ (3.28)

where $\sigma$ is the variance of each element of the state.
However, by assuming the elements are independent of each other, we have

$$
\begin{bmatrix}
\sigma_w^2 & 0 & 0 & 0 \\
0 & \sigma_x^2 & 0 & 0 \\
0 & 0 & \sigma_y^2 & 0 \\
0 & 0 & 0 & \sigma_z^2 \\
\end{bmatrix}
$$

(3.29)

The above process is necessary for the measurements obtained, resulting in $R$, and for static IMU data taken, resulting in $Q$. The covariance matrices used in this thesis can be seen in Chapter 4.4.2.

It is also important to initialize the process noise covariance matrix. Because it will be updated at every time step to converge to the most accurate value, the initial value is not very critical. Hence we set

$$
P_0 = \begin{bmatrix}
0.01 & 0 & 0 & 0 \\
0 & 0.01 & 0 & 0 \\
0 & 0 & 0.01 & 0 \\
0 & 0 & 0 & 0.01 \\
\end{bmatrix}
$$

(3.30)

### 3.2.3 System Model Construction

In order to build the state transition matrix $A$ in the system model, the gyroscope measurement values are positioned into the following matrix, $\Omega$. This serves as a matrix expansion of the gyroscope quaternion form described in Eq. (3.2) (for the derivation, see [11]).
The state transition matrix \( A \) can be calculated from the gyroscope measurements oriented in \( \Omega \). This matrix relates the state of the system at the previous time step with the state of the system at the current time step. This matrix is detailed in Chapter 11 of Adjouadi et al.[12] and can be considered as numerical differentiation of the \( \Omega \) matrix. Therefore, in this thesis, it is defined as

\[
A = I_4 + \frac{\Delta t}{2} \Omega = \begin{bmatrix}
1 & -\omega_x \frac{\Delta t}{2} & -\omega_y \frac{\Delta t}{2} & -\omega_z \frac{\Delta t}{2} \\
\omega_x \frac{\Delta t}{2} & 1 & \omega_z \frac{\Delta t}{2} & -\omega_y \frac{\Delta t}{2} \\
\omega_y \frac{\Delta t}{2} & -\omega_z \frac{\Delta t}{2} & 1 & \omega_x \frac{\Delta t}{2} \\
\omega_z \frac{\Delta t}{2} & \omega_y \frac{\Delta t}{2} & -\omega_x \frac{\Delta t}{2} & 1
\end{bmatrix}
\] (3.32)

where \( I_4 \) denotes the 4x4 identity matrix.

The observation matrix, \( H \), is defined as the 4x4 identity matrix.

\[
H = I_4
\] (3.33)
3.3 Complementary Filter

This sensor fusion method is directly adopted and summarized from Antonov et al. [9]. A complimentary filter is used to combine orientation measurements from a gyroscope, and an accelerometer and magnetometer. The filter is designed to reduce noise presented in accelerometer and magnetometer measurements and to reduce drift from gyroscope measurements, typically using a weighted average. In our work, accelerometer and magnetometer measurements update the orientation estimate individually, so two weighted averages, $\alpha_a$ and $\alpha_m$ are used.

3.3.1 Frame Transformation of Accelerometer and Magnetometer Data

The first step in our complimentary filter design is to calculate the quaternion orientation based on the gyroscope data. We use Eq. (3.4) for calculation of $q^\omega_k$.

Once the gyroscope orientation estimate is obtained, we can improve the estimate using the accelerometer data. Similar to Madgwick’s algorithm described in Chapter 3.1, our plan is to compare our sensor measurements with their corresponding measurements in the fixed frame. In the case of accelerometer measurements, we consider

$$E \mathbf{\hat{g}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.34)$$

$$S \mathbf{\hat{a}} = \begin{bmatrix} 0 & a_x & a_y & a_z \end{bmatrix} \quad (3.35)$$
where $E_g$ denotes the gravity vector in the Earth/fixed frame, and $\hat{S}a$ denotes accelerometer measurements in the sensor frame.

The goal here is to convert the sensor readings to the world frame and calculate a rotation quaternion from a tilt angle, to nudge the quaternion estimate away from the gyroscope drift to a more accurate estimation.

We convert the accelerometer readings into the world frame using the gyroscope quaternion calculated previously.

\[
E\hat{a}_k = qω_k \otimes \hat{S}a_k \otimes qω^*_k
\]

(3.36)

### 3.3.2 Correction Quaternion Calculation

Given that the accelerometer data and earth’s gravitational vector are now expressed in the same frame, they should be representative of the same information and therefore close to equal. To improve accuracy of the orientation estimation, we’d like to push these two vectors even closer. To accomplish this, the next step is to calculate the tilt angle, $ϕ_a$ through a dot product.

\[
\cos ϕ_a = E\hat{a}_k \cdot E\hat{g}
\]

(3.37)

The axis about which to tilt the measurements is similarly calculated through a cross product.
Converting the tilt angle and axis from angle-axis representation to quaternion representation gives the tilt quaternion $q^\text{tilt,a}_k$.

$$q^\text{tilt,a}_k = q((1 - \alpha_a)\phi_a; n_a)$$  \hspace{1cm} (3.39)

Here, $\alpha_a$ is the weight function that dictates how much trust we have in the accelerometer based tilt update over the original gyroscope orientation estimate. This tilt quaternion can then update the gyroscope measurement as
\[ q_k = q_k^{\text{tilt},a} \otimes \dot{q}_k^* \]  

(3.40)

This process can then be almost identically replicated for the magnetometer measurements using the Earth frame calculated for the magnetic distortion in Eq. (3.11) where

\[ E \hat{h}_t = \begin{bmatrix} 0 & h_x & h_y & h_z \end{bmatrix} = \dot{q}_k \otimes S \hat{m}_t \otimes \dot{q}_k^* \]  

(3.41)

\[ E \hat{b}_t = \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & 0 & h_z \end{bmatrix} \]  

(3.42)

\[ S \hat{m} = \begin{bmatrix} 0 & m_x & m_y & m_z \end{bmatrix} \]  

(3.43)

Now we can convert the magnetometer measurements to the Earth frame.

\[ E \hat{m}_k = \dot{q}_k \otimes S \hat{m}_k \otimes \dot{q}_k^* \]  

(3.44)

Then we calculate the tilt angle and axis.

\[ \cos \phi_m = E \hat{m}_k \cdot E \hat{b} \]  

(3.45)
\[ n_m = E \hat{m}_k \times E \hat{b} \] (3.46)

We can now use the calculated tilt angle and axis to calculate the tilt quaternion through angle-axis to quaternion conversion.

\[ q_{\text{tilt},m}^k = q((1 - \alpha_m) \phi_m, n_m) \] (3.47)

Finally we can have an accurate orientation estimate using gyroscope, accelerometer and magnetometer measurements.

\[ q_{k}^{\text{comp}} = q_{k}^{\text{tilt},m} \otimes \hat{q}_k \] (3.48)
CHAPTER 4
EXPERIMENTAL RESULT

4.1 Materials

4.1.1 Inertial Measurement Unit

The IMUs chosen for this experiment are 9-axis Bluetooth IMUs manufactured by LP-Research [29]. Specifically, three of their LPMS-B2 model IMUs shown in Fig. 4.1 are used. These each contain a 3-axis accelerometer, a 3-axis gyroscope, a 3-axis magnetometer as well as barometric and temperature data, although the temperature and barometric data is unused. LP-Research additionally created their own GUI for data collection called LPMS-Control. This GUI runs on a PC and works for most of their sensors and is very intuitive to use. Each IMU communicates with a host computer on the LPMS-Control software via a classic bluetooth 2.1 connection or a 4.1 low energy connection. This sensor supports a maximum sampling frequency of 400 Hz, although the data in this experiment is collected at 100 Hz. This sample frequency was selected based on the maximum sampling frequency (frame rate) available for the motion capture system used (100 FPS).

4.1.2 Motion Capture System

In order to make the determination on the ideal sensor fusion method, a reference for accurate joint angles calculated is required. In this thesis, that reference comes in the form
of a motion capture system. This system is composed of eight OptiTrack Flex-3 cameras positioned around a specific area in a laboratory as shown in Fig. 4.2. Each of these cameras has a resolution of $640 \times 480$ pixels and a maximum frame rate of 100 FPS.

Through computer vision analysis, these cameras are all able to isolate a specific color and reflectivity that comes in the form of small spherical gray markers. Using triangulation, a position estimation can be generated for each of these markers with respect to a predefined origin. This work is all done automatically in the Motive 2.0 software which is associated with OptiTrack.

Before positional and rotational data can be determined for each of these markers, some steps are required. First, camera calibration must be done using a wand with three markers at specific collinear positions. This is waved around in the target area until enough data is collected through each of the eight cameras to determine camera distortion and build a 3D capture volume for proper associativity between each camera and individual reflective markers.
After calibration, a global reference frame must be assigned for the motion capture system using a plate with 3 spherical markers on it shown in Fig. 4.3. All position and rotational data generated by the motion capture system is with respect to this assigned origin.

Using the Motive 2.0 software, rigid bodies can be assigned in cases where multiple markers are mounted on a fixed plate. These rigid bodies are assigned a coordinate frame that serves as the local coordinate frame for the rigid body. Having a rigid body like this, can help align the local coordinate frames for the motion capture system with the IMU sensors.

Each of these marker plates fixes four reflective markers for the motion capture system and one inertial measurement unit. This allows an ideal alignment for both local coordinate frames.
4.2 Experimental Setup and Procedure

For this experiment, three marker plates were 3D printed and mounted with four reflective markers asymmetrically and one IMU each at the center as shown in Fig. 4.4. These three plates were fixed to my upper torso, shoulder, and lower arm respectively. From these positions, a joint angle calculation allows us to better understand the motion of our elbow and shoulder joints.

Data was then collected from both the motion capture system and the three IMUs simultaneously. As the data is collected on two different computers, care is taken to begin collection simultaneously. However, an offset error in the time domain can be developed in this situation due to not beginning collection at the exact same time. In scenarios like this, matching data peaks from shifting data along the time domain provides a sufficient correction method.
Data collected reflects two different types of motions. These motions are meant to simulate two environments: reaching up from a neutral posture as if to grab something from a high shelf as shown in Fig. 4.5, and reaching across horizontally as if stretching for an object on the far side of a desk as shown in Fig. 4.6. These motions were selected as typical actions that produce a non-ideal posture.

Figure 4.5: Reaching up dataset pictures from left to right: Initial position from the back, initial position from the side, final position from the back, final position from the side
Care was taken to align the coordinate frames for a proper similarity transformation. Additional error can be introduced if the IMU is tilted slightly on the marker plate or if the data collection was initiated at slightly different times.

### 4.3 Similarity Transformation for the IMU and the Motion Capture System

To properly align the IMU data with the motion capture data, a similarity transformation is necessary. A similarity transformation allows us to transform data represented in one frame to be represented in a different frame.

In our case, we have a set of principal axes for each rigid body marker plate generated in the motion capture system shown in Fig. 4.7 and a corresponding set of principal axes for each IMU shown in Fig. 4.1. While it is important to align these frames so that the three axes from IMUs are each collinear with one of the three axes from the motion capture system, it is difficult to have the x axes lined up, the y axes lined up, and the z axes lined up, so we need to understand exactly how they are oriented with respect to one another.
As you can see from the principal axes shown in Figs. 4.1 and 4.7, by aligning these frames, we can generate a rotation matrix that transforms the IMU data into the motion capture rigid body frames. This rotation matrix can then be converted to a quaternion representation presented in Chapter 2. This allows an ideal format for converting the data between frames.

4.4 Experimental Results of Three Filters

To reduce error as much as possible, post processing is done to determine the parameters for each sensor fusion method. Parameter values are determined from testing different values for each parameter and comparing the root mean square error (RMSE) with the motion capture system data. Parameter analysis is done for each IMU to result in the most accurate data comparison with the motion capture system prior to joint angle calculation.
4.4.1 Madgwick’s Gradient Descent Algorithm

As it is explained in detail in Chapter 3, for Madgwick’s algorithm, a single parameter, $\beta$ sets the convergence rate and represents the error in the gyroscope, which results in what is essentially a trust variable for the cost function.

To determine the best value for $\beta$, values were tested at increments of 0.001 for the range $0.001 < \beta < 0.5$. The RMS error results for reaching-up and desk reach with varying $\beta$ are shown in Figs. 4.8 and 4.9, respectively. Using the values of $\beta$ that minimize error for each sensor in each dataset, optimal results compared to motion capture data are generated for each dataset and plotted in Figs. 4.10 and 4.11.

Based on the error results shown in Figs. 4.8 and 4.9, it can be seen that smaller values of $\beta$ produce the most accurate results. Using optimal parameters found, as listed in Tables 4.1 and 4.2, the errors produced for each IMU as well as joint angles (to be explained in Chapter 4.5) are listed in Tables 4.3 and 4.4, respectively.
Figure 4.9: Desk reach dataset - Madgwick filter error for each IMU while varying $\beta$

### 4.4.2 Kalman Filter

The Kalman filter parameters for error reduction were determined according to the covariance matrix calculations discussed in Chapter 3. These were necessary for the measurement noise covariance matrix, $R$, and the process noise covariance matrix, $Q$. These resulted in the $4 \times 4$ diagonal matrices from the reaching up dataset as

\[
Q_{torso} = diag(0.0005, 0.0005, 0.0018, 0.0037)
\]

\[
Q_{upperarm} = diag(0.0120, 0.0189, 0.0193, 0.1342)
\]

\[
Q_{lowerarm} = diag(0.0082, 0.0687, 0.0957, 0.1459)
\]

\[
R_{torso} = diag(0.0006, 0.0005, 0.0026, 0.0020)
\]

\[
R_{upperarm} = diag(0.0355, 0.0937, 0.0778, 0.2172)
\]
Figure 4.10: Reaching up dataset - Madgwick filter pitch, roll, and yaw for each sensor using optimal $\beta$ values

$$ R_{\text{lowerarm}} = \text{diag}(0.0155, 0.2011, 0.1266, 0.1624) $$

The following diagonal covariance matrix elements were calculated from the desk reach dataset.

$$ Q_{\text{torso}} = \text{diag}(0.0014, 0.0035, 0.0030, 0.0016) $$

$$ Q_{\text{upperarm}} = \text{diag}(0.0036, 0.0138, 0.0079, 0.0208) $$

$$ Q_{\text{lowerarm}} = \text{diag}(0.0227, 0.0301, 0.0487, 0.0084) $$
Figure 4.11: Desk reach dataset - Madgwick filter pitch, roll, and yaw for each sensor using optimal $\beta$ values

\[
R_{\text{torso}} = \text{diag}(0.0008, 0.0030, 0.0049, 0.0032)
\]

\[
R_{\text{upperarm}} = \text{diag}(0.0052, 0.0202, 0.0153, 0.0179)
\]

\[
R_{\text{lowerarm}} = \text{diag}(0.0189, 0.0009, 0.0476, 0.0014)
\]

Using the calculated covariance matrices, optimal results for each IMU and each dataset were plotted in Figs. 4.12 and 4.13.
Figure 4.12: Reaching up dataset - Kalman filter pitch, roll, and yaw for each sensor using optimal covariance matrices

4.4.3 Complementary Filter

Similar to parameter analysis done for Madgwick's algorithm, varying parameters of the adjustable parameters $\alpha_A$ and $\alpha_M$ were determined. In this case, all values for $\alpha_A$ were plotted for all values of $\alpha_M$ to generate a surface plot for the RMS error. These surface graphs can be seen in Figs. 4.14 and 4.15. Both $\alpha_A$ and $\alpha_M$ were tested at intervals of 0.02 for the range $0.02 < \alpha < 1$. Using the values for $\alpha_A$ and $\alpha_M$ that minimized error for each
sensor in each dataset (listed in Tables 4.1 and 4.2), optimal results compared to motion capture data were generated for each dataset and plotted in Figs. 4.16 and 4.17.

Based off of the results shown in Figs. 4.14 and 4.15, it should be noted that error resulting from a change in $\alpha_A$ is overshadowed by the error resulting from changes in $\alpha_M$. This can be attributed to the volatility of the magnetometer measurements. Due to the changing presence of soft and hard iron sources, magnetometer data is trusted less than accelerometer data in higher motion scenarios. This is reflected in the optimal parameters in Tables 4.1 and 4.2. It’s shown that for upper and lower arm IMUs (data undergoing more movement than the torso IMU), optimal $\alpha_M$ parameters are closer to 1, signifying less trust
in the magnetometer data according to Eq. (3.47). It’s also shown that for the torso data (that does not undergo as much movement as the lower and upper arm IMUs), there is more trust placed on the magnetometer data.

4.5 Joint Angles Estimation

Optimal parameters for the madgwick and complimentary filters that minimize error are summarized in Tables 4.1 and 4.2. Once error was reduced as much as possible for each sensor fusion method on each IMU, joint angles were calculated from this data and compared to corresponding data for the motion capture system.

The data used to calculate joint angles in this thesis for each sensor fusion method in each dataset can be visualized in Figs. 4.18 and 4.19.

\[ q_{elbow,k} = q_{lowerarm,k} \otimes q_{upperarm,k}^{-1} \quad (4.1) \]

\[ q_{shoulder,k} = q_{upperarm,k} \otimes q_{torso,k}^{-1} \quad (4.2) \]

Joint angles can be considered as the angle between IMUs at all points in time. The elbow joint angles for Reaching Up and Desk Reach shown in Figs. 4.20 and 4.21, respectively were calculated according to Eq. (4.1) for all points in time using the lower arm orientations and the upper arm orientations respectively. Similarly, the shoulder joint angles for Reaching Up and Desk Reach shown in Figs. 4.22 and 4.23, respectively were calculated according to Eq. (4.2) for all points in time using the upper arm orientations and the torso orientations respectively.
Figure 4.14: Reaching up dataset - Complimentary filter error for each IMU with varied $\alpha$ values
Figure 4.15: Desk reach dataset - Complimentary filter error for each IMU with varied $\alpha$ values
Table 4.1: Optimal parameter values per IMU for the reaching up dataset

<table>
<thead>
<tr>
<th></th>
<th>$\beta$</th>
<th>$\alpha_A$</th>
<th>$\alpha_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torso</td>
<td>0.001</td>
<td>0.8</td>
<td>0.68</td>
</tr>
<tr>
<td>Upper arm</td>
<td>0.001</td>
<td>0.5</td>
<td>0.9</td>
</tr>
<tr>
<td>Lower arm</td>
<td>0.009</td>
<td>0.5</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 4.2: Optimal parameter values per IMU for the desk reach dataset

<table>
<thead>
<tr>
<th></th>
<th>$\beta$</th>
<th>$\alpha_A$</th>
<th>$\alpha_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torso</td>
<td>0.001</td>
<td>0.96</td>
<td>1.0</td>
</tr>
<tr>
<td>Upper arm</td>
<td>0.008</td>
<td>0.5</td>
<td>0.66</td>
</tr>
<tr>
<td>Lower arm</td>
<td>0.001</td>
<td>0.5</td>
<td>0.92</td>
</tr>
</tbody>
</table>
Figure 4.17: Desk reach dataset - Complimentary filter pitch, roll, and yaw for each sensor using optimal $\alpha$ values

Table 4.3: Reaching up dataset - error for each sensor and joint angles for each sensor fusion method

<table>
<thead>
<tr>
<th></th>
<th>Madgwick</th>
<th>Kalman</th>
<th>Complimentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torso</td>
<td>5.22</td>
<td>5.07</td>
<td>5.44</td>
</tr>
<tr>
<td>Upper arm</td>
<td>15.57</td>
<td>16.52</td>
<td>16.56</td>
</tr>
<tr>
<td>Lower arm</td>
<td>8.39</td>
<td>8.65</td>
<td>12.67</td>
</tr>
<tr>
<td>Shoulder Joint Angle</td>
<td>16.09</td>
<td>16.15</td>
<td>16.88</td>
</tr>
<tr>
<td>Elbow Joint Angle</td>
<td>17.39</td>
<td>19.05</td>
<td>20.13</td>
</tr>
</tbody>
</table>
Table 4.4: Desk reach dataset - error for each sensor and joint angles for each sensor fusion method

<table>
<thead>
<tr>
<th>Desk Reach Dataset error (degrees)</th>
<th>Madgwick</th>
<th>Kalman</th>
<th>Complimentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torso</td>
<td>3.59</td>
<td>8.43</td>
<td>3.62</td>
</tr>
<tr>
<td>Upper arm</td>
<td>10.75</td>
<td>14.43</td>
<td>10.17</td>
</tr>
<tr>
<td>Lower arm</td>
<td>4.15</td>
<td>7.23</td>
<td>8.71</td>
</tr>
<tr>
<td>Shoulder Joint Angle</td>
<td>11.41</td>
<td>6.59</td>
<td>10.28</td>
</tr>
<tr>
<td>Elbow Joint Angle</td>
<td>14.07</td>
<td>14.15</td>
<td>13.74</td>
</tr>
</tbody>
</table>
Figure 4.19: Desk reach dataset - All sensor fusion methods for each sensor using optimal parameter values
Figure 4.20: Reaching up dataset - All sensor fusion method elbow joint angles using optimal parameter values

Figure 4.21: Desk reach dataset - All sensor fusion method elbow joint angles using optimal parameter values
Figure 4.22: Reaching up dataset - All sensor fusion method shoulder joint angles using optimal parameter values

Figure 4.23: Desk reach dataset - All sensor fusion method shoulder joint angles using optimal parameter values
CHAPTER 5
DISCUSSIONS AND CONCLUSIONS

In this thesis, three sensor fusion methods (Madgwick's gradient descent algorithm, Kalman filter, and complimentary filter) were derived and compared for use in arm and shoulder joint angle calculation. Three IMUs were used for two representative motions: reaching up, and reaching across a desk. These IMUs were placed on the upper torso, upper arm, and lower arm respectively. Data from these IMUs was taken simultaneously with and compared to an optical motion tracking system to determine the overall accuracy for each sensor fusion method. For each method, optimal parameters were determined per sensor to minimize overall joint angle errors. Data was processed using quaternions as the ideal orientation representation in order to avoid mathematical singularities throughout the process. These quaternions were converted to Euler angles once processed to intuitively visualize the motions that took place. Errors for each method might be due to inconsistent frame alignment in the similarity transformation, or from data misalignment in the time domain due to inconsistent data collection start times. Additional errors might be in the Kalman filter due to the nature of the measurement vector (converting from Euler angles could introduce singularities). While the error for each method is comparable, Madgwick’s gradient descent algorithm performed better than the Kalman and complimentary filters on average. However, with joint angle errors typically placing over 10 degrees in this thesis, more research is necessary before IMUs serve as a sufficient methodology for diagnosing WMSDs. Future work could include exploring other sensor fusion approaches such as Unscented Kalman filters (UKF) or Extended Kalman filters (EKF), or using multiple IMUs per joint to improve overall accuracy.
REFERENCES


APPENDIX

CODE
Contents

• Dissecting Data
• Processing Motion Capture Data
• Implementing a Kalman Filter based on the Book: Intuitive Understanding of Kalman Filtering with Matlab
• Madgwick’s Gradient Descent Algorithm
• Complementary Filter
• Localizing data
• Calculating Joint Angles
• Calculating Euler Angles
• Fixing singularities from quaternion to Euler angle conversion
• Calculating Error

%-------------------------------------------------------------------------------
%  
%  This program takes in an excel data file from an LPMS-B2 IMU from LP-Research.
%  
%  Aaron Freedkin
%  Masters Thesis under Dr. Ji-Chul Ryu
%  Northern Illinois University
%  Dekalb, IL, 2021-2022

%-------------------------------------------------------------------------------
Dissecting Data

clc

clear

%Reading the Excel data spreadsheet generated by the LPMS-B2 IMU

% M = readmatrix("Desk_Reach_IMU_10_6_2022.csv");
% MotionCaptureData = readmatrix("Desk_Reach_MoCap_10_06_2022.csv");
% P=0;

M = readmatrix("Reach_up_Data_IMU_10_6_2022.csv");
MotionCaptureData = readmatrix("Reaching_Up_MoCap_10_06_2022.csv");
P=1;

ID = M(:,1);

count1 = 1;
count2 = 1;
count3 = 1;

for i = 1:length(M(:,1)) %Sorting the IMU output into individual IMUs
    if ID(i) == 1
        M1(count1,:) = M(i,:);
count1 = count1 + 1;

elseif ID(i) == 2
    M2(count2,:) = M(i,:);
    count2 = count2 + 1;
elseif ID(i) == 3
    M3(count3,:) = M(i,:);
    count3 = count3 + 1;
end
end

if P == 1
    M1 = M1(1:1561,:); % Cropping the data in the case of a time domain offset
    M2 = M2(1:1561,:);
    M3 = M3(1:1561,:);
end

% Sorting IMU data
[Ax1,Ay1,Az1,Gx1,Gy1,Gz1,Gx1_rad,Gy1_rad,Gz1_rad,Mx1,My1,Mz1,Ex1,Ey1,...
 ,Ez1,IMU1_qw,IMU1_qx,IMU1_qy,IMU1_qz,t1,dt1,N1] = SortLPMSB2_IMU(M1);
[Ax2,Ay2,Az2,Gx2,Gy2,Gz2,Gx2_rad,Gy2_rad,Gz2_rad,Mx2,My2,Mz2,Ex2,Ey2,...
 ,Ez2,IMU2_qw,IMU2_qx,IMU2_qy,IMU2_qz,t2,dt2,N2] = SortLPMSB2_IMU(M2);
[Ax3,Ay3,Az3,Gx3,Gy3,Gz3,Gx3_rad,Gy3_rad,Gz3_rad,Mx3,My3,Mz3,Ex3,Ey3,...
 ,Ez3,IMU3_qw,IMU3_qx,IMU3_qy,IMU3_qz,t3,dt3,N3] = SortLPMSB2_IMU(M3);
Calculating base conjugates and inverses

\[
\text{qinit1} = [\text{IMU1}_w(200), \text{IMU1}_x(200), \text{IMU1}_y(200), \text{IMU1}_z(200)]';
\]
\[
\text{qinit1} = \text{qinit1}/\text{norm(qinit1)};
\]

\[
\text{qinit2} = [\text{IMU2}_w(200), \text{IMU2}_x(200), \text{IMU2}_y(200), \text{IMU2}_z(200)]';
\]
\[
\text{qinit2} = \text{qinit2}/\text{norm(qinit2)};
\]

\[
\text{qinit1Conj} = [\text{qinit1}(1), -\text{qinit1}(2), -\text{qinit1}(3), -\text{qinit1}(4)];
\]
\[
\text{qinit1Inv} = (\text{qinit1}/(\text{norm(qinit1Conj)}^2))';
\]

\[
\text{qinit2Conj} = [\text{qinit2}(1), -\text{qinit2}(2), -\text{qinit2}(3), -\text{qinit2}(4)];
\]
\[
\text{qinit2Inv} = (\text{qinit2}/(\text{norm(qinit2Conj)}^2))';
\]

\[
\text{qinit3} = [\text{IMU3}_w(200), \text{IMU3}_x(200), \text{IMU3}_y(200), \text{IMU3}_z(200)]';
\]
\[
\text{qinit3} = \text{qinit3}/\text{norm(qinit3)};
\]

\[
\text{qinit3Conj} = [\text{qinit3}(1), -\text{qinit3}(2), -\text{qinit3}(3), -\text{qinit3}(4)];
\]
\[
\text{qinit3Inv} = (\text{qinit3}/(\text{norm(qinit3Conj)}^2))';
\]

\[
\text{IMU1}_q = [\text{IMU1}_w, \text{IMU1}_x, \text{IMU1}_y, \text{IMU1}_z];
\]
\[
\text{IMU2}_q = [\text{IMU2}_w, \text{IMU2}_x, \text{IMU2}_y, \text{IMU2}_z];
\]
\[
\text{IMU3}_q = [\text{IMU3}_w, \text{IMU3}_x, \text{IMU3}_y, \text{IMU3}_z];
\]
Processing Motion Capture Data

RotMat = [0 0 -1; ... % Rotation matrix for similarity transform
-1 0 0; ...
0 1 0];

if P==0
    [MCt, Torso_quat, Lower_arm_quat, Upper_arm_quat] = MoCapProcess(...
        MotionCaptureData, RotMat); % Processing motion capture data
elseif P==1
    [MCt, Torso_quat, Lower_arm_quat, Upper_arm_quat] = MoCapProcess(...
        MotionCaptureData, RotMat);
    [Torso_quat] = DataShift(Torso_quat, 0.33, 0.01);
    [Lower_arm_quat] = DataShift(Lower_arm_quat, 0.33, 0.01);
    [Upper_arm_quat] = DataShift(Upper_arm_quat, 0.33, 0.01);
end

Implementing a Kalman Filter based on the Book: Intuitive Understanding of Kalman Filtering with Matlab

Q1 = CalcQ(IMU1_q); % Calculating the process noise covariance matrix
Q2 = CalcQ(IMU2_q);
Q3 = CalcQ(IMU3_q);
[KalmanQ1,EulerKF1] = Kalman2(Ax1,Ay1,Az1,Gx1_rad,Gy1_rad,Gz1_rad...
   ,Mx1,My1,Mz1,Q1,qinit1Conj',dt1);  \%Computing kalman filter
[KalmanQ2,EulerKF2] = Kalman2(Ax2,Ay2,Az2,Gx2_rad,Gy2_rad,Gz2_rad...
   ,Mx2,My2,Mz2,Q2,qinit2Conj',dt2);
[KalmanQ3,EulerKF3] = Kalman2(Ax3,Ay3,Az3,Gx3_rad,Gy3_rad,Gz3_rad...
   ,Mx3,My3,Mz3,Q3,qinit3Conj',dt3);

\textbf{Madgwick’s Gradient Descent Algorithm}

if P == 0

Beta1 = 0.001;  \%Setting values for Beta depending on Beta
Beta2 = 0.008;
Beta3 = 0.001;

elseif P == 1

Beta1 = 0.001;
Beta2 = 0.001;
Beta3 = 0.009;
end

\%Processing Madgwick’s algorithm

[MadgQ1, EulerM1] = Madgwick(Ax1,Ay1,Az1,Gx1_rad,Gy1_rad,Gz1_rad...
   ,Mx1,My1,Mz1,Beta1,qinit1Conj,dt1);
[MadgQ2, EulerM2] = Madgwick(Ax2, Ay2, Az2, Gx2_rad, Gy2_rad, Gz2_rad...
 , Mx2, My2, Mz2, Beta2, qinit2Conj, dt2);

[MadgQ3, EulerM3] = Madgwick(Ax3, Ay3, Az3, Gx3_rad, Gy3_rad, Gz3_rad...
 , Mx3, My3, Mz3, Beta3, qinit3Conj, dt3);

### Complementary Filter

if P==0
    Alpha1 = 0.96; % Setting values of alphaA and alphaM per dataset
    AlphaM1 = 1;

    Alpha2 = 0.5;
    AlphaM2 = 0.66;

    Alpha3 = 0.5;
    AlphaM3 = 0.92;

elseif P==1
    Alpha1 = 0.8;
    AlphaM1 = 0.68;

    Alpha2 = 0.5;
    AlphaM2 = 0.9;

    Alpha3 = 0.5;
AlphaM3 = 1;
end

% Processing Complimentary filter
[CompQ1,EulerCF1] = Complimentary2(Ax1,Ay1,Az1,Gx1_rad,Gy1_rad...
    ,Gz1_rad,Mx1,My1,Mz1,Alpha1,AlphaM1,qinit1Conj,dt1);
[CompQ2,EulerCF2] = Complimentary2(Ax2,Ay2,Az2,Gx2_rad,Gy2_rad...
    ,Gz2_rad,Mx2,My2,Mz2,Alpha2,AlphaM2,qinit2Conj,dt2);
[CompQ3,EulerCF3] = Complimentary2(Ax3,Ay3,Az3,Gx3_rad,Gy3_rad...
    ,Gz3_rad,Mx3,My3,Mz3,Alpha3,AlphaM3,qinit3Conj,dt3);

## Localizing data

for i = 1:1:length(IMU1_q(:,1))
    IMU1_q(i,:) = quatmultiply(IMU1_q(i,:),q1initInv);
end
for i = 1:1:length(IMU2_q(:,1))
    IMU2_q(i,:) = quatmultiply(IMU2_q(i,:),q2initInv);
end
for i = 1:1:length(IMU3_q(:,1))
    IMU3_q(i,:) = quatmultiply(IMU3_q(i,:),q3initInv);
end

for i = 1:1:length(KalmanQ1(1,:))
    KalmanQ1(:,i) = quatmultiply(KalmanQ1(:,i)',qinit1Inv);
end
for i = 1:1:length(KalmanQ2(1,:))
    KalmanQ2(:,i) = quatmultiply(KalmanQ2(:,i)',qinit2Inv);
end
for i = 1:1:length(KalmanQ3(1,:))
    KalmanQ3(:,i) = quatmultiply(KalmanQ3(:,i)',qinit3Inv);
end

for i = 1:1:length(MadgQ1(:,1))
    MadgQ1(i,:) = quatmultiply(MadgQ1(i,:),qinit1Inv);
end
for i = 1:1:length(MadgQ2(:,1))
    MadgQ2(i,:) = quatmultiply(MadgQ2(i,:),qinit2Inv);
end
for i = 1:1:length(MadgQ3(:,1))
    MadgQ3(i,:) = quatmultiply(MadgQ3(i,:),qinit3Inv);
end

for i = 1:1:length(CompQ1(:,1))
    CompQ1(i,:) = quatmultiply(CompQ1(i,:),qinit1Inv);
end
for i = 1:1:length(CompQ2(:,1))
    CompQ2(i,:) = quatmultiply(CompQ2(i,:),qinit2Inv);
end
for i = 1:1:length(CompQ3(:,1))
    CompQ3(i,:) = quatmultiply(CompQ3(i,:),qinit3Inv);
end
Calculating Joint Angles

\[
\begin{align*}
[IMU\_JA1, \text{Euler\_IMUJA1}] &= \text{JointAngle}(IMU\_1\_q, IMU\_2\_q); \\
[IMU\_JA2, \text{Euler\_IMUJA2}] &= \text{JointAngle}(IMU\_2\_q, IMU\_3\_q); \\

[KalmanJA1, KalmanJA1\_Euler] &= \text{JointAngle}(KalmanQ1', KalmanQ2'); \\
[KalmanJA2, KalmanJA2\_Euler] &= \text{JointAngle}(KalmanQ2', KalmanQ3'); \\

[MadgJA1, MadgJA1\_Euler] &= \text{JointAngle}(MadgQ1, MadgQ2); \\
[MadgJA2, MadgJA2\_Euler] &= \text{JointAngle}(MadgQ2, MadgQ3); \\

[CompJA1, CompJA1\_Euler] &= \text{JointAngle}(CompQ1, CompQ2); \\
[CompJA2, CompJA2\_Euler] &= \text{JointAngle}(CompQ2, CompQ3); \\

[MoCapJA1, MoCapJA1\_Euler] &= \text{JointAngle}(Torso\_quat, Upper\_arm\_quat); \\
[MoCapJA2, MoCapJA2\_Euler] &= \text{JointAngle}(Upper\_arm\_quat, Lower\_arm\_quat);
\end{align*}
\]

Calculating Euler Angles

\[
\begin{align*}
\text{for } i &= 1:1:length(IMU\_1\_q(:,1)) \\
\text{EulerIMU1}(i,:) &= \text{rad2deg}(~\text{quatern2euler}(IMU\_1\_q(i,:))); \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{for } i &= 1:1:length(IMU\_2\_q(:,1)) \\
\text{EulerIMU2}(i,:) &= \text{rad2deg}(~\text{quatern2euler}(IMU\_2\_q(i,:))); \\
\text{end}
\end{align*}
\]
for i = 1:1:length(IMU3_q(:,1))
    EulerIMU3(i,:) = rad2deg(quatern2euler(IMU3_q(i,:)));
end

for i = 1:1:length(KalmanQ1(1,:))
    EulerKF1(i,:) = rad2deg(quatern2euler(KalmanQ1(:,i)'));
end
for i = 1:1:length(KalmanQ2(1,:))
    EulerKF2(i,:) = rad2deg(quatern2euler(KalmanQ2(:,i)'));
end
for i = 1:1:length(KalmanQ3(1,:))
    EulerKF3(i,:) = rad2deg(quatern2euler(KalmanQ3(:,i)'));
end
for i = 1:1:length(KalmanJA1(:,1))
    EulerKFJA1(i,:) = rad2deg(quatern2euler(KalmanJA1(i,:)));
end
for i = 1:1:length(KalmanJA2(:,1))
    EulerKFJA2(i,:) = rad2deg(quatern2euler(KalmanJA2(i,:)));
end

for i = 1:1:length(MadgQ1(:,1))
    EulerM1(i,:) = rad2deg(quatern2euler(MadgQ1(i,:)));
end
for i = 1:1:length(MadgQ2(:,1))
    EulerM2(i,:) = rad2deg(quatern2euler(MadgQ2(i,:)));  
end
for i = 1:1:length(MadgQ3(:,1))
    EulerM3(i,:) = rad2deg(quatern2euler(MadgQ3(i,:)));  
end
for i = 1:1:length(MadgJA1(:,1))
    EulerMJA1(i,:) = rad2deg(quatern2euler(MadgJA1(i,:)));  
end
for i = 1:1:length(MadgJA2(:,1))
    EulerMJA2(i,:) = rad2deg(quatern2euler(MadgJA2(i,:)));  
end
for i = 1:1:length(CompQ1(:,1))
    EulerCF1(i,:) = rad2deg(quatern2euler(CompQ1(i,:)));  
end
for i = 1:1:length(CompQ2(:,1))
    EulerCF2(i,:) = rad2deg(quatern2euler(CompQ2(i,:)));  
end
for i = 1:1:length(CompQ3(:,1))
    EulerCF3(i,:) = rad2deg(quatern2euler(CompQ3(i,:)));  
end
for i = 1:1:length(CompJA1(:,1))
EulerCFJA1(i,:) = rad2deg(quat2euler(CompJA1(i,:)));  
end
for i = 1:length(CompJA2(:,1))
    EulerCFJA2(i,:) = rad2deg(quat2euler(CompJA2(i,:)));  
end

for i = 1:length(Lower_arm_quat(:,1))
    EulerMoCap3(i,:) = rad2deg(quat2euler(Lower_arm_quat(i,:)));  
end
for i = 1:length(Upper_arm_quat(:,1))
    EulerMoCap2(i,:) = rad2deg(quat2euler(Upper_arm_quat(i,:)));  
end
for i = 1:length(Torso_quat(:,1))
    EulerMoCap1(i,:) = rad2deg(quat2euler(Torso_quat(i,:)));  
end

for i = 1:length(MoCapJA1(:,1))
    EulerMoCapJA1(i,:) = rad2deg(quat2euler(MoCapJA1(i,:)));  
end
for i = 1:length(MoCapJA2(:,1))
    EulerMoCapJA2(i,:) = rad2deg(quat2euler(MoCapJA2(i,:)));  
end
Fixing singularities from quaternion to Euler angle conversion

\[
\begin{align*}
\text{EulerKF3} &= \text{EulerFix} ( \text{EulerKF3} ) \\
\text{EulerM3} &= \text{EulerFix} ( \text{EulerM3} ) \\
\text{EulerIMU3} &= \text{EulerFix} ( \text{EulerIMU3} ) \\
\text{EulerMoCap3} &= \text{EulerFix} ( \text{EulerMoCap3} ) \\
\text{EulerCF3} &= \text{EulerFix} ( \text{EulerCF3} ) \\
\end{align*}
\]

Calculating Error

\[
\begin{align*}
[\text{Kalman1Error}] &= \text{RMSEuler} ( \text{EulerMoCap1(1:N1,:)}, \text{EulerKF1} ) \\
[\text{Madg1Error}] &= \text{RMSEuler} ( \text{EulerMoCap1(1:N1,:)}, \text{EulerM1} ) \\
[\text{Comp1Error}] &= \text{RMSEuler} ( \text{EulerMoCap1(1:N1,:)}, \text{EulerCF1} ) \\
[\text{Kalman2Error}] &= \text{RMSEuler} ( \text{EulerMoCap2(1:N1,:)}, \text{EulerKF2} ) \\
[\text{Madg2Error}] &= \text{RMSEuler} ( \text{EulerMoCap2(1:N1,:)}, \text{EulerM2} ) \\
[\text{Comp2Error}] &= \text{RMSEuler} ( \text{EulerMoCap2(1:N1,:)}, \text{EulerCF2} ) \\
[\text{Kalman3Error}] &= \text{RMSEuler} ( \text{EulerMoCap3(1:N1,:)}, \text{EulerKF3} ) \\
[\text{Madg3Error}] &= \text{RMSEuler} ( \text{EulerMoCap3(1:N1,:)}, \text{EulerM3} ) \\
[\text{Comp3Error}] &= \text{RMSEuler} ( \text{EulerMoCap3(1:N1,:)}, \text{EulerCF3} ) \\
[\text{KalmanJA1Error}] &= \text{RMSEuler} ( \text{EulerMoCapJA1(1:N1,:)}, \text{EulerKFJA1} ) \\
[\text{MadgJA1Error}] &= \text{RMSEuler} ( \text{EulerMoCapJA1(1:N1,:)}, \text{EulerMJA1} ) \\
[\text{CompJA1Error}] &= \text{RMSEuler} ( \text{EulerMoCapJA1(1:N1,:)}, \text{EulerCFJA1} ) \\
\end{align*}
\]
[KalmanJA2Error] = RMSEuler(EulerMoCapJA2(1:N1,:), EulerKFJA2);

[MadgJA2Error] = RMSEuler(EulerMoCapJA2(1:N1,:), EulerMJA2);

[CompJA2Error] = RMSEuler(EulerMoCapJA2(1:N1,:), EulerCFJA2);