2019

Tracking Using Fusion of Multiple Inertial Measurement Units

Jacob Reed McCall
University of Mississippi

Follow this and additional works at: https://egrove.olemiss.edu/hon_thesis
Part of the Electrical and Computer Engineering Commons

Recommended Citation
McCall, Jacob Reed, "Tracking Using Fusion of Multiple Inertial Measurement Units" (2019). Honors Theses. 1218.
https://egrove.olemiss.edu/hon_thesis/1218
DEDICATION

To my friend Michael. May you always have minutes on your phone.
ACKNOWLEDGEMENTS

First, I would like to thank Dr. John N. Daigle for his continual selflessness, accessibility, and dedication. He has met every question, every request, every favor, and every complaint by freely giving his time to help me with anything I needed. He has taught me much about what it means to conduct valuable research, what it means to be a good engineer, and what it means to have great character. He is truly an outstanding faculty member and has played a key role in my success at The University of Mississippi. I would like to thank Dr. Goggans for the use of his lab and his tools for configuring each IMU device. I would also like to thank my father Mark McCall for helping me build the RC truck IMU system.

I would like to thank both my parents Mark and Jackie McCall. They have supported me throughout my academic career, and they have always encouraged me to be the best version of myself. They have taught me discipline and determination. I would like to thank my grandfather Larry Silvey, who has taught me the value of hard-work. I would also like to thank my uncle Larry Dean Silvey because he is the reason I am an engineering student today. I’d like to thank my siblings and the rest of my family for their love and kindness toward me.

Lastly, I’d like to thank Jesus for his faithfulness to me. He has been constant through every success, every failure, every award, and every mistake. It is in serving Him with my life that I find my true purpose.
ABSTRACT

The objective of this thesis is to determine the effectiveness of fusing data from multiple inertial measurement units (IMU) to reduce bias and noise with an overall goal of achieving accurate tracking, which is the process of locating a body as it moves in a fixed environment. Every sensor is subject to noise, and each sensor has its own unique set of biases. The thesis presents a systematic overview of the sensors used in this research, which feature linear acceleration, angular acceleration, and a directional sensor, to form an inertial navigation system (INS). Sources of noise and bias that affect utility of the IMUs as well as data processing algorithms used for estimation and filtering are also presented. Related work in the topic area is summarized. Finally, seven experiments, which evaluate the accuracy of the acceleration measurements and overall displacement from both the fusion method and the raw single-sensor method, are presented. The accuracy of the acceleration measurements is evaluated by comparing the sensor measurements to the known theoretical acceleration using common statistical metrics. Tracking accuracy is evaluated by overall displacement accuracy and path displacement accuracy. It is found that multiple sensor fusion is not always capable of estimating the overall displacement more accurately than a single sensor. Additionally, fusion increases the signal-to-noise ratio of the accelerometer data. However, our results indicate that neither the fusion nor the single-sensor method are capable of accurately estimating the displacement path.
Contents

1 Introduction 1
   1.1 Localization, Tracking and Navigation Systems . . . . . . . . . . . . . 1
   1.2 Summary of Thesis Content . . . . . . . . . . . . . . . . . . . . . 2

2 Inertial Measurement Units 4
   2.1 Accelerometers . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
   2.2 Gyroscopes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
   2.3 Magnetometers . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
   2.4 Limitations and Sources of Error . . . . . . . . . . . . . . . . . . . 10
      2.4.1 Input Sensitivity . . . . . . . . . . . . . . . . . . . . . . . . 11
      2.4.2 Bias . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
      2.4.3 Noise and Interference . . . . . . . . . . . . . . . . . . . . . 13

3 Data Processing Algorithms 14
   3.1 Extended Kalman Filter . . . . . . . . . . . . . . . . . . . . . . . . 14
   3.2 Window Peak Limiter . . . . . . . . . . . . . . . . . . . . . . . . . 16
   3.3 Timing Alignment . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
   3.4 Frame of Reference Alignment . . . . . . . . . . . . . . . . . . . . 20
   3.5 Numerical Integration . . . . . . . . . . . . . . . . . . . . . . . . . 22

4 Related Work 24
4.1 Data Filtering .......................................................... 24
4.2 Methods of Tracking ............................................... 26
4.3 Summary of Related Work ....................................... 28

5  Experimental Setup ................................................. 30
5.1 Equipment ............................................................ 30
5.2 Data Collection and Processing Method ......................... 32
  5.2.1 Remotely Accessing the Sensors .............................. 32
  5.2.2 Collection and Pre-processing ................................ 33
  5.2.3 Sensor Data Centralization ................................. 35
  5.2.4 Post-processing ................................................. 35

6  Determining the Effectiveness of Multiple IMU Fusion .......... 38
6.1 Indoor Tests ......................................................... 38
  6.1.1 Test 1: Uncalibrated Static ................................. 41
  6.1.2 Test 2: Calibrated Static ..................................... 43
  6.1.3 Test 3: Calibrated Line Translation ....................... 45
6.2 Outdoor Tests ....................................................... 48
  6.2.1 Test 1: Static .................................................. 48
  6.2.2 Test 2: Power Drill Pull ..................................... 50
  6.2.3 Test 3: Hand Pull North ..................................... 53
  6.2.4 Test 4: Hand Pull South ..................................... 55
6.3 Signal-to-Noise Ratio Analysis ................................ 57
  6.3.1 Timing Alignment for SNR Analysis ....................... 62
6.4 Discussion of Results ............................................ 65

7  Conclusion ................................................................ 69
# List of Figures

2.1 Mass-Spring-Damper Model of an accelerometer .................. 5  
2.2 Differential capacitive bridge ..................................... 6  
2.3 MEMS accelerometer with the tooth configuration [3] ............ 7  
2.4 Dynamical model of a MEMS gyroscope [6] .......................... 8  
2.5 Two spring MEMS magnetometer configuration [8] ............... 10  
2.6 X-axis accelerometer in the presence of vibrational noise ......... 11  
3.1 EKF state prediction and measurement update equations [10] ...... 15  
3.2 Mock sensor data. The black plot represents data recorded by the  
    initial whereas the red represents data recorded by the subsequent. 18  
3.3 Euler Angles [13] .................................................. 21  
4.1 Comparison of raw and filtered yaw angle measurements from a smart-  
4.2 Robot Square Test using Double Integration Method [19] .......... 27  
5.1 Six IMU Raspberry Pi setups mounted on the RC truck ............ 31  
5.2 RTIMULib2 library output ......................................... 32  
6.1 Acceleration outputs from each device ............................. 40  
6.2 Test 1: uncalibrated static placement. Acceleration, velocity, and dis-  
    placement ............................................................ 42
6.3 Test 2: calibrated static placement. Acceleration, velocity, and displacement ........................................ 44
6.4 Test 3: movement in a line. Acceleration, velocity, and displacement . 47
6.5 Test 1: static placement. Acceleration, velocity, and displacement . 49
6.6 Test 2: power drill pull. Acceleration, velocity, and displacement . . 52
6.7 Test 3: hand pull in the north direction. Acceleration, velocity, and displacement ........................................ 54
6.8 Test 4: hand pull in the south direction. Acceleration, velocity, and displacement ........................................ 56
6.9 Misaligned acceleration signals of imudev2 and imudev3 .......... 63
6.10 Cross correlation of imudev2 and imudev3 ......................... 63
6.11 Aligned cross correlation of imudev2 and imudev3 ................. 64
6.12 Aligned acceleration signals of imudev2 and imudev3 ............ 65
List of Tables

2.1 Sensor Biases .......................................................... 12

6.1 Noise, signal plus noise, and signal energies and SNR of each sensor . 58

6.2 Noise, signal plus noise, and signal energies and SNR of various averages of sensor acceleration output ........................................ 59

6.3 Noise energies for all possible two-sensor combinations ............... 60

6.4 Signal plus noise energies for all possible two-sensor combinations .. 60

6.5 Signal energies for all possible two-sensor combinations ............. 60

6.6 SNR for all possible two-sensor combinations .......................... 61

6.7 Mean and standard deviation comparison between the fused method and the single-sensor method for indoor Test 1 and Test 2 ....... 66
Chapter 1

Introduction

The objective of this thesis is to quantify the effect of utilizing multiple inertial measurement units (IMU) to track a body using dead-reckoning. Most of the research in this area focuses on using only one IMU, which contains a gyroscope, accelerometer, magnetometer, and sometimes a barometer. Most studies in the topic area recognize the erratic nature of IMU sensors, which are subject to noise and bias. The motivation for using more than one IMU sensor is to reduce sensor bias, which can play a role in the erratic drifting of the device’s estimated position over time.

1.1 Localization, Tracking and Navigation Systems

Localization is defined as the “process of determining the position of an object in space,” whereas tracking refers to the process of localizing over time [1]. This is not a novel concept. Localization and navigation have been historically based on stargazing and dead-reckoning. Localization is the objective of several technologies today, including sonar, GPS, and optical-based systems, such as infrared systems.

As opposed to the localization technologies mentioned previously, the use of IMUs for localization would be referred to as an “unassisted” means of tracking that relies on locally generated information [1]. In other words, there are no outside points
or landmarks the device can use to estimate its position. Instead, IMUs are used to provide the data needed to perform dead-reckoning, which is the process of estimating position based on a known starting point, the direction of movement, and the velocity of the body.

Because IMUs rely on inertial data, a navigation system that uses IMUs is often referred to as an inertial navigation system (INS). The motivation for developing a reliable INS is to eliminate reliance on infrastructure. Another advantage of an INS is its potential to track more accurately on a smaller scale than GPS and where GPS is not available, such as inside buildings. One example is a scenario in which a firefighter is navigating inside a burning building whose power has been shut off and where smoke causes occlusion. In this case, common optical line-of-sight and signal time-of-arrival methods of tracking are rendered useless.

1.2 Summary of Thesis Content

This thesis has identified the problem and presented the motivation for developing an inertial-based navigation system. Chapter 2 outlines the mechanics of micro-electrical-mechanical systems (MEMS) technology, namely the basic operation of accelerometers, gyroscopes, and magnetometers. Basic structure and some dynamical systems modeling are shown for each of these sensors. Then in Chapter 3, the algorithms used for processing data on this device will be discussed. These include algorithms, such as Kalman filters, quaternion transformations, window peak limiters, sensor fusion timing alignment, and numerical integration. Related work is discussed in Chapter 4. Specifically, methods for filtering sensor data and other techniques which use IMUs for localizing are presented. The details of the experimental setup and data collection are discussed in Chapter 5. Several tests, along with a signal-to-noise ratio analysis, are presented and discussed in Chapter 6. Lastly, the
conclusions are drawn in Chapter 7. The results of the research and avenues for possible further research are summarized.
Chapter 2

Inertial Measurement Units

This chapter begins with system descriptions of MEMS accelerometers, gyroscopes, and magnetometers and concludes with a discussion of their limitations and sources of error. It will be seen that the system structure of the sensors used in IMUs have undesirable characteristics, which may introduce noise and bias to the sensor output.

2.1 Accelerometers

The most common type of MEMS accelerometer, including the one used in this research, uses capacitive sensing as an indirect means of measuring acceleration along one axis [2]. The accelerometer model can be approximated as a mass-spring-damper system. Figure 2.1 illustrates the approximated model of the system. The proof mass \( m \) is connected to a stationary frame by a spring with a spring constant \( k \). In reality, the damping coefficient, \( b \), characterizes the viscous effects of gases confined in the MEMS accelerometer packaging [2]. The dynamical equation for the system can be written as follows:

\[
m\ddot{x} + b\dot{x} + k = ma_{in}
\]  

(2.1)
In (2.1), \( a_{in} \) represents the input as the acceleration of the accelerometer as a whole, not just the acceleration of the mass \( m \). The output, represented by \( x \), is the displacement of the accelerometer along its axis, which is defined by the manufacturer and labeled on the IMU. Taking the Laplace transform of the dynamical equation, the system can be represented by a transfer function as follows:

\[
\frac{X(s)}{A_{in}(s)} = \frac{1}{s^2 + \frac{b}{m}s + \frac{k}{m}}
\]  

(2.2)

From this, the natural oscillation frequency \( \omega_n \) and damping ratio \( \zeta \) can be determined as

\[
\omega_n = \sqrt{\frac{k}{m}}
\]  

(2.3)

\[
\zeta = \frac{b}{\sqrt{2m}}
\]  

(2.4)

Accelerometers are typically manufactured so that the damping ratio is either underdamped (\( \zeta < 1 \)) or critically damped (\( \zeta = 1 \)) [3, 2]. Making the damping ratio over damped can increase stability of the proof mass; however, it can intensify Brownian noise, which refers to noise caused by the random movement of microscopic particles in a fluid [3]. Oftentimes, underdamping is also avoided because it results in large amplitudes when excited near the resonance frequency. When excited at frequencies much lower than the resonance frequency, the sensitivity of the accelerometer ceases
to be a function of the excitation frequency [2]. This range where the sensitivity is independent of the excitation frequency is referred to as the accelerometer passband [2]. It is desirable for the accelerometer to operate in its passband.

To measure the displacement of the proof mass, a variable capacitor is placed on each side of the mass as shown in Figure 2.2. This configuration forms a differential capacitive bridge, which has the advantage of producing a more linearized output [2]. The term $d_0$ represents the distance halfway between the two variable capacitors. Terms $C_1$ and $C_2$ represent the capacitors formed between the proof mass and each electrode.

The model analyzed briefly in this section is limited to the parallel plate capacitor configuration. However, there are other electrostatic sensing accelerometer configurations, such as the tooth model depicted in Figure 2.3. The sensors used in this research rely on the capacitive sensing principle, but its product specification document is unspecific about its configuration [4]. The discussion of the accelerometer mechanics will not include the analysis of other types as it is outside the scope of this thesis. For an in-depth analysis and more information on other configurations, see
2.2 Gyroscopes

MEMS gyroscopes measure the angular rate of rotation of a moving body. There are multiple ways to realize a MEMS gyroscope, but the most common relies on the Coriolis effect, which will be described shortly. This particular implementation is similar in structure to a MEMS accelerometer. A MEMS gyroscope can be modeled as a proof mass connected to a rigid frame by four springs and dampers connected on four sides of the proof mass, resulting in two degrees of freedom [6]. One axis is considered the drive while the other is the sense. The drive axis is driven to resonance. When the rigid frame is rotated, the Coriolis effect causes acceleration in the sense axis [6]. The Coriolis effect is a phenomenon in which a body moving in a rotating system experiences a force perpendicular to the direction of motion and the axis of rotation. By sensing the oscillation in the sense axis, the angular velocity of the rigid body can be determined [6]. Figure 2.4 illustrates the dynamical model of
the gyroscope.

\[ \ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2x + \omega_{xy}y - 2\Omega\dot{y} = \frac{k}{d}u_d \] (2.5)

\[ \ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2y + \omega_{xy}x + 2\Omega\dot{x} = \frac{1}{m}N(t) \] (2.6)

There are a few key terms to note in the above equations. The terms $2\Omega\dot{y}$ and $2\Omega\dot{x}$ model the Coriolis accelerations, where $\Omega$ is the rotation of the rigid frame. The terms $\omega_{xy}y$ and $\omega_{xy}x$ are the quadrature errors due to spring couplings in the x and y axes [6]. See [6, 7] for more information about MEMS gyroscopes.
2.3 Magnetometers

The MEMS magnetometer measures the magnetic field with one degree of freedom. Like the gyroscope and accelerometer, there are multiple ways of implementing a magnetometer. One way is to take advantage of the Lorentz force which acts in the direction orthogonal to the current flow through a wire and the direction of the magnetic field. The idea is to flow an AC current through a spring or torsion bar, which will flex in the presence of a magnetic field, as a result of the Lorentz force [8]. The displacement of the spring or bar can be sensed by placing differential capacitors in such a way that the capacitance is changed as the spring or bar flexes. Figure 2.5 illustrates one configuration, which utilizes two springs and several capacitors.

As seen in Figure 2.5, the springs are connected by a shuttle, which has several teeth. Each of the inner teeth passes between two electrodes, forming multiple differential capacitors. Langfelder et al. [8] derive the equation for the displacement of the rigid shuttle and frame with respect to the current passing through the spring. The equation is shown in (2.7) [8].

![Figure 2.5: Two spring MEMS magnetometer configuration [8]](image-url)
\[ x(t) = \frac{I(t)B \cdot L \cdot Q}{2 \cdot k} \]  \quad (2.7)

\( I(t) \) represents the current passing through the spring as a function of time. \( B \) represents the magnitude of the magnetic field vector in the direction orthogonal to the Lorentz force and the current \( I(t) \). \( L \) is the length of the spring. \( Q \) is some quality factor which amplifies the displacement, and \( k \) is the stiffness of the device [8]. By using capacitance sensing to determine \( x(t) \), the magnitude of the magnetic field acting in the direction illustrated in Figure 2.5 can be determined. See [8] for more information on MEMS magnetometers.

### 2.4 Limitations and Sources of Error

This section details several sources of error that are important to keep in mind when working with IMUs. Specifically discussed are input sensitivity, bias, and noise.

#### 2.4.1 Input Sensitivity

As discussed previously, an accelerometer can be designed to work at different ranges of frequency excitation. The challenge is that the device must not filter out frequencies that are too low or too high. As a result, accelerometers are often subject to vibrational noise. The effects of vibration can make it more difficult to distinguish the true measurement of the body in motion from the noise caused by the vibration [9]. Figure 2.6 shows the effect of vibrational noise in an acceleration plot. The vibration causes the consistent large peaks, ranging up to \( 6 \text{m/s}^2 \) and down to around \(-5 \text{m/s}^2\). Since the actual translational acceleration of the sensor is nearly zero, it can be seen that a low input sensitivity will yield erroneous results.
There are several sources and types of bias in a MEMS sensor. The first is static bias, which describes the constant bias of the sensor [9]. Each sensor will have its own static bias unique to itself. However, this bias can change over time due to aging of the device components. Additionally, the bias can change each time the sensor is powered up due to the initialization of the signal processor in the IMU. This is referred to as “turn-on to turn-on bias” [9]. The bias can also change during use, which is called “in-run bias.” This is caused by fluctuations in temperature, pressure, and mechanical strain on the system [9]. Because MEMS sensors, such as capacitive accelerometers and gyroscopes, measure data indirectly, the outputs of the sensors are scaled. A scaling bias, therefore, occurs when the scaling factor is not perfectly accurate. It is apparent that IMU sensors are riddled with bias. Therefore, frequent calibration is necessary to obtain suitable readings. The biases outlined here are summarized in Table 2.1 below.
### Table 2.1: Sensor Biases

<table>
<thead>
<tr>
<th>Type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>Bias unique to the sensor due to manufacturing or materials characteristics</td>
</tr>
<tr>
<td>Turn-on to Turn-on</td>
<td>Changes to the static bias which occur due to signal processing initial conditions</td>
</tr>
<tr>
<td>In-run</td>
<td>Alterations to the static bias due to fluctuations in the environment or due to mechanical strain</td>
</tr>
<tr>
<td>Scaling</td>
<td>Bias due to an incorrect scaling factor between the measured data and the scaled output</td>
</tr>
</tbody>
</table>

#### 2.4.3 Noise and Interference

MEMS technology is subject to random noise such as, Brownian noise, mechanical-thermal noise, and various other types. Some of this noise, such as Brownian, is directly correlated to the design of the system. Interference, such as EMI, can vary greatly depending on the surrounding environment of the sensor. The effect of fluctuating magnetic environments was observed during experimentation. It was found that the system’s estimated direction of magnetic north could be altered by placing some mass of metal near the magnetometer. Knowledge of these sources of error is important so the experimentation can be designed in such a way as to eliminate controllable interferences.
Chapter 3

Data Processing Algorithms

There are five main data processing algorithms used in this experiment, all of which are key to regulating the output of the IMU sensors. These include an extended Kalman filter, a window peak limiter, timing alignment, frame of reference alignment, and numerical integration. Each of these algorithms will be discussed in the following sections.

3.1 Extended Kalman Filter

A Kalman filter is a recursive method of estimating the state of a linear discrete-time controlled process [10]. A Kalman filter iteratively estimates state by first predicting the state of a process, then measuring the state, and finally using the measurement to correct the prediction. The motivation for using a Kalman filter in navigation is its ability to fuse sensor data to improve accuracy of the estimate of the state. An extended Kalman filter (EKF) is only different from a Kalman filter in the fact that it is used for nonlinear systems. An EKF works by linearizing around the estimation using partial derivatives [10]. Figure 3.1 below illustrates the equations used to predict the next state of the process and correct with the measurement.

The notation $\hat{x}_k$ describes the a priori state, which represents the prediction of
Figure 3.1: EKF state prediction and measurement update equations [10]

the state prior to its correction using the measurement, which is represented by $z_k$. $\hat{x}_k$ represents the a posteriori state, which is the value of the state after measurement correction. The estimation and measurement error covariances, $P_k$ and $R_k$, respectively, are useful for determining the Kalman gain. The Kalman gain is computed in such a way that it weights the residual $[z_k - h(\hat{x}_k)]$ based upon the magnitude of the error covariances, where the residual represents the difference between the actual measurement and the predicted measurement of the state. If $R_k$ goes to zero, then the Kalman gain becomes $H_k^{-1}$, which weights the residual more heavily. Conversely, if the a priori error covariance $P_k^-$ becomes zero, the Kalman gain goes to zero, which means the a priori estimate is given all the weight while the residual is given none [10].

Kalman filters are capable of fusing data together from multiple sensors. This capability is available in the library provided in [11] to fuse accelerometer, gyroscope, and magnetometer data together to estimate pose. This can be implemented in various ways. One way is to use the Kalman filter to estimate the state vector by assigning
weights based on the noise covariance matrices of each sensor [12]. This is similar to the idea mentioned in the previous description of Kalman filters, where residuals are weighted based on the error covariance. Richard Barnett takes a different approach when fusing data in the library [11]. The method used in the RTIMULib2 library calculates a quaternion, the measured pose, using the accelerometer and magnetometer data. The pose is calculated by performing the quaternion rotational transformation described in Section 3.4 of this thesis. The measured pose is then combined with the measured gyroscope data to make a state prediction using equations similar to those shown in in Figure 3.1.

3.2 Window Peak Limiter

The idea of a peak limiter is to filter out spikes in the dataset. The goal is to obtain a more accurate representation of the measured data by regulating noisy data points. The filter moves along the dataset, taking N points at a time and attenuating the spikes from that subset of the data. When N is small enough, the data is filtered in such a way that a smoother signal is obtained. The algorithm implemented in this experiment follows the following structure:

1. Select a subset of the data, starting from 0 to N - 1.

2. Compute the average, \( \mu \) and standard deviation, \( \sigma \), of the subset.

3. Loop through the subset of data and evaluate each point, \( d_n, n \in 0, 1, ..., N - 1 \), to determine whether the data point lies within a \( \sigma \) of the mean, that is

\[
d_n \in (\mu - a\sigma, \mu + a\sigma),
\]

where \( a \) is a parameter selected by trial and error. The term \( d_n \) is the data point, \( \mu \) is the mean, \( \sigma \) is the standard deviation, and \( a \) is the filtering multiplier. The filtering multiplier is constant. If \( d_n \in (\mu - a\sigma, \mu + a\sigma) \), it is left unaltered,
but if the datapoint lies outside the range, it is clipped to $\mu - a\sigma$ or $\mu + a\sigma$, whichever is nearest.

4. The window is shifted forward to evaluate the points $N$ to $2N - 1$. Steps 2 through 4 are repeated.

5. At the end of the dataset, if the window is larger than the number of remaining data points, the algorithm is repeated simply using all leftover data points, which inherently has a smaller window size.

### 3.3 Timing Alignment

Timing alignment is a simple idea, but one which is necessary in the use of multiple sensors. When multiple sensors are measuring the same data as a function of time, it is inherent for there to be some amount of delay between start times data collection. When multiple sensors are being used, and each has its own delay from the initial start time, it becomes more crucial to align the timings of each sensor. The solution used here is interpolation of each sensor’s dataset, which shifts data points of all sensors to some regular interval agreed upon among all sensors. The initial sensor is described as the sensor that begins collecting data first. The subsequent sensors are the sensors that begin after the initial sensor. Here, the interpolation will be described for the initial and one subsequent. However, it should be noted that the process must occur between the initial and each subsequent.

Figure 3.2 illustrates the problem more clearly. Each of the signals shown in the figure are plotted relative to their own starting point. In actuality, the black signal (the initial) started at some point in time earlier. The difference in time is represented by $\Delta t_{\text{start}}$. It can easily be observed that if the red signal (the subsequent) were shifted by $\Delta t_{\text{start}}$, then the two signals would be in phase. The result of the shift is two signals which are aligned within the same relative time frame.
Figure 3.2: Mock sensor data. The black plot represents data recorded by the initial whereas the red represents data recorded by the subsequent.

When recording data in this experiment, each datapoint recorded comes with a Unix timestamp. With this data, each subsequent can be aligned in the relative time frame of the initial by subtracting every recorded timestamp by the start time of the initial sensor, or the first cell in the initial sensor’s time array. This process is shown in the arrays illustrated below. Let $t_{\text{init}} = t_i[0]$.

$$
t_i = [t_0 - t_{\text{init}}, t_1 - t_{\text{init}}, ..., t_n - t_{\text{init}}]
$$

$$
t_{\text{sub}} = [t_0 - t_{\text{init}}, t_1 - t_{\text{init}}, ..., t_n - t_{\text{init}}]
$$

$$
d_{\text{sub}} = [d_0, d_1, ..., d_n]
$$

The term $t_i$ represents the time array of the initial sensor. The term $t_{\text{sub}}$ represents the time array of the subsequent sensor, and $d_{\text{sub}}$ represents the data array, which would contain acceleration data on some axis. Notice how all cells in both $t_i$ and $t_{\text{sub}}$
are subtracted by $t_{init}$. The result is $t_i$ beginning at 0 and $t_{sub}$ beginning at the $\Delta t_{start}$ between the initial and itself.

The next step is to interpolate data from the initial and data from the subsequent to some regular timing interval. For this thesis, an interval of 0.01 was chosen because it was a round number near the nominal sampling period, which was about 0.01205 seconds. Had each sensor been able to sample data at a constant frequency, that period would have been chosen for the interval. However, some small variability to the sampling frequency was observed when analyzing the data. Iteration through the datasets of each sensor enabled the interpolation. The interpolation followed the following rules:

1. Select the first cell in the acceleration array and the time array. These will be referred to as $a_n$ and $t_n$, where $n$ represents the index in the array.

2. Interpolate the value of $a_n$ in cell $n$ using the following equation.

$$a_{n,\text{new}} = a_n + \left( \frac{m \times t_{\text{interval}} - t_n}{t_{n+1} - t_n} \right) (a_{n+1} - a_n) \quad (3.1)$$

$t_{\text{interval}}$ represents the interval value mentioned previously and $m$ represents the multiplier of the interval.

3. Select the data in cell $n + 1$ ($a_{n+1}$ and $t_{n+1}$).

- If $t_{n+1} > m \times t_{\text{interval}}$, then increase $m$ until $t_{n+1} < m \times t_{\text{interval}}$.
- Else if $t_{n+1} < m \times t_{\text{interval}}$, then do not change $m$.

The conditions above ensure that the time to which the acceleration is being interpolated ($m \times t_{\text{interval}}$) always lies within the range $t_n < m \times t_{\text{interval}} < t_{n+1}$.

4. Repeat steps 2 and 3 until completion.
3.4 Frame of Reference Alignment

A frame of reference is an abstract coordinate system relative to some particular reference. For example, the navigation frame aligns its positive axes with true north, east, and the direction that points toward the center of the earth. Every body has its own relative frame of reference, often called the body frame, which changes with respect to the navigation frame as the body changes orientation. Before integrating acceleration data to estimate position, it is very useful to align the IMU’s body frame with the navigation frame so that double integration of acceleration data in the $x$, $y$, and $z$ axes yields displacements along the north-south, east-west, and down-up axes.

The IMU measures acceleration along three axes which are orthogonal to each other. These make up the $x$, $y$, and $z$ axes of the body frame of reference. To align these axes with the navigation frame, the body frame must be rotated. This can be done using Euler angles, which refer to the roll, pitch, and yaw of the body. Figure 3.3 illustrates these three angles. However, using Euler angles suffers from the Gimbal lock problem. This problem occurs when two rotational axes align, eliminating one degree of freedom.

![Figure 3.3: Euler Angles](image)

It is more useful to use quaternions, which provide a four-dimensional representation of pose and don’t suffer from Gimbal lock. To rotate the body frame, a quaternion rotation transformation can be applied to each three-dimensional acceleration vector output by the accelerometer. Since the library provided by richardstechnotes [11] calculates the quaternion for each state of the output data, no calculation must be
performed to obtain the quaternion. The quaternion in the output represents the pose in which the $+x$-axis is aligned with magnetic north, the $+y$-axis aligned with east, and the $+z$-axis aligned with the direction toward the center of the earth. The rotation transformation can be done by first converting the three-dimensional acceleration vector into four dimensions.

$$a_{out} = [0, x_{out}, y_{out}, z_{out}]$$  \hspace{1cm} (3.2)

$x_{out}$, $y_{out}$, and $z_{out}$ represent the acceleration outputs of the IMU’s accelerometers on each axis. The following series of quaternion multiplications can then be applied to rotate the vector [14].

$$a_{out, nav} = qa_{out}q^{-1} = 0 + ix_{nav} + jy_{nav} + kz_{nav}$$  \hspace{1cm} (3.3)

$q$ represents the quaternion returned in the output of RTIMULib2 library [11]. This is the same as the quaternion mentioned previously. The result $a_{out, nav}$ is a four-dimensional vector whose scalar term is zero. $x_{nav}$, $y_{nav}$, and $z_{nav}$ represent the coefficients of the three-dimensional acceleration vector in terms of the navigation frame. For further information on quaternion operations, see [14].

Rotational transformations of a vector using quaternions enables an easy way to understand translational movements of the body. In this way, the acceleration in the north direction is known no matter which way the device is oriented. The result is that instead of reading the body frame of reference accelerations and trying to determine the direction by analyzing the pose, the navigation frame of reference accelerations are known.
3.5 Numerical Integration

A common technique for numerical integral is the trapezoidal method. The form computes the integration in a piecewise-linear fashion by calculating the area under the triangle created by drawing a straight line from one point to the next. By knowing the acceleration and time of measurement, the velocity and then displacement can be computed.

\[
v_n = \frac{(t_n - t_{n-1})(a_n + a_{n-1})}{2}
\]  \hspace{1cm} (3.4)

\[
s_{n+1} = \frac{(t_{n+1} - t_n)(v_{n+1} + v_n)}{2}
\]  \hspace{1cm} (3.5)

The term \(a_n\) represents the acceleration at the \(n\)th point in the measured sensor output, \(t_n\) represents the time at the same point, \(v_n\) is the calculated instantaneous velocity at time \(t_n\), and \(s_{n+1}\) is the displacement from the origin \((n = 0)\) by time \(t_{n+1}\). It can be seen that three measured data points are required for the double integration to obtain displacement.
Chapter 4

Related Work

Presented here is a summary of work selected from two areas, data filtering and methods of tracking, that are directly related to achieving the objectives of this thesis. Each topic is discussed in a separate section.

4.1 Data Filtering

Zhang implements a Kalman filter in a system which uses a smartphone to track a user’s position [13]. His experimental data appears to smooth their yaw orientation estimates significantly. Figure 4.1 from his report seems to show that the Kalman filter is useful for smoothing the orientation estimates, effectively filtering out noise which may affect the MEMS gyroscope. Zhang notes that the spike in the smartphone measurements near 26 seconds is likely to be caused by magnetic interference from a stairway handrail and theorizes that the particular smartphone used in their experiment is relying on the accelerometer and magnetometer for orientation estimation [13]. Nevertheless, it appears that Kalman filtering, when applied to the smartphone, is able to detect the interference and accurately estimate the orientation. Most of the literature seems to utilize Kalman filters solely for estimating attitude (orientation). Nothing in the literature has mentioned using Kalman filters to smooth data from a
single sensor (take the accelerometer, for example), which is often very noisy. Both Kok [15] and Zhang [13] take notice of the inconsistency of magnetometer readings inside a building, and thus make a point to leave its readings out of one of the EKF presented.

Figure 4.1: Comparison of raw and filtered yaw angle measurements from a smartphone with a commercial gyroscope [13]

Other means of filtering, such as variable bandwidth estimation, support vector machines (SVM), and complementary filters, exist and are presented in [15, 16, 17]. The variable bandwidth estimation technique presented in [17] utilizes sinusoidal estimation to dynamically adjust the filtering bandwidth of the accelerometer in order to remove sensor and vibrational noise. The method is a pre-processing algorithm designed to smooth accelerometer data before integration. Xu [16] takes a machine learning approach by applying a support vector regression (SVR) to reduce sensor error. In terms of mean square error (MSE), SVR was shown to have a much greater reduction of error on both the accelerometer and the gyroscope when compared to methods such as autoregressive (AR) or neural network (NN) algorithms. Xu performs an experiment in which the sensor is set still and data is recorded for nearly 190 seconds. The result showed that the MSEs of position estimates for the most effective AR model and NN model were 25.84 meters and 66.71 meters, respectively,
while the MSE of the best SVR model was as low as 4.92 meters. The magnitude of this error is high when compared to the results in this thesis; however, the accuracy of accelerometers has likely improved since 2009, the time of Xu’s writing [16]. The model of the accelerometer used by Xu in the study was not given, so no comparison can be made between the equipment in Xu’s work and the equipment used in this thesis. Regardless, Xu’s experiment showed that the SVM approach is able to keep position estimates of a stationary IMU closer to 0 meters than the uncompensated method, which yielded position estimates that were off by a magnitude of more than 150 meters. For the gyroscope, the SVM method is capable of keeping attitude estimates near zero while the AR method grows in error linearly up to 5 degrees after 190 seconds and the uncompensated gyroscopic output error grows linearly up to almost 20 degrees in 190 seconds [16].

4.2 Methods of Tracking

The most straight-forward method of estimating position is to doubly integrate the acceleration data over time to obtain displacement. By integrating on all three axes of the navigation frame, the distance moved in each direction can easily be plotted as a function of time. Kok discusses this method but assumes the distance travelled in comparison to the earth, the Coriolis effect, and the magnitude of the earth’s rotation are all negligible [15]. Tsai [18] also implements the double integration method using two accelerometers placed in various arrangements on the arms, chest, and back. Tsai claims at one point to obtain 4cm accuracy after displacing the device a total of 45cm in a series of single-axis translational movements. The double integration method is also used in [19] to estimate the position of a robot. Figure 4.2 illustrates the results of the experiment, in which a robot was driven in a square path. Wongwirat [19] does not seem to achieve nearly as great of accuracy as Tsai. One reason could be that
Wongwirat’s test was conducted on a much larger scale, which would allow for more time for drift to occur. Another reason could be poorer filtering of IMU data.

Another commonly used tracking approach is to place the IMU on the foot of the pedestrian, whose position is then calculated by estimating the step length and counting the number of steps [13, 20, 21]. Zhang showed promising results by implementing this method using the IMU in a smartphone [13]. Figure 4.3 shows the results of a Zhang’s square test, which is similar to the one shown in Figure 4.2. Zhang uses three different methods of determining step-length and plots the three paths calculated along with the preset true path on the $x$-$y$ plane. The problem with this methodology is its inapplicability to non-biped animals and rolling vehicles. Additionally, this tracking method typically requires calibration to the subject, so little deviation to normal walking patterns would cause inaccuracy. A limp, for example, could have great impact on the system’s accuracy.
Unfortunately, much of the research reported in the literature features vague or non-existent descriptions of integration techniques. Some researchers use the simple trapezoidal method, in which a fixed time interval between IMU readings is assumed. Others use more specialized means to integrate. For example, Zhang [13] gives a nice explanation of the integration techniques used for estimating step length, but his approach consequently utilizes variables such as the length of the leg, which is not very generalizable.

4.3 Summary of Related Work

Most attempts in the literature to localize using inertial sensors alone have been too inaccurate to have meaningful use. Some tracking methods, such as the step length estimation method, are not generalizable to all types of moving bodies. Based on searches done for this research, it does not appear that an effective solution to this problem has yet been devised up to this point in time. It is clear, however, that highly effective filtering algorithms are required to obtain any kind of accuracy.
One interesting venture, and one not described in any of the literature, could be to implement an EKF solely for the purpose of smoothing accelerometer measurements. Other work hits on the idea of using duplicate sensors in an attempt to eliminate bias and obtain a more *true* reading by combining the measurements of all devices [18, 22].

One problem in IMU tracking discussed by [23] is developing a benchmark for accuracy. One example of a flawed benchmark occurring commonly in the literature is the use of graphs to compare a *true* path and a calculated path. The issue with this benchmark for evaluating accuracy is the uncertainty of position at a given point in time. For example, the estimated position could be two meters ahead of the actual position, yet the plot will appear accurate because it is representing solely the path traveled, and not comparing the time of arrival at each point. Therefore, careful attention must be given to the representation of experimental data. Eyobu [23] recognizes this need for a universal system for representing the accuracy.
Chapter 5

Experimental Setup

5.1 Equipment

For this experiment, six Seeed Studio IMU 10DOF sensors, which housed the MPU-9250, were connected to six Raspberry Pis using the GrovePi+ add-on board by Dexter Industries and Seeed Studio. The MPU-9250 is a sensor developed by InvenSense, which includes a three-axis accelerometer, three-axis gyroscope, three-axis magnetometer. More specific information on the MPU-9250 can be found in the product specification document [4]. Each of these sensors were mounted to an old RadioShack remote control (RC) truck. Each Pi was powered using an Anker PowerCore 5000 portable battery. Figure 5.1 shows the setup.

It can be seen from the figure that the sensors are oriented in different poses. This is done not only for convenience in mounting, but also to illustrate that the specific orientation of the device doesn’t matter. This is because the three acceleration axes of the body frame of reference are aligned with the axes of the navigation frame of reference (North and South, East and West, up and down). The devices were labeled 1, 2, 3, 4, 5, and 6 in order to facilitate positive identification of specific devices. Device $n$ was given the hostname imudev$<n>$ in order to facilitate positive
identification of specific devices.

A major component of the experiment relied on open source software from GitHub created by Richard Barnett [11]. This software, called RTIMULib2, enables Python interfacing with the sensors. Additionally, it provides calibration software, implements an EKF to estimate pose data, computes pose quaternions, and outputs all of its measured and computed data with a single function call.

Figure 5.2 shows an example software output. The third but last line, labeled ‘fusionQPose’, provides the quaternion value, which represents the \( q \) in (3.3). The rotational transformation is achieved by combining this \( q \) with the vector represented by ‘accel’. The tuple given by referencing ‘fusionPose’ provides the Euler angles estimations, which represent the pose of the device. The output ‘fusionPose’ is the Euler angle representation of the quaternion given by the output ‘fusionQPose’. Recall the motivation for using the quaternion for representing the pose instead of
the Euler angles is described in Section 3.4.

Figure 5.2: RTIMULib2 library output

5.2 Data Collection and Processing Method

This section presents a detailed explanation of the methods by which data was collected from the sensors and processed to produce the data shown in Section 6.1.

5.2.1 Remotely Accessing the Sensors

To collect data efficiently, a Python script was written and run on the experimenter’s laptop. The Python script uses SSH to remotely access every Raspberry Pi and execute the data collection scripts on each. The device must be connected to the same WiFi as the laptop in order for the remote access to work. Additionally, to use the subprocess library from Python for SSH, it is required to set up an SSH key on the experimenter laptop and copy the key to every Raspberry Pi.

The programs in the various Raspberry Pis are coordinated by using a timestamp ten seconds from the current time. This timestamp is passed to each device and is used as a command line argument for the execution of the Python data collection program described in 5.2.2. The device receives the timestamp and calculates how
much time is remaining until the current time matches the received timestamp. The Python collection script then uses the \texttt{time} Python package to sleep for the duration of the remaining time. This method ensures small delays between data collection start times.

### 5.2.2 Collection and Pre-processing

Several Python scripts, which work together to collect and process the data, were written and downloaded to each device. The total package was organized in the following way.

1. **main.py**

   This is the program to be executed. The \texttt{main.py} program communicates with all other scripts to collect and process the data. This script does the following:

   - Communicate with \texttt{collector.py} to load the calibration file and initialize the IMUs as described by the documents in [11].

   - Communicate with \texttt{collector.py} to obtain the collected data.

   - Send the collected data to \texttt{state.py} to process the data. The processed data is returned to \texttt{main.py}.

   - Write the processed data to a csv file, which is labeled with the naming convention \texttt{imulocout<year>-<month>-<day>-<hour>-<minute>.csv}.

2. **collector.py**

   The functionality of this program is described below.

   - Read the calibration file and initialize the IMUs based on the calibration settings. The calibration file is entitled \texttt{RTIMULib.ini}. This file is output after calibrating the device as described in the documents by Richard Barnett [11].
• Collect the data from the sensors using the RTIMULib2 Python binding. To do this, the function `getIMUData()` is called. This returns the data presented earlier in Figure 5.2. The returned object is placed in an array. Data is collected for a specified number of iterations. The default number of iterations is 2000. However, a different number of iterations can be supplied as a command line argument to `main.py`.

• The collected data is returned to `main.py`.

3. `state.py`

The functionality of this program is described below.

• The collected data is received from `main.py`.

• The state variables are updated by rotating the body frame of the IMU to the navigation frame. This is accomplished using the quaternion transformation described in Section 3.4.

• `state.py` returns the 3-axis acceleration vector respective to the navigation frame along with the timestamp for each acceleration vector to `main.py` to be output to a csv file.

5.2.3 Sensor Data Centralization

The data collection process occurs in each device. Each of the csv outputs must be centralized to a single location for post-processing. This was done manually using SFTP through a program called FileZilla. Each of the csv outputs was placed on the centralized laptop into a separate folder, which were labeled using the hostnames of each device.
5.2.4 Post-processing

The recorded experimental data was post-processed using several, locally prepared MATLAB scripts. MATLAB was used for its plotting functionality during the testing stages of the script development. The MATLAB scripts are described below.

1. plotter.m

This program is the main program, which communicates with all other described in the rest of this section. The main processing steps of this program are as follows:

(a) The data from each sensor’s output csv file is read into separate arrays which contain the accelerations of the x, y, and z axes and the timestamp for each datapoint.

(b) The data from one sensor is integrated twice using integrateArrays.m to obtain the velocity and displacement over time. The acceleration, velocity, and displacement of this sensor are plotted in MATLAB for later comparison to the fusion outcome.

(c) To view all sensor data together, the acceleration data from each sensor is plotted in subplots of the same figure.

(d) The window peak limiter is applied to the x-axis acceleration for each sensor using limitData.m.

(e) The timing alignment and interpolation is applied to the peak-limited data using interpolateToStepValue.m. Subsequently, the arrays are normalized as described in Section 3.3 using normalizeArraysToSameLength.m.

(f) The processed data from each sensor is then fused together using fuseRedundantData5.m.
(g) The fused acceleration data is integrated twice to obtain the velocity and displacement data. This data is plotted similarly to the raw data plot mentioned earlier.

(h) Finally, the acceleration, velocity, and displacement for the raw and the fused methods are written to six separate csv files to be used for plotting the curves presented in Section 6.1.

2. integrateArrays.m

This function iteratively executes the trapezoidal integration algorithm described in Section 3.5.

3. limitData.m

This function executes the window peak limiter described in Section 3.2.

4. normalizeArraysToSameLength.m

This function measures the length of each data array and then makes all data and time arrays the same length as the shortest array. The difference is never more than one or two cells since the programs execute nearly simultaneously.

5. fuseRedundantData5.m

This function fuses the data from five sensors. The reason for only fusing data from five sensors is described in the next section of this thesis. This function simultaneously iterates through each sensor’s processed acceleration array and averages the data from each cell.
Chapter 6

Determining the Effectiveness of Multiple IMU Fusion

Presented in this chapter are the experimental descriptions and results. In order, the chapter contains four sections: indoor tests, outdoor tests, signal to noise ratio analysis, and a discussion of the results.

6.1 Indoor Tests

Three different types of tests were performed in this experiment. The tests were performed inside a residential apartment. In order to further control the experiment, possible sources of magnetic interference (for example, a metal bar stool) were removed from the environment near the sensors. Additionally, for Test 3, pose estimation was monitored at all points along the test line prior to recording of data. This was performed to ensure there were no potential sources of magnetic interference along the path of the line. Because the RC truck has metallic components, it was also necessary to ensure that there was no magnetic interference due to the proximity of the sensors to the truck. It was found by monitoring the magnetometer data as the sensor approached the RC truck that no magnetic interference was present.
The necessity for observing the magnetic interference arose from a previous discovery, which seemed to show that the sensors were prone to a shift in pose estimation when brought near a metallic or magnetic object. Additionally, algorithms developed for this experimentation were tested in order to determine their accuracy. For example, the fusion algorithm was tested by fusing six duplicate instances of the same acceleration dataset. An accurate algorithm would theoretically yield the same acceleration plot as the input, and this was indeed the result.

It was determined in the process of the experimentation that the sensor imudev1 had higher acceleration magnitudes and much higher displacement estimations than other sensors. Figure 6.1 shows the acceleration output of each sensor for one static trial. It is clear from the figure that imudev1’s precision was much lower than all other sensors. The reason for the seemingly noisy readings from imudev1 is unknown, but they could be due to poor handling of the sensor or degradation of the sensor’s components. The sensor imudev1 was used in the research for approximately a year longer than all other sensors. Because of imudev1’s poor performance, fusion was performed without imudev1, and displacement accuracy for the fusion method significantly increased. As a result, the data from imudev1 was discarded, and the fusion only utilized the data from the other five sensors. Eliminating this source of error yielded much cleaner and clearer results.
Figure 6.1: Acceleration outputs from each device
6.1.1 Test 1: Uncalibrated Static

In this test, the truck with its mounted sensors was set stationary and data was recorded. All sensors were powered and remotely accessed to execute the python script to collect data for approximately 24 seconds. After writing collected data to a csv file, each sensor’s data was transferred via SFTP to a laptop for post-processing. Post-processing included timing alignment, window peak limiting, and integration to obtain velocity and displacement data.

Figure 6.2 shows the acceleration, velocity, and displacement after fusion of all sensors (red) and the raw single-sensor acceleration, velocity, and displacement (black). The single sensor measurements were recorded using imudev2. The plot of the raw acceleration data shows a displacement drift in the negative x direction. Based on the relative linear trend of the raw velocity, it is likely that the accelerometer is biased negatively. It can be seen that fusion significantly reduces the bias, which is illustrated by the roughly zero-slope of the fused velocity. The result yields a raw estimated displacement of approximately $24 \times 10^{-2}$ meters in the negative x direction (south) after 24 seconds. The fused estimated displacement is $1.6 \times 10^{-2}$ meters in the positive x direction (north) after 24 seconds.
Figure 6.2: Test 1: uncalibrated static placement. Acceleration, velocity, and displacement
6.1.2 Test 2: Calibrated Static

For this test, the sensors were calibrated using the RTIMULib2 calibration feature. This calibrates the accelerometers and magnetometers by analyzing the minimum and maximum values of each axis. Data was recorded while the RC truck was static. The objective of this test was to determine the effect of the calibration software and determine whether the sensors were more or less accurate with calibration.

Figure 6.3 shows the acceleration, velocity, and displacement estimated by raw single-sensor measurements and five-sensor fusion measurements. The single sensor measurements were recorded using *imudev2*. Both modes appear to have an initial negative acceleration bias; however, the fused data gains a positive acceleration bias after the initial downturn, which is clear by the linearity of the velocity data beginning around 2 seconds. The raw velocity, however, oscillates around the zero velocity mark. The result is an oscillation reflected in the displacement. At 24 seconds, the raw displacement reaches $4.7 \times 10^{-2}$ meters in the negative x direction, whereas the fused displacement is around $4.3 \times 10^{-2}$ meters in the same direction.

In terms of the overall displacement, the sensor fusion method is only barely more accurate than the single-sensor method. If accuracy is defined as the distance from the true displacement at each point in time, the plots seem to illustrate that the fusion method is still more accurate than the single-sensor method most of the time. As in Test 1, it is clear that sensor fusion centers the acceleration measurements more tightly around its *true* theoretical value.

The results indicated that calibration was effective for increasing the overall displacement accuracy of the single-sensor method. However, overall displacement accuracy slightly decreased for the fusion method.
Figure 6.3: Test 2: calibrated static placement. Acceleration, velocity, and displacement
6.1.3 Test 3: Calibrated Line Translation

The purpose of this test was to determine whether accuracy could be retained in a case where the sensors were moving. In this test, a straight line in the magnetic north direction was measured out using a tape measure and markers were placed at 1 meter, 2 meters, 3 meters, and four meters. Data recording began and the experimenter waited until 10 seconds had passed. Then, the experimenter pushed the RC truck along the straight line to the two meter mark. Then, the experimenter stopped and waited until the 23 second mark. The experimenter then continued to push the RC truck along to the 3 meter mark. The initial intention in this experiment was to utilize the remote control capabilities of the RC truck; however, an earlier trial using the remote showed the functionality to be erratic. In order to move the truck along more smoothly, the experimenter simply pushed the RC truck by hand.

Figure 6.4 plots the acceleration, velocity, and displacement of three different estimation strategies: raw, fused, and fused without peak limiting. All methods show near-zero velocity until 10 seconds. The fused method shows a displacement of roughly 13 centimeters after 10 seconds, the raw method estimates a displacement of 21 centimeters, and the non-limiting fusion method estimates a displacement of 15 centimeters. This displacement drift is slightly higher than drifts in earlier trials. For a total displacement at the end of recording, the fusion method estimates a displacement of 2.33 meters in the correct direction of displacement. The non-limiting fusion method estimates a displacement of 2.54 meters. The raw method estimates a displacement of 3.74 meters at the end of its recording time (24 seconds).

Despite relative accuracy in the total displacement, the displacement path over time was not estimated well by any of the methods. None of the methods were able to detect the stop at the 2 meter mark. The velocity plots illustrate a major decrease in velocity around 19 seconds and subsequently illustrate a period of constant velocity. However, it appears that the system was not capable of measuring the drop to 0
\( m/s \) accurately. This seems to indicate the presence of some bias as a result of accelerometer excitation. This bias is discussed in more detail later.

The purpose of plotting the estimated acceleration, velocity, and displacement from the non-limiting fusion method was to determine the effect of the peak limiting algorithm when handling more dynamic movement. Additionally, it was desirable to observe how much the peak limiter played a role in the ability of the fusion data to hone in on the \textit{true} acceleration value. The results showed that the peak limiter was helpful in removing large spikes in the data, which could be attributed to compounded noise from each of the sensors. The peak limiter fusion was able to attenuate the magnitude of the acceleration readings during more dynamic movement more effectively than the non-limiting fusion method. However, in times of zero acceleration, the peak limiter appeared to minimally improve the \textit{true} measurement centering ability.

When analyzing the acceleration data up to 10 seconds, during which the acceleration is roughly zero, it is found that the fusion method with limiting has a mean of \( 1.43 \times 10^{-3} \) \( m/s^2 \) and a standard deviation of \( 5.39 \times 10^{-3} \) while the fusion method without limiting has a mean of \( 1.52 \times 10^{-3} \) \( m/s^2 \) and a standard deviation of \( 7.97 \times 10^{-3} \). Therefore, limiting increases the accuracy of the mean and reduces the overall magnitude of the peaks. This information is helpful because it further supports the claim that fusion alone is better at obtaining a more accurate measurement of the \textit{true} acceleration.
Figure 6.4: Test 3: movement in a line. Acceleration, velocity, and displacement
6.2 Outdoor Tests

This section details the experiments which were conducted outdoors. The purpose of conducting outdoor experiments was to collect data along a longer path of displacement. One limitation of this set of tests is that no level stretch of ground with plenty of room to displace perfectly in the north direction was found. Therefore, the tests were conducted pointing approximately 15° toward the East direction from the North axis.

In each trial, the single-sensor method is represented by the sensor which most accurately estimated the displacement of the system. The purpose of this is to accurately examine the effectiveness of multiple sensor fusion. Comparing the output of the fusion method only to the worst sensor would result in faulty conclusions. The tests in the following subsections aim to compare the fusion method to both the worst-case and best-case single-sensor measurements.

6.2.1 Test 1: Static

The purpose of this test was simply to evaluate the accuracy of the system when stationary outside. The test simply involved placing the system outside on the same concrete pad which was used for all tests in this section.

Figure 6.5 shows the acceleration, velocity, and displacement plots for the single-sensor method and the fusion method. It can be observed from the figure that the overall estimated displacement of the fusion method after 24 seconds was 7.14 centimeters, and the estimated displacement of the single-sensor method was 21.09 centimeters. These estimations are relatively consistent with those observed in the indoor static tests.
Figure 6.5: Test 1: static placement. Acceleration, velocity, and displacement
6.2.2 Test 2: Power Drill Pull

The purpose of this test was to measure the behavior of the system as it was pulled mechanically at a speed that was roughly constant. In this test, the RC truck was tied to a string, which was attached at the other end to a power drill. The trigger was pulled on the power drill to reel the RC truck toward the drill. This test took place over a distance of about 3 meters. Displacement of the system began at around 14 seconds and ended at around 29 seconds.

From the acceleration plots in Figure 6.6, it is clear that the fused method once again seems to effectively decrease the variance on the average acceleration statistic. However, it is clear from the velocity and displacement plots that the non-limited fusion method did not estimate the velocity and displacement more accurately than the single-sensor method. The overall displacement estimation for the non-limited fusion method was 15.38 meters. The limited fusion method, with an overall displacement estimation of 7.69 meters, is only slightly more accurate than the single sensor method, which had an overall displacement estimation of 8.08 meters.

Based on the linearity of the non-limited method’s velocity plot, it seems that the non-limited method had a greater acceleration bias during times of motion than the other two methods. It should be noted that the single sensor plotted in Figure 6.6, imudev5, had the best overall estimation when compared to the other sensors. The worst single sensor output was imudev6, which had an overall displacement estimation of approximately 22 meters. The sensor imudev2, which had similar inaccurate displacement estimations as imudev6, also contributed to an increase in the overall displacement. When removing these two sensors from the fusion, the non-limited fusion overall displacement estimation reduces to approximately 10 meters.

All methods fail to detect the stop at 29 seconds. All methods show a near-zero acceleration after 29 seconds, and all show a nearly constant velocity from 29 seconds through the remainder of the data collection time. However, none of the methods show
any drop in velocity, and therefore, do not detect any stopping in the displacement path. If the displacement is measured at 29 seconds, or the time of stopping, both the single-sensor method and the fusion method estimate a displacement of approximately 2.3 meters, and the non-limited method estimates a displacement of 4.20 meters. If the stop had been detected, therefore, the displacement estimations would have been much more accurate.

As in indoor Test 3, there appears to be some bias induced only when the accelerometer is in motion. This is indicated by the linear increase in velocity only during the period of motion. While some of this velocity is due to the actual acceleration of the system, some component of it appears to be due the motion induced bias. In reality, the device could not have increased velocity linearly over the period of movement because there must have been some negative acceleration while the system was stopping. Therefore, the velocity plot should perhaps have some initial increase in velocity, then hold a relatively constant velocity, then decrease in velocity back to zero. Because the velocity before and after the period of motion is relatively constant, there seems to be some indication of an acceleration bias only while the system is in motion. This bias will be discussed more in the discussion of the results.
Figure 6.6: Test 2: power drill pull. Acceleration, velocity, and displacement.
6.2.3 Test 3: Hand Pull North

The purpose of this test was to increase the distance by which the system could be displaced. The system was pulled a distance of approximately 9 meters using a string attached to the RC truck. The string was pulled by hand starting around 13 seconds until the end of the data collection, which lasted a duration of 48 seconds.

Figure 6.7 shows the results of Test 3. The non-limited fusion method, which estimated a total displacement of 26.89 meters, again performed the most poorly out of all three methods. The overall displacement estimation for the fusion method was 13.87 meters and the overall displacement estimation for the single-sensor method was 17.88 meters. The limited fusion method, therefore, had the best overall displacement estimation.

Every method overshot the actual displacement by at least four meters. This is indicative of a bias in the accelerometer, which would cause a greater increase in velocity over the duration of data collection. Because the sensor has near zero displacement and velocity until motion, it is again likely that some kind of bias induced by motion exists in the accelerometers.

The non-limited fusion method appeared to have a greater acceleration bias, when compared to the other two methods in the trial. This bias is similar to that which was observed in the outdoor Test 2. It was observed that imudev2 and imudev6 again gave poor measurements. When removing these from the fusion, the non-limited estimation decreased to 22.66 meters. It is clear that though fusion can reduce the overall displacement estimation error of a sensor, a bad sensor can cause greater inaccuracy in the displacement estimation.
Figure 6.7: Test 3: hand pull in the north direction. Acceleration, velocity, and displacement
6.2.4 Test 4: Hand Pull South

The purpose of this test was to determine whether the system could accurately measure displacement in the negative direction. Therefore, this test was conducted in the opposite direction of Test 2 and Test 3, both of which preceded this test. The motivation for this test was based on the inability of the system to detect stops in outdoor Test 2 and indoor Test 3. It was hypothesized that the system would not measure a negative displacement since negative acceleration was not measured accurately in previous tests. The system was again displaced approximately 9 meters in this test.

Figure 6.8 illustrates the results of this test. It is difficult to evaluate the accuracy of the displacement estimations of each method since each method estimated displacement in the incorrect direction. Regardless, the estimated displacement for the single-sensor method was 6.00 meters, for the fusion method was 13.58 meters, and for the non-limited fusion method was 8.53 meters.

As predicted, the system estimated a displacement in the positive direction instead of the negative direction. However, the estimated displacement of the system, overall, was less than the estimated displacement in Test 3. This result may indicate the presence of a bias in the accelerometer measurements, which occurs only during motion. The idea is that the system estimated a lower displacement distance, despite the fact that the magnitude of the displacement was the same as that in Test 3, because the acceleration in the negative direction caused by the experimenter counteracted some positive bias of the accelerometer. The bias would also explain why each method’s displacement estimation in outdoor Test 2 and outdoor Test 3 overshot the actual displacement.
Figure 6.8: Test 4: hand pull in the south direction. Acceleration, velocity, and displacement
6.3 Signal-to-Noise Ratio Analysis

The objective of this thesis is to determine the effectiveness of fusing inertial data from multiple IMUs. Therefore, a good metric must be determined for evaluating the effect of multiple sensors. It has been observed from the majority of tests conducted in this experiment that either the non-limited fusion method or the fusion method with limiting often estimated the overall displacement more accurately. It was also observed from the static tests that the effects of noise seemed to be reduced through fusion.

To quantify the reduction of noise, a signal-to-noise ratio (SNR) analysis was performed using data from a static trial and data from a straight line movement trial, the latter of which will be referred to as the dynamic trial in this section. The autocorrelation function was used to compute the signal and noise energies. The autocorrelation function of a sequence $X_i$, $i = 1, 2, ..., N$ is computed as

$$R_{XX}(\tau) = \sum_{i=1}^{N} X_i X_{i-\tau}$$  \hspace{1cm} (6.1)

To get the signal energy, the autocorrelation can be evaluated at $\tau = 0$; that is,

$$E_{\text{signal}} = R_{XX}(\tau)|_{\tau=0}.$$  \hspace{1cm} (6.2)

The autocorrelations for the static and dynamic trials of each sensor were evaluated at $\tau = 0$ to determine the signal energy. Since the static trial theoretically measured no acceleration, the static acceleration data essentially represents ambient noise. The autocorrelation of the static data evaluated at a delay offset of zero, therefore, should produce the energy of the noise. The dynamic trial recorded acceleration and noise, so the autocorrelation evaluated at an offset of zero results in the energy of the signal and noise. The energy of the signal can be determined by subtracting the energy of the noise from the energy of the signal and noise, as shown in (6.3).
\[ E_{\text{acceleration}} = E_{\text{acceleration plus noise}} - E_{\text{noise}} \]  
\[ \text{SNR} = \frac{P_{\text{acceleration}}}{P_{\text{noise}}} = \frac{E_{\text{acceleration}}}{E_{\text{noise}}} \]  

The SNR was then calculated using (6.4).

\[ \Delta T \] represents the time period during which data was recorded. Since data was recorded for the same time length for the static and dynamic trials, the number of samples for each dataset is the same, and the time cancels. Therefore, the SNR can be calculated by taking the ratio of the energies of the acceleration and noise, represented by \( E_{\text{acceleration}} \) and \( E_{\text{noise}} \), respectively. The result of the energy and SNR calculations is depicted in Table 6.1.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>( E_{\text{noise}} )</th>
<th>( E_{\text{accel and noise}} )</th>
<th>( E_{\text{accel}} )</th>
<th>SNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>imudev2</td>
<td>0.2322</td>
<td>188.8</td>
<td>188.6</td>
<td>812.2</td>
</tr>
<tr>
<td>imudev3</td>
<td>0.2135</td>
<td>154.3</td>
<td>154.1</td>
<td>721.9</td>
</tr>
<tr>
<td>imudev4</td>
<td>0.2286</td>
<td>109.2</td>
<td>108.9</td>
<td>476.6</td>
</tr>
<tr>
<td>imudev5</td>
<td>0.2384</td>
<td>128.8</td>
<td>128.6</td>
<td>539.3</td>
</tr>
<tr>
<td>imudev6</td>
<td>0.3478</td>
<td>103.5</td>
<td>103.2</td>
<td>296.6</td>
</tr>
</tbody>
</table>

With the exception of imudev6, all noise energies are approximately 0.2. The acceleration energies range between 100 and 200. The SNRs have a wide range, which extends from 296.6 to 812.2. The SNR is large for all sensors. This is valuable because larger values of the acceleration are not drowned out by noise. As the noise energy approaches zero and the signal energy is constant, the SNR increases infinitely. Therefore, the objective is to increase the SNR by reducing the noise energy. Table 6.2 shows the signal and noise energies along with the resulting SNR for various combinations of sensors included in fusion. The values in this table were calculated using the autocorrelation of various sensor fusion combinations.
Table 6.2: Noise, signal plus noise, and signal energies and SNR of various averages of sensor acceleration output

<table>
<thead>
<tr>
<th>imudev Combination</th>
<th>$E_{\text{noise}}$</th>
<th>$E_{\text{accel plus noise}}$</th>
<th>$E_{\text{accel}}$</th>
<th>SNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 &amp; 3</td>
<td>0.1098</td>
<td>127.5</td>
<td>127.4</td>
<td>1159</td>
</tr>
<tr>
<td>2 &amp; 3 &amp; 4</td>
<td>0.07423</td>
<td>107.4</td>
<td>107.3</td>
<td>1445</td>
</tr>
<tr>
<td>2 &amp; 3 &amp; 4 &amp; 5</td>
<td>0.05538</td>
<td>100.8</td>
<td>100.8</td>
<td>1820</td>
</tr>
<tr>
<td>all</td>
<td>0.04942</td>
<td>89.04</td>
<td>88.99</td>
<td>1801</td>
</tr>
</tbody>
</table>

The table clearly indicates that the noise energy increases as the number of sensors used in fusion increases, with the exception the inclusion of imudev6 in the fusion. An increased SNR from fusion means that noise in the system is being reduced, which would likely increase the acceleration measurement accuracy. This supports the hypothesis that the use of sensor fusion is effective for tracking.

The reason that imudev6 decreases the SNR when included in fusion is because its SNR is so low compared to the SNRs of the other sensors. Additionally, it is found that the fusion of imudev6 with any other sensor increases the noise energy of the other sensor and decreases the signal energy of the other sensor. This is shown in the tables below. It was decided to try all two-sensor combinations of fusion to determine whether some pairs had a greater increase in SNR and to analyze the effect of imudev6’s reduction of the fused SNR. The noise energies are shown in Table 6.3. The signal plus noise energies are shown in Table 6.4. The signal energies are shown in Table 6.5. Finally, the corresponding SNRs for all possible two-sensor combinations are shown in Table 6.6. It should be noted that the diagonal entries in all of the following tables represent the single-sensor data only. Therefore, the single-sensor noise energies, signal energies, signal plus noise energies, and SNR are displayed for all sensors in the tables below.
### Table 6.3: Noise energies for all possible two-sensor combinations

<table>
<thead>
<tr>
<th></th>
<th>imudev2</th>
<th>imudev3</th>
<th>imudev4</th>
<th>imudev5</th>
<th>imudev6</th>
</tr>
</thead>
<tbody>
<tr>
<td>imudev2</td>
<td>0.2322</td>
<td>0.1098</td>
<td>0.1144</td>
<td>0.1158</td>
<td>0.1469</td>
</tr>
<tr>
<td>imudev3</td>
<td>0.1098</td>
<td>0.2135</td>
<td>0.1114</td>
<td>0.1091</td>
<td>0.1423</td>
</tr>
<tr>
<td>imudev4</td>
<td>0.1144</td>
<td>0.1114</td>
<td>0.2286</td>
<td>0.1174</td>
<td>0.1445</td>
</tr>
<tr>
<td>imudev5</td>
<td>0.1158</td>
<td>0.1091</td>
<td>0.1174</td>
<td>0.2384</td>
<td>0.1427</td>
</tr>
<tr>
<td>imudev6</td>
<td>0.1469</td>
<td>0.1423</td>
<td>0.1445</td>
<td>0.1427</td>
<td>0.3478</td>
</tr>
</tbody>
</table>

### Table 6.4: Signal plus noise energies for all possible two-sensor combinations

<table>
<thead>
<tr>
<th></th>
<th>imudev2</th>
<th>imudev3</th>
<th>imudev4</th>
<th>imudev5</th>
<th>imudev6</th>
</tr>
</thead>
<tbody>
<tr>
<td>imudev2</td>
<td>188.8</td>
<td>127.5</td>
<td>110.2</td>
<td>124.5</td>
<td>119.0</td>
</tr>
<tr>
<td>imudev3</td>
<td>127.5</td>
<td>154.3</td>
<td>116.9</td>
<td>121.4</td>
<td>90.83</td>
</tr>
<tr>
<td>imudev4</td>
<td>110.2</td>
<td>116.9</td>
<td>109.2</td>
<td>93.45</td>
<td>77.78</td>
</tr>
<tr>
<td>imudev5</td>
<td>124.5</td>
<td>121.4</td>
<td>93.45</td>
<td>128.8</td>
<td>88.45</td>
</tr>
<tr>
<td>imudev6</td>
<td>119.0</td>
<td>90.83</td>
<td>77.78</td>
<td>88.45</td>
<td>103.5</td>
</tr>
</tbody>
</table>

### Table 6.5: Signal energies for all possible two-sensor combinations

<table>
<thead>
<tr>
<th></th>
<th>imudev2</th>
<th>imudev3</th>
<th>imudev4</th>
<th>imudev5</th>
<th>imudev6</th>
</tr>
</thead>
<tbody>
<tr>
<td>imudev2</td>
<td>188.6</td>
<td>127.4</td>
<td>110.1</td>
<td>124.4</td>
<td>118.8</td>
</tr>
<tr>
<td>imudev3</td>
<td>127.4</td>
<td>154.1</td>
<td>116.8</td>
<td>121.3</td>
<td>90.69</td>
</tr>
<tr>
<td>imudev4</td>
<td>110.1</td>
<td>116.8</td>
<td>108.9</td>
<td>93.34</td>
<td>77.63</td>
</tr>
<tr>
<td>imudev5</td>
<td>124.4</td>
<td>121.3</td>
<td>93.34</td>
<td>128.6</td>
<td>88.30</td>
</tr>
<tr>
<td>imudev6</td>
<td>118.8</td>
<td>90.69</td>
<td>77.64</td>
<td>88.30</td>
<td>103.2</td>
</tr>
</tbody>
</table>
As illustrated by Table 6.6, the results seem to indicate that, with the exclusion of \textit{imudev6}, the result of sensor fusion between any two sensor pairs is an increase in SNR for both sensors over their respective single-sensor SNR values. This seems to indicate that in the majority of cases, fusion of two sensors will increase the SNR in an acceleration signal.

From analyzing the tables, the reason for the reduction of SNR when fusing \textit{imudev6} becomes more clear. From Table 6.3, it can be seen that fusion of a sensor with \textit{imudev6} increases the noise energy of that sensor. Furthermore, Table 6.5, shows that fusing a sensor with \textit{imudev6} results in the reduction of that sensor’s signal energy. It is clear that the reduction of a sensor’s SNR when paired with \textit{imudev6} is due to an increase in noise energy and a decrease in signal energy.

Conversely, the SNR of \textit{imudev6} can be increased by fusing it with another sensor that has a higher SNR. This is helpful for \textit{imudev6}. The results indicate that fusion of a bad sensor with good sensors can increase the signal strength of the bad sensor. However, the cost is the reduction of signal strength for the good sensor.

One source of error in this analysis is that the energy of the signal is an estimate. The reason is that the bias seems to only occur during the movement of the sensor. To show this, the acceleration data from the dynamic trial can be represented as
\[ \dot{a}_{\text{measured}}(t) = a_{\text{actual}}(t) + n(t), \quad (6.5) \]

where \( a_{\text{measured}}(t) \) represents the acceleration measured and output by the sensor, \( a_{\text{actual}}(t) \) describes the actual acceleration of the system, and \( n(t) \) represents the noise. Based on the bias, which seems to only occur during movement, it appears that (6.6) shown below more accurately describes the acceleration signal of the dynamic movement trial.

\[ a_{\text{measured}}(t) = a_{\text{actual}}(t) + n(t) + b + \Delta b(t), \quad (6.6) \]

where \( b \) represents some constant bias and \( \Delta b(t) \) describes some changing bias over time. This bias is difficult to characterize, but must be done in order to accurately approximate the energy of the signal. This is discussed in the conclusions.

### 6.3.1 Timing Alignment for SNR Analysis

It was discovered during the analysis that the dynamic and static trials used to calculate the energies and SNRs had faulty time stamps. This was discovered because it was found that peaks in the acceleration measurements were not aligned when plotted together. This was true despite the fact that the sensors began recording at the same time within an average of ±0.001 second accuracy. The cross correlation showed that the misaligned peaks were offset by close to 0.15 seconds while the time stamps of the data points reported an offset of 0.0003 seconds. Figure 6.9 shows the misaligned acceleration signals. An example of misaligned peaks can be seen between 17 and 18 seconds.
Cross correlation was used to solve this problem. The cross correlation of two acceleration datasets results in a peak at a location which indicates the timing offset between the two acceleration signals. Figure 6.10 shows the cross correlation between imudev2 and imudev3 before alignment.

The x-axis represents half the number of data points in the acceleration data array. The reason the x-axis represents half of the data points is because cross correlations are symmetric. The right half of the cross correlation is plotted here. It can be
seen from the figure that the peak is not perfectly at zero; it is shifted over by 12 data points. To fix this, \texttt{imudev2} was aligned by removing 12 data points from the beginning of its array. Consequently, 12 data points were removed from the end of the acceleration data array for \texttt{imudev3} to make both arrays equal length for fusion. Figure 6.11 shows the cross correlation of the shifted acceleration data. The result is location of the peak at zero. Figure 6.12 shows the resulting aligned acceleration signals.

![Cross Correlation](image1)

**Figure 6.11: Aligned cross correlation of \texttt{imudev2} and \texttt{imudev3}**

![Aligned Acceleration Plots](image2)

**Figure 6.12: Aligned acceleration signals of \texttt{imudev2} and \texttt{imudev3}**
The observation of the misaligned peaks was critical to the SNR analysis. Previously, the analysis had indicated that the SNR decreased as the number of sensors included in fusion increased. This further illustrates the necessity for aligned timings when fusing data. Even the slightest shift can be shown to significantly change the SNR, thereby further burying the actual acceleration amidst the noise.

6.4 Discussion of Results

The objective of this thesis is to determine the effect of fusing data from multiple IMUs to track displacement. The results of the indoor tests seemed to indicate that fusion is helpful in obtaining a more accurate reading of the acceleration data over time. The decrease in the variance on the mean in the accelerometer is the most supportive of this claim. Whereas single-sensor output showed static acceleration nominally at $\pm 0.02m/s^2$, fused-sensor output showed static acceleration at nominally $\pm 0.002m/s^2$, a magnitude reduction factor of 10. Additionally, indoor Tests 1 and 3 illustrated that the fusion method was able to reduce the bias in the sensors. Bias removal becomes clear when analyzing the slopes of the velocity plots. For those two tests, the fusion velocity consistently showed smaller magnitudes in slopes. In Test 1, the result was a near-zero constant velocity. In Test 3, smaller slope magnitudes in the fusion data resulted in an overall shift downward of the fusion velocity plot in comparison with the raw velocity plot. Table 6.7 shows the mean and standard deviation for the samples recorded during indoor Test 1 and indoor Test 2, in which the system was static. Therefore, the mean should theoretically be 0 $m/s^2$. From the data, it is clear that both the mean and the standard deviation around the mean are reduced by almost a factor of ten. Not only did fusion measure the acceleration to be closer to the true acceleration of the system, but fusion also reduced the overall range of measurements around the mean.
Table 6.7: Mean and standard deviation comparison between the fused method and the single-sensor method for indoor Test 1 and Test 2

<table>
<thead>
<tr>
<th></th>
<th>Fused</th>
<th>Raw</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test 1</strong></td>
<td>Mean ($m/s^2$)</td>
<td>Mean ($m/s^2$)</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>5.16e-5</td>
<td>-4.96e-4</td>
</tr>
<tr>
<td><strong>STD</strong></td>
<td>0.00236</td>
<td>0.0104</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Fused</th>
<th>Raw</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test 2</strong></td>
<td>Mean ($m/s^2$)</td>
<td>Mean ($m/s^2$)</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>4.43e-5</td>
<td>-1.73e-4</td>
</tr>
<tr>
<td><strong>STD</strong></td>
<td>0.00227</td>
<td>0.0140</td>
</tr>
</tbody>
</table>

Attenuation from the peak limiting algorithm seemed to be helpful for the indoor tests and for most of the outdoor tests, excluding outdoor Test 4. In most cases, peak limiting resulted in more accurate total displacement estimates, as well as closer centering of measured data around the true acceleration value. Such an observation is mostly in regard to the ability of the combined peak limiter and sensor fusion to detect zero acceleration more accurately, as discussed previously.

The results of the outdoor tests did not perfectly coincide with the indoor tests. In every case, except Test 4, the limiting fusion method estimated a more accurate overall displacement. However, the fusion method without limiting, seemed to perform the most poorly in every case except Test 4, where it outperformed the limiting fusion method.

In all outdoor trials, every method’s displacement estimations were more erratic than the estimations from the indoor tests. This result can be attributed to several sources of error present in the outdoor tests. First, the sensors were not pointed perfectly in the North direction as they were in the indoor tests. Second, there was likely some magnetic interference due to the large metal building next to the concrete pad on which the outdoor tests were conducted. Finally, it was found that two sensors, `imudev2` and `imudev6`, appeared to output more inaccurate measurements when compared with the most accurate sensor, `imudev5`. The accuracy mentioned here is based upon the overall displacement estimation of each sensor. The result of the inaccuracy is poorer performance in the fusion method.
It was found during all dynamic trials that there appears to be some bias which is induced by movement of the sensor. This is indicated by the velocity plots, which report a linear velocity during times of acceleration. The linear velocity seems to be independent of whether the acceleration is constant or not. In other words, whether the acceleration oscillates or is constant, there is some bias which causes the velocity to increase linearly with small oscillations. In outdoor Test 2, for example, the velocity increases linearly during the period of time in which the acceleration signal is active, meaning the acceleration measurements are not zero. However, the velocity plot does not show a decrease in velocity during the stopping period. Before the acceleration begins and after it ends, the velocity plot shows relatively no increase in velocity, which indicates a zero acceleration. This means that there is no significant acceleration bias during periods of inactivity. During periods of excitement, however, the accelerometer appears to adopt some bias, which causes inaccuracy in the velocity and displacement estimations.

The signal-to-noise ratio analysis seemed to support the use of multiple sensors for fusion. Generally, the SNR increases as the number of sensors included in fusion increases. This indicates that the use of fusion can reduce the noise energy in an acceleration sample, which would theoretically reduce the displacement estimation error. Analysis of all two-sensor fusion combinations revealed that the SNR increases for both sensors, excluding the combinations which involve \texttt{imudev6}. The sensor \texttt{imudev6}, which had a lower SNR than all other sensors, was shown to decrease the SNR when included in the fusion. Therefore, the results indicate that the inclusion of a noisy dataset in fusion will most likely decrease the SNR of the overall system.

There is some limitation to the SNR analysis, however. The signal energy is based on the assumption that there is no bias in the dynamic acceleration trial. The reality, however, is that there is some constant bias throughout the whole signal, and some bias which seems to be induced by accelerometer excitation.
Chapter 7

Conclusion

In this thesis, basic system function for common sensors located on IMUs was discussed. Algorithms utilized for filtering data, estimating pose, aligning frames of reference, and performing numerical integration were presented. Subsequently, current work in the topic area was presented. The previous topics mentioned supplied background information that was useful in evaluating the effectiveness of multiple IMUs to improve tracking accuracy.

Some of the results of the experiments conducted in this thesis appeared to support the hypothesis that fusion of multiple IMU data is able to represent more accurately the true kinematics of the system, thereby increasing tracking accuracy. Tighter standard deviation in the fused acceleration illustrates the elimination of biases and noise in the measurements, and a reduction in the mean measured acceleration by a factor of ten for the static case indicates that sensor fusion yields greater measurement accuracy. In each indoor test, the fused system consistently estimated a more accurate total displacement than the single-sensor system. However, Test 3 illustrated a limitation of the INS system in this thesis: path displacement was not estimated accurately when measuring more dynamic movement. Neither the single-sensor method nor the fusion method were able to detect the halt in displacement in the middle of
Test 3.

Although the fusion method often most accurately estimated the overall displacement in the outdoor tests, the results of the outdoor tests were not very conclusive. It was observed, however, that there seems to be some bias which is induced by the motion of the accelerometers and only active during the period of motion. This is certainly a point for further investigation. Particularly, it would be helpful to characterize this bias mathematically and utilize the result to determine whether displacement can be estimated more accurately.

Analysis using signal-to-noise ratio was presented and discussed. The resulting signal and noise energies, along with the SNRs, from several combinations of sensor fusions were calculated and shown in various tables. Limitations in the analysis were discussed. Overall, fusion was shown to increase the SNR of the acceleration signals.

Future work would focus on investigating the sources of error in sensor data. Through the experimentation, three main types of sensor error were discovered: static bias, motion induced bias, and noise. Future work would include the characterization of each source of error. It would also be valuable to quantify the errors from each source. If one particular source of error was found to affect the system more, it could provide good direction for which source to investigate. The purpose for understanding the sources of bias and noise would be to design filters and controllers to reduce error. Error reduction would likely result in more accurate tracking.

It would also be valuable to investigate the use of different IMUs. It was found during the experimentation that the Seeed Studio IMU 10DOF sensors significantly declined in performance after roughly a year since the purchase date. This could have been due to exposure to hot or cold temperatures, or due to the deterioration of the sensor’s components. Therefore, future work would test the accuracy of other IMUs.
Bibliography


