Pendulum State Estimation Via Sensor Fusion Using MPU6050 with Arduino UNO

Braedan Kennedy
bkenne07@calpoly.edu

Luis David Garcia
lgarc120@calpoly.edu

California Polytechnic State University San Luis Obispo
San Luis Obispo, California, USA

EE 525-01 Stochastic Processes
Team Group Number: 01

Professor: Dr. Clay McKell

March 13, 2025

Abstract

This report details the final phase of a project focused on estimating the state of a pendulum using sensor fusion with a 6DOF Inertial Measurement Unit (IMU) based on the MPU6050 sensor, interfaced with an Arduino UNO. The study employs a linear state-space model to estimate the pendulum's angular position and velocity. Data collected from the IMU and a camera-based tracking system, which provides ground truth via visual markers, is used to validate the model. Sensor noise is characterized, enabling a comparison of two sensor fusion techniques: the Kalman filter and the complementary filter. The Kalman filter emerges as the superior method, demonstrating its effectiveness in accurately estimating the pendulum's states from noisy sensor readings.

Contents

1 Introduction

Sensor fusion is a critical technique for enhancing the accuracy and reliability of sensor data by combining measurements from multiple sources. In this project, we utilize the MPU6050 from Adafruit, which integrates a 3-axis accelerometer and a 3-axis gyroscope, providing a total of 6 degrees of freedom (DOF) [1]. This allows for the estimation of both the angle and angular velocity of a pendulum, which is the focus of this study.

The MPU6050 facilitates sensor fusion by allowing us to leverage the complementary measurements from its accelerometer and gyroscope, thus mitigating noise and improving the precision of our readings.

We employ an Arduino UNO interfaced with the MPU6050 through Adafruit's MPU6050 Arduino library [2], utilizing the I2C communication protocol. This library enables us to obtain acceleration and angular velocity data in three axes (x, y, z), along with timestamps for data acquisition. The data is transmitted over UART (Universal Asynchronous Receiver/Transmitter) for storage and analysis. Additionally, a camera system with visual markers is used to track the pendulum's position, providing a ground truth for validating our measurements.

In this report, we begin by detailing the specifications of the MPU6050 and describing our experimental setup. We then analyze the data collected from both the IMU and the camera, focusing on characterizing the noise in the sensor readings. A linear state space model is introduced to estimate the pendulum's states, with the optimal model parameters determined by fitting the model to the camera tracking data. Finally, we compare the performance of both a Kalman filter and a complementary filter in estimating the angular displacement and velocity of the pendulum.

2 Background

This study aims to estimate the states of a pendulum system by collecting data from two carefully designed experiments using the MPU6050 Inertial Measurement Unit (IMU). These experiments lay the groundwork for applying advanced tools, such as the Kalman filter, to accurately estimate the system's states:

1. **Pendulum at Rest**: When the pendulum is stationary, the experiment aims to characterize the noise present in the sensor readings. Analytical techniques like autocorrelation and power spectral density (PSD) are used to describe the noise.

2. **Pendulum Swing**: This experiment captures the pendulum's motion and examines its dynamics. Sensor data from the gyroscope and accelerometer are used to estimate key state variables, such as angular displacement and velocity.

   A vision system is used simultaneously to track the pendulum's motion, providing a ground truth for comparing the IMU-based state estimations and validating the state-space model's alignment with real-world observations.

These experiments are interconnected: the noise characterization from the stationary pendulum informs the assumptions required for modeling the pendulum's dynamic behavior.

2.1 Sensors - MPU6050 Specifications

The MPU6050 is a versatile 6-axis IMU that features an accelerometer and gyroscope. Key specifications are summarized in Table 1, sourced from the manufacturer's datasheet [3].

The MPU6050 uses 16-bit registers for both the accelerometer and gyroscope, meaning it can represent measurements using 65,536 discrete values across each selectable range of linear acceleration and angular velocity. In later sections we discuss how we configured the MPU6050 for operation in our experiment, and how those choices affect the resolution of the data we were able to collect.

The MPU6050's accelerometer and gyroscope are each optimized for different sample rates and power levels, balancing performance and efficiency. The accelerometer can sample up to 1 kHz, while the gyroscope reaches up to 8 kHz in fast mode. This versatility allows us to gather detailed data without excessive power consumption; the accelerometer draws around 0.165 mW, while the gyroscope requires just 0.012 mW.

Communication with the MPU6050 is managed through the I2C protocol, ensuring efficient data transmission and straightforward device configuration. Address options and clock frequency settings are documented in the device's register map [4].

Regarding calibration, the manufacturer did not recommend any as they include built-in firmware with the sensor which can perform calibration at runtime [3]. While the device is factory-calibrated for sensitivity, a user-performed static calibration can further minimize bias in both the accelerometer and gyroscope, especially as factory calibration may vary slightly between units.

Both the accelerometer and gyroscope deliver direct measurements along three axes (x, y, z), capturing linear acceleration and angular velocity, respectively. Figure 1 visualizes

Table 1: IMU Sensor (MPU6050) Specifications

Specification	**Details**
Resolution	Accelerometer: 16-bit (±2g, ±4g, ±8g, ±16g) Gyroscope: 16-bit (±250, ±500, ±1000, ±2000 °/s)
Sample Rate	Accelerometer: 4 Hz (min), 1 kHz (max) Gyroscope: 1 kHz (slow mode), 8 kHz (fast mode)
Power Consumption	Accelerometer: 0.165 mW (typical) Gyroscope: 0.012 mW (typical)
IDD Power Supply	Accelerometer: 500μA (typical) Gyroscope: 3.6μA (typical)
VDD Power Supply	2.375V (min), 3.46V (max)
Temperature Range	-40 °C (min), +85 °C (max)
I2C Address	0x68 or 0x69 (user selectable)
I2C Clock Frequency	100 kHz (standard mode), 400 kHz (fast mode)
Calibration Options	Manufacturer does not provide any recommendations for calibrations instead they note the sensor has run-time calibration firmware
Direct Measurements	Accelerometer: Linear acceleration in 3 axes (x, y, z) Gyroscope: Angular velocity in 3 axes (x, y, z)
Indirect Measurements	Accelerometer: Velocity and displacement (via integration) Gyroscope: Orientation (via integration)

the sensor axes in a 3D coordinate system.

The accelerometer utilizes Micro-Electro-Mechanical Systems (MEMS) technology, with each axis (x, y, z) featuring a proof mass that shifts slightly in response to acceleration. This shift alters the capacitance between the proof mass and fixed components, producing a measurable change that corresponds to acceleration on each axis [3].

The gyroscope operates in a similar fashion, relying on MEMS technology to detect rotational changes. Each axis has a small vibrating structure that responds to rotation via the Coriolis effect, which causes the structure to shift slightly. This movement changes the capacitance, which is then amplified, filtered, and converted into a voltage proportional to the rotation rate. Both sensors employ 16-bit analog-to-digital conversion, delivering precise digital outputs across the selectable resolution ranges presented in Table 1.

By leveraging these direct measurements, our ultimate objective is to implement sensor fusion through Kalman filtering, allowing us to derive accurate orientation of a pendulum.

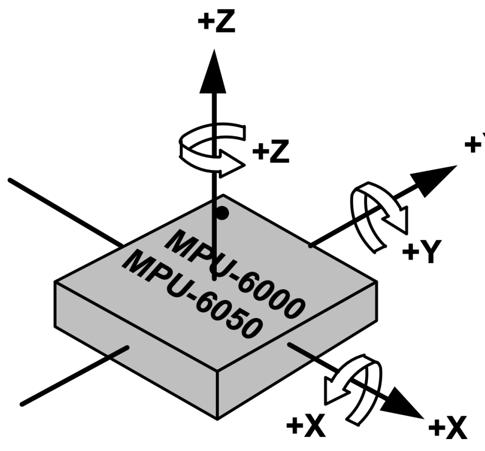

Figure 1: Three-Axis Coordinate System with x, y, and z Coordinates [3]

2.2 Random Processes

2.2.1 Linear Systems

A linear system is defined by a fundamental principle known as superposition. To clarify, superposition means that when multiple inputs act on a system simultaneously, their combined effect equals the sum of their individual effects [5]. This principle is essential and can be broken down into two key characteristics:

- **Additivity**: When two different inputs are applied to a system separately and then together, the system will behave predictably. If input $x_1(t)$ produces output $y_1(t)$, and input $x_2(t)$ produces output $y_2(t)$, then applying both inputs together $(x_1(t) + x_2(t))$ will result in an output that is the sum of the individual outputs $(y_1(t) + y_2(t))$.
- **Homogeneity (or Scaling)**: Changing the size (amplitude) of an input by some factor will cause the output to change by the same factor. For example, if doubling an input doubles the output, the system demonstrates linear scaling behavior.

These properties make linear systems particularly valuable in engineering, as they enable the decomposition of complex problems into simpler parts that can be solved individually and then combined.

2.2.2 Linearity and Time-Invariance

When analyzing systems, two important properties often go hand in hand [5]:

- **Linearity**: As described above, a linear system responds proportionally to its inputs and allows us to break down complex inputs into simpler components.
- **Time-Invariance**: A system is time-invariant if it behaves the same way regardless of when we apply an input. In other words, if we apply the same input at different times, the parameters remain constant and the output is the same just shifted in time.

While many real-world systems are not perfectly linear or time-invariant, they often behave this way within certain operating ranges. Understanding these properties helps us predict how systems will behave and design better ways to control them.

2.2.3 Stationarity and Ergodicity

In the analysis of random processes and sensor measurements, two key statistical properties are particularly important [6]:

- **Stationarity**: A process is considered stationary when its statistical characteristics, such as mean and variance, remain constant over time. This property is crucial for developing reliable models and making predictions about system behavior.
- **Ergodicity**: In an ergodic process, the statistical properties calculated from a single, sufficiently long observation of the process are equivalent to those calculated from multiple observations at a single time point. This means that time averages can effectively represent ensemble averages, making it possible to characterize the system's behavior through long-term observations.

These fundamental properties are essential for developing robust stochastic models and implementing effective filtering techniques in real-world applications.

2.2.4 Autocorrelation and Power Spectral Density

In the analysis of random processes and signals, two fundamental analytical tools provide essential insights into signal characteristics [6]:

- **Autocorrelation**: This function measures the statistical correlation between different points in a signal separated by a time lag. For a signal $x(t)$, the autocorrelation $R_{xx}(\tau)$ at time lag τ reveals temporal patterns and periodicities within the data. High autocorrelation indicates strong temporal dependencies, while low autocorrelation suggests more random behavior. This analysis is particularly valuable for:
  - Identifying repeating patterns in signals
  - Determining the persistence of random processes
  - Evaluating the independence of measurements over time
- **Power Spectral Density (PSD)**: The PSD function $S_{xx}(f)$ for the same signal $x(t)$ describes how the power of a signal is distributed across different frequencies. It is mathematically related to the autocorrelation function through the Fourier transform, providing complementary information in the frequency domain. This analysis helps in:
  - Understanding the frequency composition of signals
  - Identifying dominant frequencies and their relative strengths
  - Characterizing noise types (e.g., white noise, colored noise)

Together, these tools provide a comprehensive framework for analyzing and characterizing random processes, enabling better system modeling and filter design. The relationship between autocorrelation and PSD, known as the Wiener-Khinchin theorem, establishes that they are Fourier transform pairs, allowing analysis in both time and frequency domains.

2.3 State Space

State space representation is a fundamental framework for analyzing and controlling dynamic systems [5]. This approach describes systems through five key components:

- **Inputs**: External influences that affect the system's behavior. These can be categorized into two types:
  - **Deterministic Inputs**: Known and controlled signals or forces that can be precisely determined and manipulated, such as control commands or reference signals.
  - **Stochastic Inputs**: Random disturbances and uncertainties that affect the system, including measurement noise, process noise, and environmental factors.
- **States**: A set of variables $x(t)$ that completely characterize the system's internal condition at any given time t. The state vector contains the minimum amount of information needed to:
  - Predict the system's future behavior given future inputs
  - Represent the system's dynamic properties
  - Enable the design of control strategies
- **Stability**: Stability refers to the behavior of the system's response over time, specifically whether the system's output will remain bounded or eventually diverge [5]. The Routh-Hurwitz criterion is used to analyze the stability of systems by examining the characteristic polynomial's roots (system eigenvalues) [7].
  - Stability requires all eigenvalues of the A matrix (state transition matrix) to have negative real parts.
  - The Routh-Hurwitz criterion determines stability by constructing a Routh array from the characteristic polynomial.
  - If all entries in the first column of the array are positive, the system is stable.
  - Stability implies that the system's states won't grow without bound and will converge to equilibrium.
- **Outputs**: Measurable quantities $y(t)$ that can be observed or recorded from the system. These are typically:
  - Direct measurements from sensors
  - Derived quantities calculated from sensor data
  - Functions of the system states and inputs, expressed mathematically as $y(t) = h(x(t), u(t))$, where $h(\cdot)$ is the measurement function that maps states and inputs to observable outputs. For linear systems, this becomes $y(t) = \mathbf{C}x(t) + \mathbf{D}u(t)$, where $\mathbf{C}$ is the output matrix and $\mathbf{D}$ is the feedthrough matrix

- **Observability**: A crucial system property that determines whether all states can be reconstructed from output measurements over time. This concept is analyzed using two mathematical tools:
  - **Observability Matrix $\mathcal{O}$**: A mathematical construct that determines if the system is completely observable. For a linear system described by matrices $(\mathbf{A}, \mathbf{C})$, where $\mathbf{A} \in \mathcal{R}^{n \times n}$ is the state transition matrix and $\mathbf{C} \in \mathcal{R}^{p \times n}$ is the output matrix, the observability matrix is:

    $$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

    where n is the dimension of the state vector. The system is observable if this matrix has full rank.
  - **Observability Gramian $\mathcal{G}$**: A matrix that quantifies the degree of observability for each state, providing insight into estimation quality. For a continuous time-invariant system, it is defined as:

    $$\mathcal{G}_c = \int_0^{T_s} e^{\mathbf{A}^T t} \mathbf{C}^T \mathbf{C} e^{\mathbf{A}t} dt$$

    where T_s is the observation time interval, $\mathbf{A}^T$ denotes the transpose of matrix $\mathbf{A}$, and $e^{\mathbf{A}t}$ represents the matrix exponential of $\mathbf{A}t$. The matrix exponential describes how the system's states evolve over time in the absence of inputs.

    The discretized obervability Gramian can be described as:

    $$\mathcal{G} = \mathcal{O}^T \mathcal{O}$$

    Where the observability Gramian in discrete time is the sum of the outer products of the rows of the observability matrix.

These components form the foundation for modern control theory and state estimation techniques, enabling the development of advanced control systems and observers for various applications in engineering and science.

2.4 Estimation and Fusion

2.4.1 Kalman Filter

The Kalman filter is a recursive estimation algorithm designed to minimize the mean square error (MSE) between the measured and estimated states. This draws heavily from the concept of mean square error estimators, which aim to make the following quantity as small as possible:

$$E\left[(x - \hat{x})^2\right]$$

A visual representation of the Kalman filter algorithm is shown in Figure 2, adapted from [6]. The algorithm involves several key variables and operations, which are detailed in Table 2.

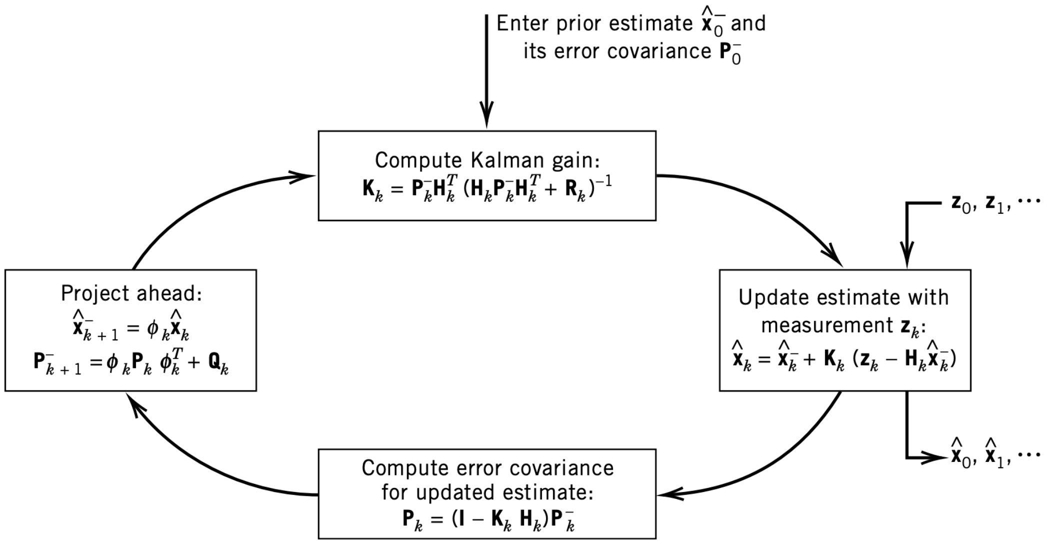
Enter prior estimate $\hat{\mathbf{x}}_0^-$ and its error covariance $\mathbf{P}_0^-$
Compute Kalman gain: $\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$
$\mathbf{z}_0, \mathbf{z}_1, \cdots$
Update estimate with measurement $\mathbf{z}_k$: $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)$
$\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \cdots$
Compute error covariance for updated estimate: $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^-$
Project ahead: $\hat{\mathbf{x}}_{k+1}^- = \phi_k \hat{\mathbf{x}}_k$ $\mathbf{P}_{k+1}^- = \phi_k \mathbf{P}_k \phi_k^T + \mathbf{Q}_k$

Figure 2: Kalman Filter Loop [6]

Table 2: Kalman Filter Symbols and Descriptions

Symbol	**Description**	**Notes**
$\hat{x}_k^-$	A priori state estimate	Estimate before incorporating the measurement.
P_k^-	A priori error covariance matrix	Covariance of a priori estimate.
ϕ_k	State transition matrix	Models how the state evolves.
Q_k	Process noise covariance	Represents process noise uncertainties.
K_k	Kalman gain	Balances measurement and state estimation.
H_k	Measurement matrix	Maps state space to measurement space.
R_k	Measurement noise covariance	Represents uncertainties in the measurements.
z_k	Measurement input	Observed measurement at time k.
$\hat{x}_k^+$	A posteriori state estimate	Estimate after incorporating the measurement.
P_k^+	A posteriori error covariance matrix	Updated covariance matrix after incorporating the measurement.

The concepts discussed earlier, such as states and system modeling, are integral to implementing the Kalman filter. However, the algorithm relies on several strict assumptions, the most important being that the system must be linear. Additionally, it requires an initial estimate of the process state, referred to as the a priori state estimate, along with an initial estimate of the associated priori error covariance matrix.

Referring to Figure 2, the core operations of the Kalman filter can be summarized as follows:

1. Compute the Kalman gain.
2. Update the state estimate using the measurement z_k.
3. Compute the error covariance for the updated estimate.
4. Project ahead to the next state estimate.

This process repeats iteratively, aiming for convergence to the minimal $E[(x-\hat{x})^2]$. While the filter can continue indefinitely, it is often unnecessary to do so beyond achieving sufficient convergence, as it would waste computational resources.

2.4.2 Complementary Filter

The complementary filter is a simpler and computationally less expensive alternative to the Kalman filter, also a minimum mean square estimator. While it is less accurate, it effectively reduces the mean square error (MSE) in the state estimate. Unlike the Kalman filter, the complementary filter does not rely on strict assumptions such as system linearity or prior knowledge of the covariance matrices. Instead, it leverages a combination of low-pass and high-pass filters, as illustrated in Figure 3.

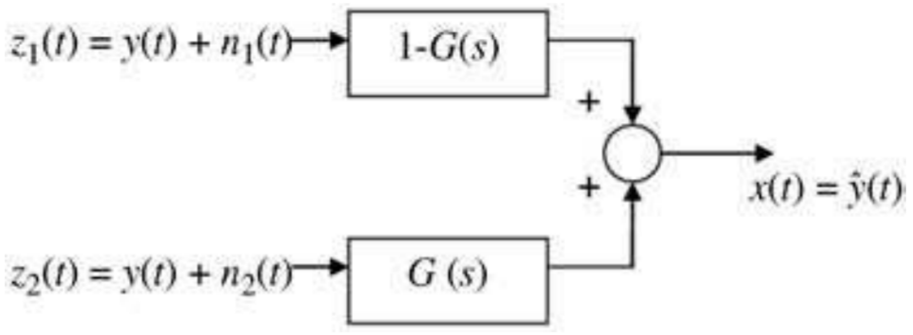

Figure 3: Complementary Filter Diagram

The complementary filter operates by separating measurements into low-frequency and high-frequency components. This approach allows it to effectively reduce noise and improve the accuracy of the estimated signal. The resulting output, $X(s)$, is an optimal combination of the filtered components, as shown in Figure 4.

$$X(s) = \underbrace{Y(s)}_{\text{Signal term}} + \underbrace{N_1(s)[1 - G(s)] + N_2(s)[G(s)]}_{\text{Error term}}$$

Figure 4: Complementary Filter Output Equation

The symbols used in Figures 3 and 4 are detailed in Table 3 for reference.

Table 3: Complementary Filter Symbols and Descriptions

Symbol	**Description**	**Notes**
$z_1(t)$	Input measurement with noise $n_1(t)$	Represents the noisy low-frequency signal.
$z_2(t)$	Input measurement with noise $n_2(t)$	Represents the noisy high-frequency signal.
$Y(s)$	True signal component	Assumed to be common to both inputs.
$G(s)$	Filter gain	Combines low-pass and high-pass filtering for fusion.
$N_1(s)$	Noise component of $z_1(t)$	Low-frequency noise present in $z_1(t)$ in the Laplace domain.
$N_2(s)$	Noise component of $z_2(t)$	High-frequency noise present in $z_2(t)$ in the Laplace domain.
$X(s)$	Filtered output signal	Combines the filtered contributions of $z_1(t)$ and $z_2(t)$.
$1 - G(s)$	Complement of the filter gain	Filters out low-frequency components.

The gain $G(s)$ plays a critical role in optimizing the filter's performance. By carefully selecting $G(s)$, the complementary filter effectively balances the contribution of low-frequency and high-frequency components, reducing the overall noise while preserving the true signal.

A key aspect of the complementary filter is the error term in the output equation:

$$N_1(s)[1 - G(s)] + N_2(s)[G(s)]$$

Assuming $N_1(t)$ and $N_2(t)$ are independent, this can be decomposed into two separate error terms, each associated with a specific measurement:

$$N_1(s)[1 - G(s)]$$

$$N_2(s)[G(s)]$$

These terms represent the mean square error contributions from the low-frequency and high-frequency measurements, respectively. The total mean square error (MSE) is then the sum of these two components, $E[e_1^2]$ and $E[e_2^2]$. This formulation highlights the complementary filter's ability to achieve a balance between the errors from each measurement, providing an effective estimate of the true signal.

3 Experimental Procedure

3.1 MPU6050 Configuration

Due to the versatile nature of the MPU6050 sensor, it can be configured in many ways. For our experiment, we selected the configuration parameters listed in Table 4. Additionally, the complete hardware and software requirements for the experiment are provided in Appendix C.

Table 4: IMU Sensor (MPU6050) Experiment Configuration

Specification	**Configuration**
Resolution	Accelerometer: ±4g Gyroscope: ±500°/s
Sample Rate	Accelerometer: 1 kHz (max) Gyroscope: 8 kHz (fast mode)
Power Consumption	0.0125 W (gyro + accel, DMP disabled)
VDD Power Supply	3.3V (nominal)
Temperature	+20 °C (nominal)
I2C Address	0x68
I2C Clock Frequency	100 kHz (standard mode)
Calibration	No user calibration
Direct Measurements	Accelerometer: Linear acceleration in 3 axes (x, y, z) Gyroscope: Angular velocity in 3 axes (x, y, z)

Our choice of resolution was ultimately determined through trial and error. Starting at the lowest range, of $\pm 2g$ for the accelerometer and ±250°/s for the gyroscope we conducted our experiment and then checked to see if the sensor at any point became saturated in its output. To check for this, a MATLAB script B was developed to see if any sensor reading matched the maximum value allowable in the selected range. If so, we would increase the range for that sensor and repeat the experiment.

For our selected ranges of $\pm 4g$ for the accelerometer and ±500°/s for the gyroscope, at a 16-bit resolution, this equates to a sample resolution of:

$$\frac{2 \cdot 4g}{65536} \approx 0.00012g$$

for acceleration, and

$$\frac{2 \cdot 500^\circ/\text{s}}{65536} \approx 0.015^\circ/\text{s}$$

for angular velocity.

We opted for the maximum possible sample rates of 1 kHz and 8 kHz for the accelerometer and gyroscope, respectively. These rates allow us to capture as many samples as possible, enabling us to study subtle variations in the pendulum's acceleration and angular velocity. The MPU6050 is set to sample the gyroscope and accelerometer at these maximum rates, ensuring that the latest data is available in the sensor's first in, first out (FIFO) buffer when read. However, due to limitations in the processing speed of the Arduino UNO and the bandwidth constraints of the serial communication, the actual data logging rate is limited to 125 Hz. This update rate is enforced in software to accommodate these hardware restrictions.

The power consumption for this setup is 0.0125 W, which includes both the gyroscope and accelerometer, with the Digital Motion Processor (DMP) disabled. Disabling the DMP reduces power consumption and aligns with our goal of measuring raw sensor outputs.

We used a 3.3V nominal voltage for the VDD power supply, as our Arduino board conveniently provides this voltage through a dedicated pin. The sensor was operated at normal room temperature of +20°C, as the pendulum experiment was conducted in a controlled lab environment at standard room temperature.

For communication via I2C, we set the I2C address to 0x68 and the clock frequency to 100 kHz (standard mode), as these settings were compatible with our setup. The I2C address did not conflict with other devices, and the 100 kHz frequency was the only option supported for our selected Arduino library [2].

No additional user calibration was performed, as the factory calibration provided sufficient accuracy for the predictable motion of the pendulum swing, allowing us to focus on data collection.

As discussed in a previous section, the accelerometer and gyroscope are capable of measuring linear acceleration and angular velocity along three axes (x, y, z). For our experiment, all three axes were used for both types of measurements. The linear acceleration captures the pendulum's back-and-forth motion, while the angular velocity tracks rotational changes around the pivot, giving us a comprehensive view of the pendulum's dynamics.

3.2 Wiring Connections

Figure 5 illustrates the wiring diagram of the MPU6050 connected to the Arduino Uno.

The essential connections include VCC, GND, the serial clock line (SCL), and the serial data line (SDA). The SCL and SDA lines are necessary for I2C communication, while VCC and GND provide power to the MPU6050. Although the INT pin is shown in the connection diagram, we do not utilize it for interrupts; instead, we employ a software polling mechanism to acquire readings from both the accelerometer and gyroscope.

3.3 Experiment Design and Diagram

For our experiment, we decided to swing the Arduino and MPU6050 from its USB cable like a pendulum and record the raw sensor data output.

First, we made the necessary connections from the MPU6050 to the Arduino UNO, as shown in Figure 5, using jumper wires and a breadboard. We then secured the Arduino and MPU6050 to the breadboard with tape to physically connect all components.

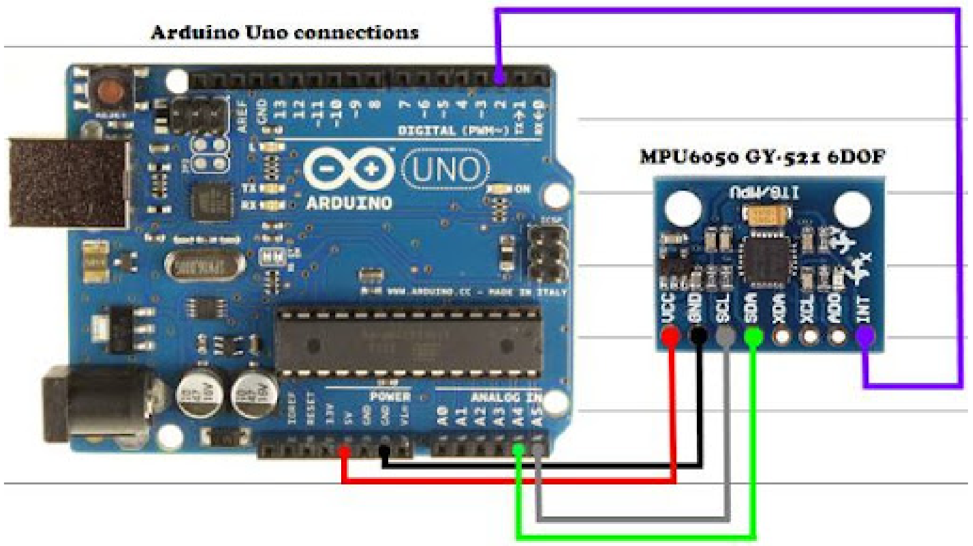

Figure 5: Wiring Diagram from Connecting MPU6050 with Arudino Uno [8]

Next, we attached a 6-foot USB cable to the Arduino and connected it to an x64-based laptop. Using the Arduino IDE on the laptop, we uploaded our custom firmware to the Arduino UNO.

Then, we hung the Arduino and MPU6050 by the USB cable, raised it to one side, started our data logging script on the laptop, and then released the Arduino and MPU6050, allowing it to swing by its USB cable until the oscillations appeared to stop.

In parallel, a green visual marker of negligible mass was attached to the Arduino to facilitate tracking its motion. A video camera was then positioned perpendicular to the plane in which the pendulum was to be swung to capture the movement of the pendulum during the oscillations.

Additionally, four visual markers were placed on the plane in which the pendulum was to be swung, arranged in the shape of a rectangle. To ensure that the markers formed a precise rectangle, we were inspired by the ancient Roman Groma. The markers were suspended by a string of known length from a reference marker at the top, ensuring accurate placement and alignment.

Finally, when the oscillations have stopped, we performed an additional data collection for later analysis of the noise present in the sensor readings when in a static state.

Figure 6 depicts how we conducted the pendulum swing experiment, and Figure 7 details the orientation of the MPU6050 axis during the experiment. Note that in Figure 6, the reader is actually in the position of the video camera. Also, the orientation of the MPU6050 axis as shown in Figure 7 is depicted such that the negative X-axis points out of the page towards the reader.

3.4 Visual Data Extraction

To extract the pendulum's motion from the video feed, a semi-automated computer vision analysis tool was designed and implemented.

The process begins with an algorithm that detects the precise pixel coordinates of the

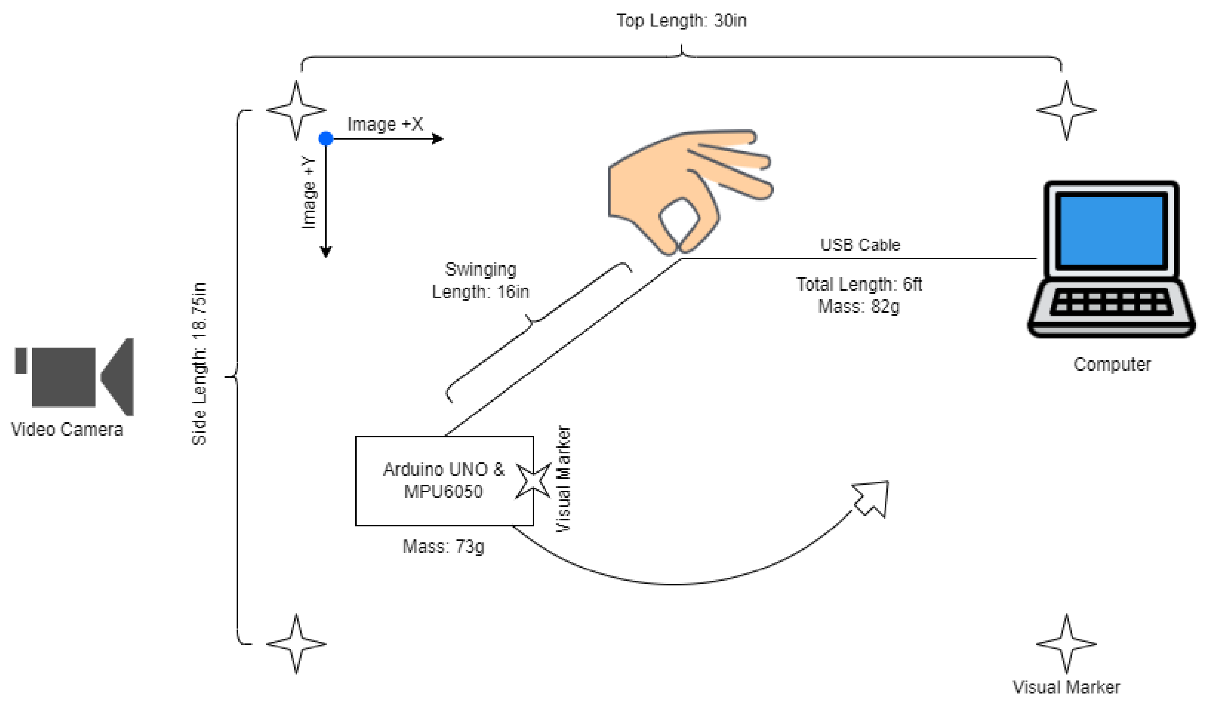

Figure 6: Experiment (Pendulum Swing) System Diagram

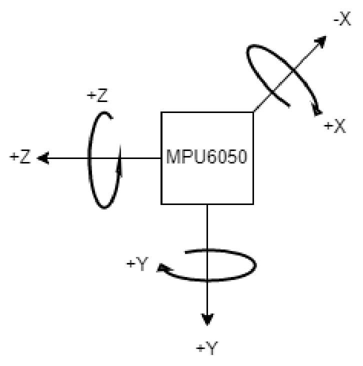

Figure 7: Experiment (Pendulum Swing) MPU6050 Axis Diagram

visual markers in each frame. These coordinates are then organized into a polygon, which is rendered for visualization, as shown in Figure 8a.

Next, a homography is computed to transform the video frame's 2D plane into the plane of pendulum motion, which is defined by the visual markers. A homography is a projective transformation between two planes [9], allowing for the mapping of 2D image coordinates to real-world coordinates. This transformation is possible because the dimensions of the pendulum motion plane are known. As a result, pixel values are converted to real-world units (e.g., 20 pixels per inch), allowing for a direct observation of the pendulum's motion. The effect of this transformation is illustrated in Figure 8b, where the transformed frame is shown.

Subsequently, the frame is cropped to contain only the area within the markers. The only manual step in this process is determining the location of the pendulum's pivot point (the top of the pendulum). As shown in Figure 9, the user is prompted to click on the pivot point in the frame to indicate its position.

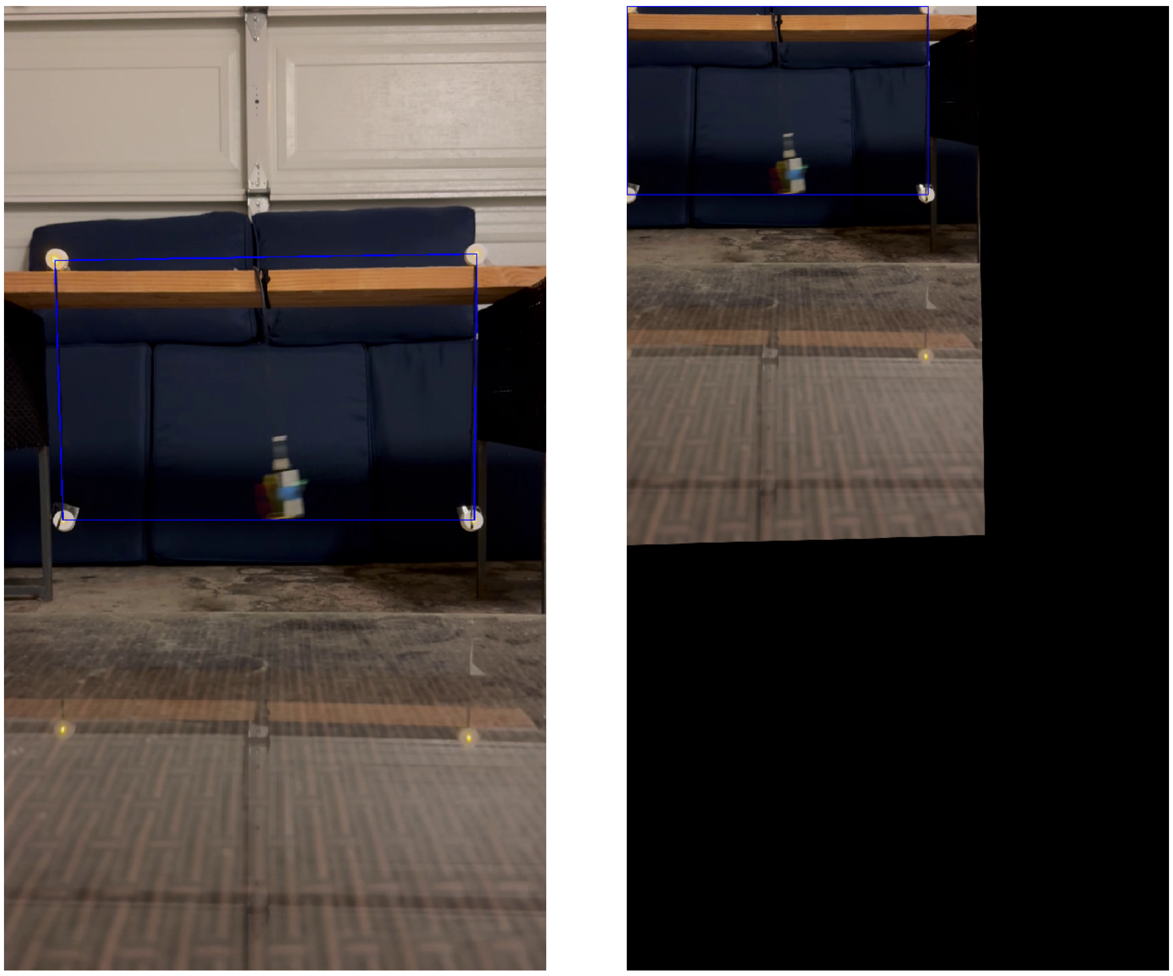
(a) Detected Visual Markers in Frame (b) Transformed Frame

Figure 8: Detected Markers and Transformed Frame

Once the top of the pendulum is identified, an automated algorithm tracks the position of the green marker attached to the bottom of the pendulum. This position is recorded over time, providing the necessary data for motion analysis. The final state of the visual analysis tool, showing the tracking of the pendulum's position, is presented in Figure 10.

After all video frames are processed, the tool generates a CSV file containing the tracked data. This file can be imported into MATLAB for further analysis and integration with other sensor data.

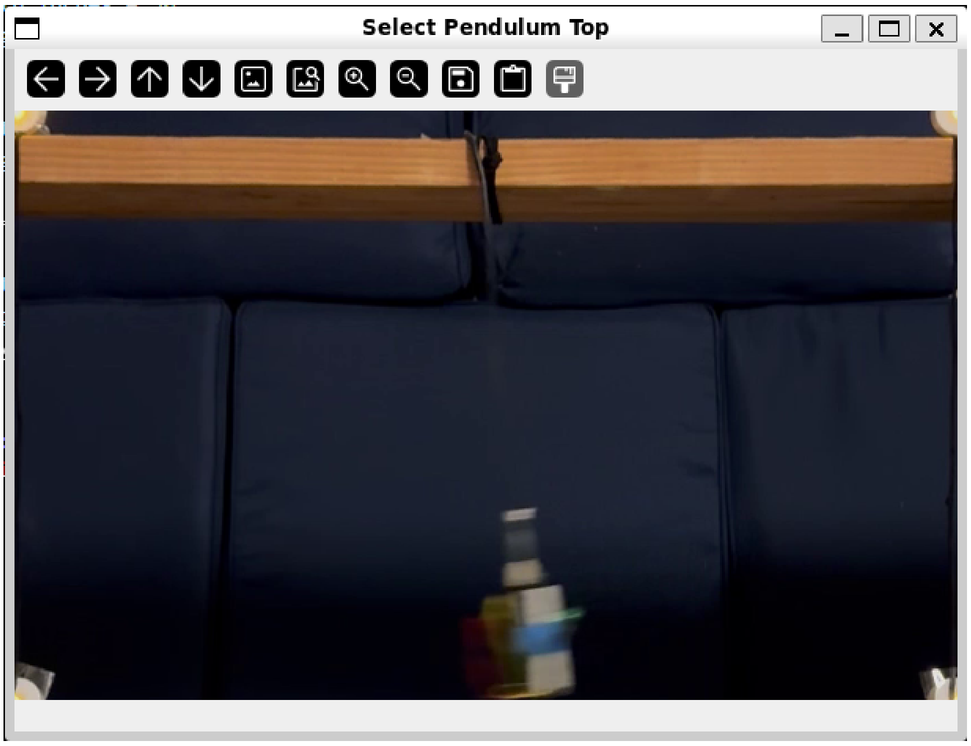

Figure 9: Prompt to Select Pendulum Pivot Location

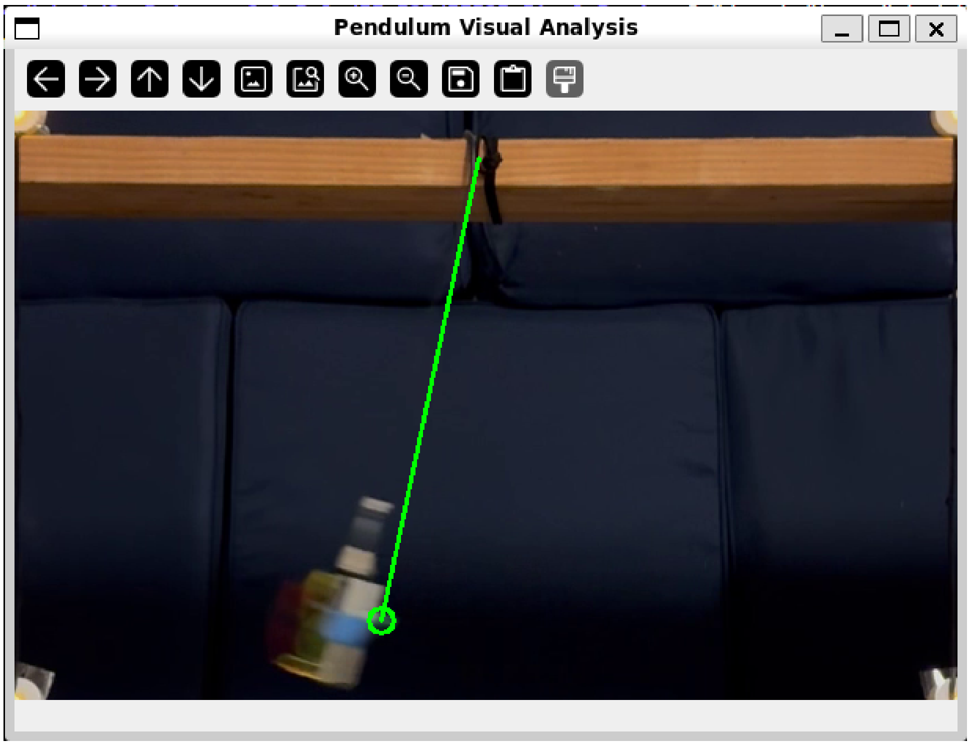

Figure 10: Tracking the Pendulum's Motion

4 Data Analysis and Discussion

4.1 Swinging Pendulum Data Analysis

For our experiment, we collected linear acceleration and angular velocity data using an MPU6050 sensor swinging like a pendulum, sampled at 125Hz, during a 70-second duration until the system reached steady state. Simultaneously, we collected position data using a camera-based tracking system, sampled at 30Hz, for later use as a ground truth in validating pendulum state estimation methods.

4.1.1 Accelerometer Data

Figure 11 presents the three-axis accelerometer measurements over time. The data shows linear acceleration in $\frac{\text{m}}{\text{s}^2}$ along the x, y, and z axes plotted against time in milliseconds.

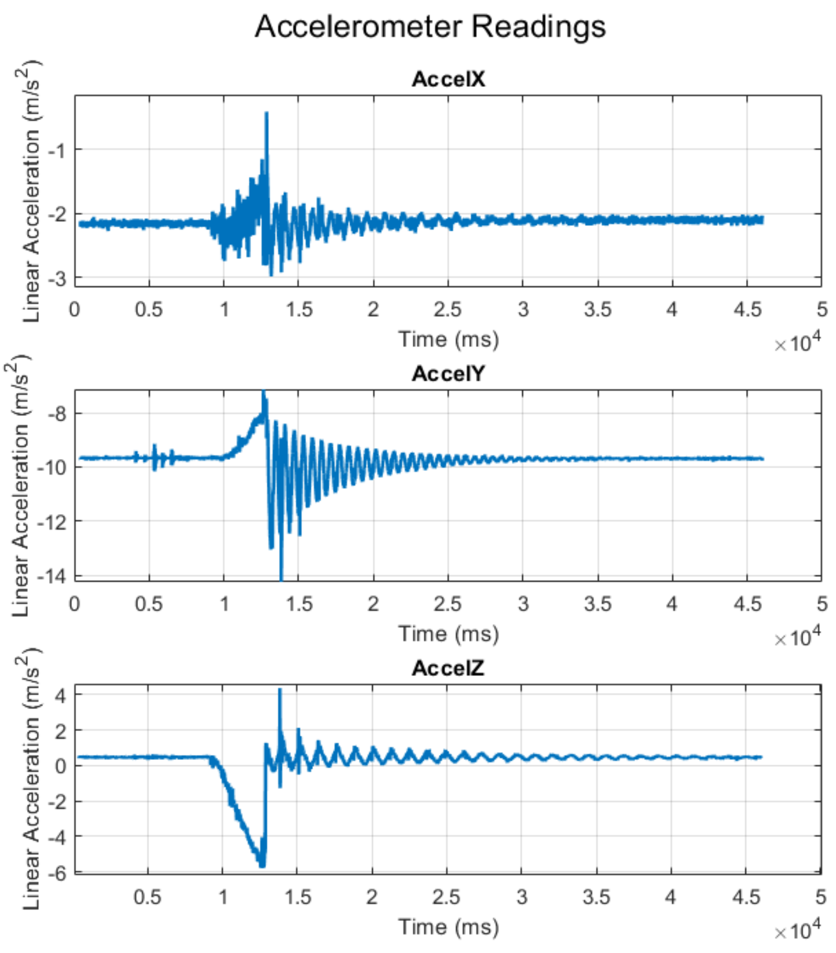

Figure 11: Raw accelerometer measurements showing tri-axial acceleration during pendulum motion

The acceleration along all three axes exhibits damped oscillations. It reflects the motion of a pendulum, with each consecutive swing gradually slowing down until the mass comes to rest.

4.1.2 Gyroscope Data

Figure 12 presents the three-axis angular velocity measurements over time. The data shows angular velocity in $\frac{\text{rad}}{\text{s}}$ along the x, y, and z axes plotted against time in milliseconds.

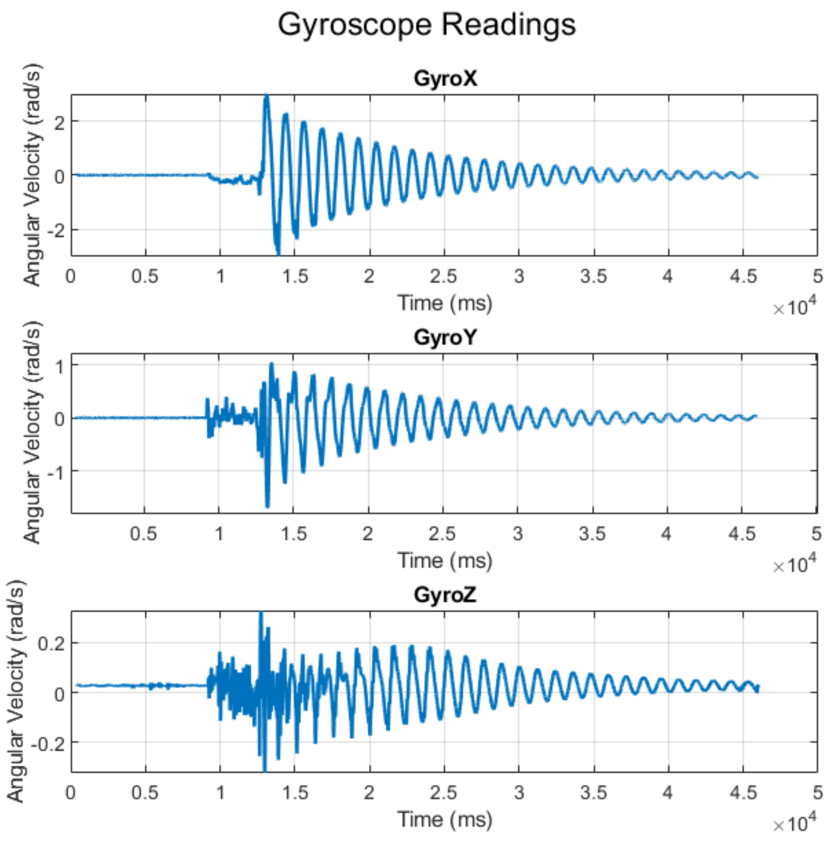

Figure 12: Raw gyroscope measurements showing tri-axial angular velocity during pendulum motion

Similar to the accelerometer data, the measured angular velocity decays over time.

4.1.3 Visual Data

Figure 13 presents the observed pendulum position in pixels over time. The data shows position in pixels along the image x and y axes as defined in Figure 6, plotted against time in seconds.

Similar to the accelerometer and gyroscope data, the observed pendulum position over time creates a damped signal that decays over time.

However, it is also evident that the visual data collection started and stopped at different times than the IMU data collection. As a result, each sensor operates within its own distinct time frame of reference, causing the time axis for the visual data to differ in length from that of the IMU data. Consequently, visual measurements taken at one moment do not align with IMU measurements taken at the same reported time.

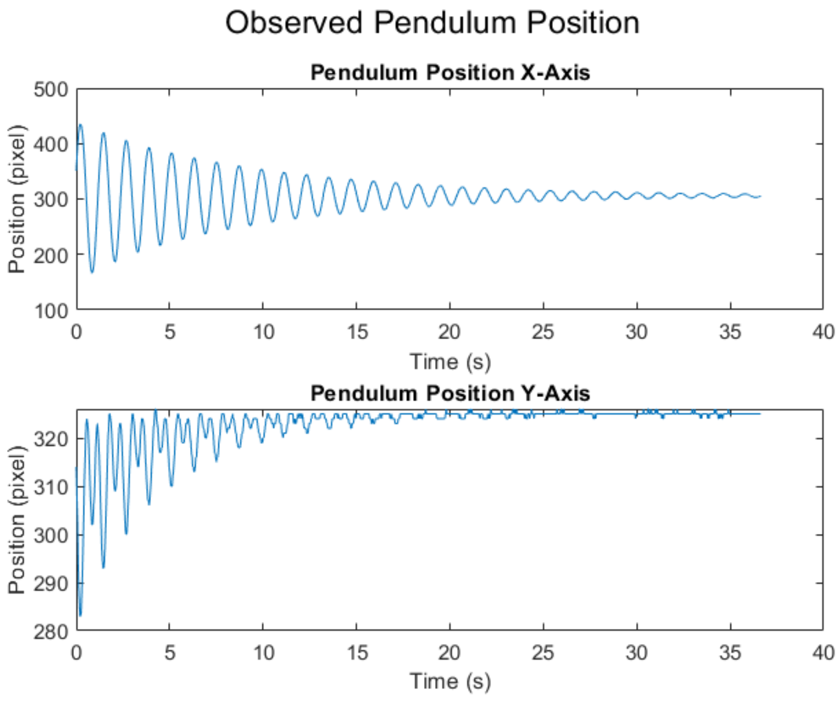

Figure 13: Raw visual position measurements during pendulum motion

4.2 Noise Characterization

In this section we investigate the noise present in the sensor readings taken from the MPU6050 over a 70-second duration while the system was in a steady state.

4.2.1 Isolating the Noise

By keeping the sensor in this static state, the sensor's output is representative of some true state that is being held constant, in addition to the noise that we are trying to analyze.

The sensor output in discrete time can then be described as the following:

$$\text{Reading}_k = \text{State}_k + \text{Noise}_k$$

Since it is known that State_k represents some constant value, and through observing the accelerometer and gyroscope readings plots (14, 15) not drifting away from some central value, it can be said that the noise appears to have a zero mean (this is further validated by inspecting a rolling mean of the data in the following section).

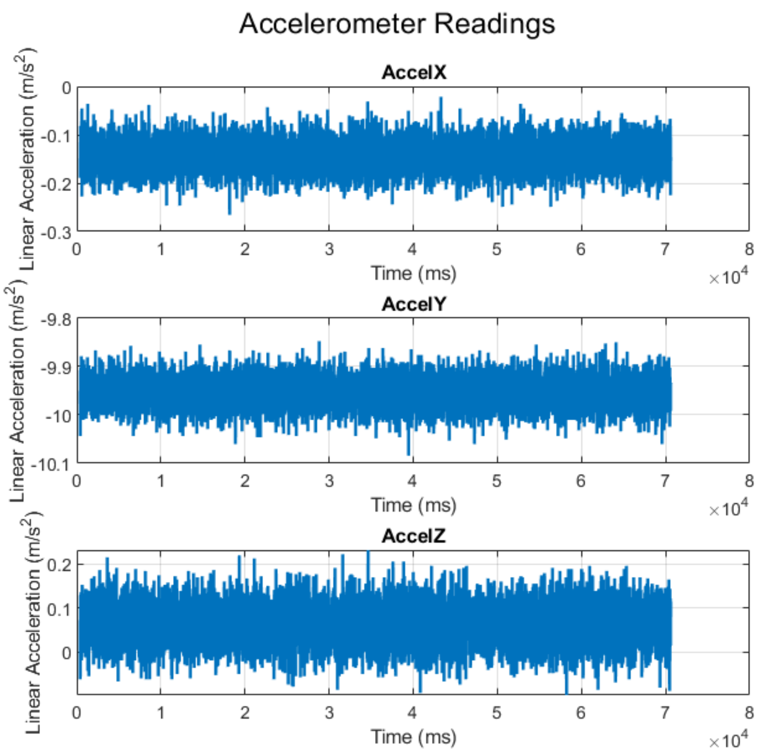

Figure 14: Accelerometer Readings from Static Data Collection

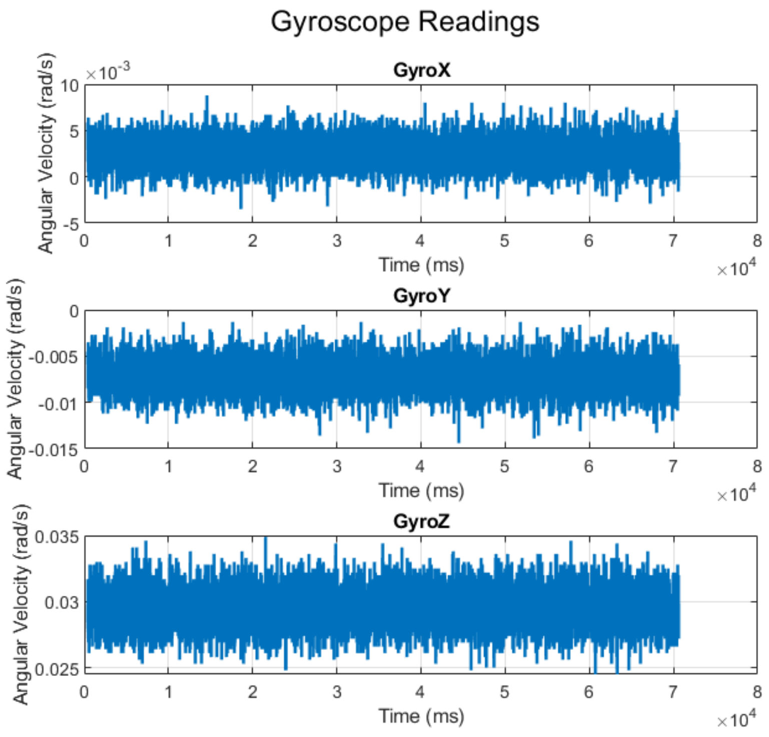

Figure 15: Gyroscope Readings from Static Data Collection

In other words, the following is true:

$$\text{State}_k = C$$

$$E[\text{Noise}_k] = 0$$

$$E[\text{Reading}_k] = E[\text{State}_k] + E[\text{Noise}_k] = C$$

Then to isolate the noise:

$$\text{Noise}_k = \text{Reading}_k - E[\text{Reading}_k] \tag{1}$$

From the results of this steady-state experiment, we obtained the accelerometer and gyroscope readings and isolated the noise as defined in equation 1. The baseline noise characteristics of the accelerometer and gyroscope sensors along the x, y, and z axes, when the sensor is stationary, are shown in Figures 16 and 17.

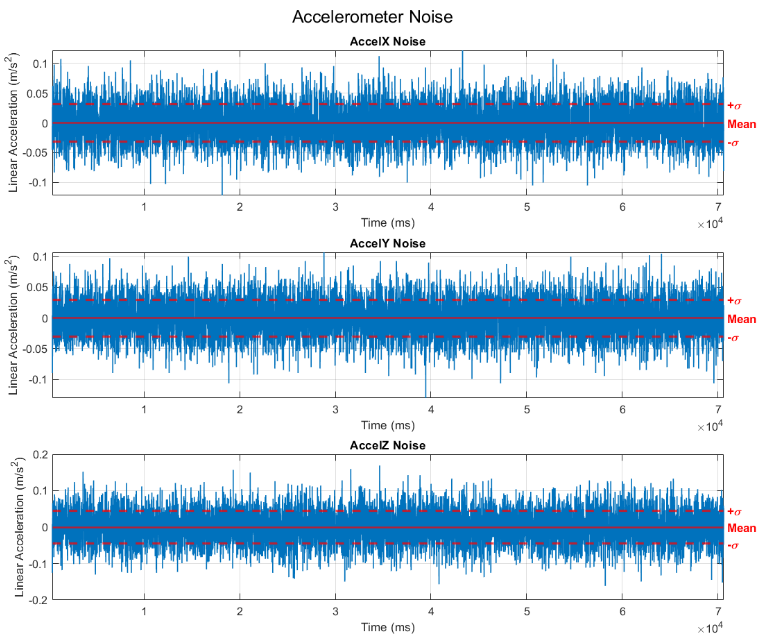

Figure 16: Accelerometer Noise Readings from Static Data Collection

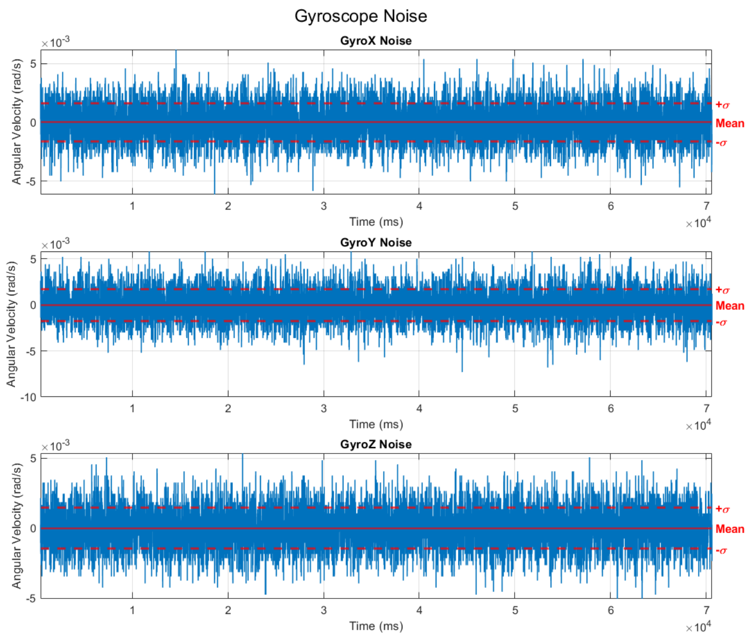

Figure 17: Gyroscope Noise Readings from Static Data Collection

Table 5: Mean and Standard Deviation for Sensor Noise

Sensor	**Mean**	**Standard Deviation**
Accelerometer X	0	0.031089
Accelerometer Y	0	0.030007
Accelerometer Z	0	0.045036
Gyroscope X	0	0.001596
Gyroscope Y	0	0.001728
Gyroscope Z	0	0.001456

4.2.2 Noise Amplitude Distribution

The isolated noise from Section 4.2.1, and plotted in Figures 16 and 17 contain the mean and standard deviation values documented in Table 5.

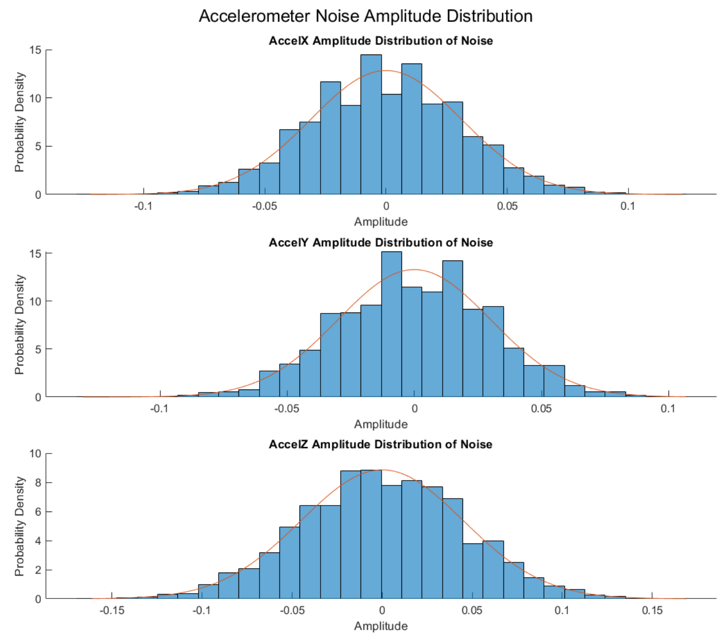

Figure 18: Accelerometer Noise Amplitude Distribution

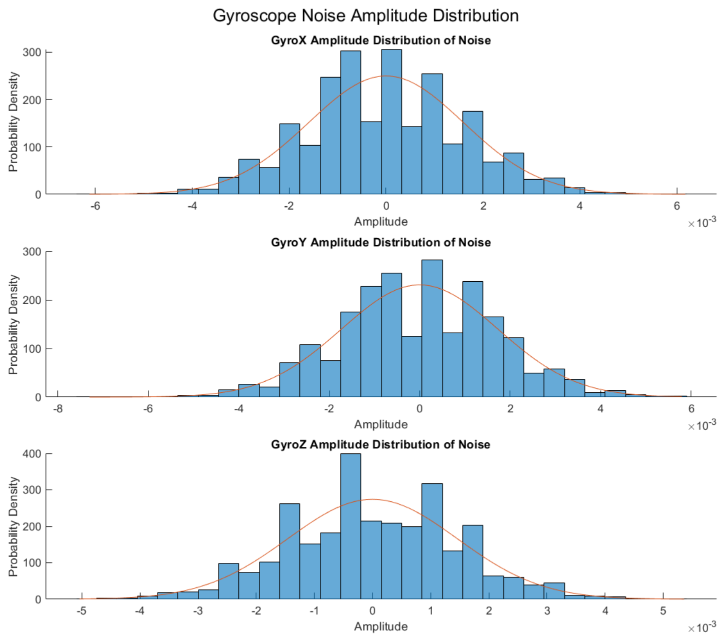

Figure 19: Gyroscope Noise Amplitude Distribution

Figure 18 and 19 show a normalized histogram of the noise amplitudes, which demonstrate that the noise amplitude is normally distributed.

4.2.3 Determining Stationarity and Ergodicity

To determine if the noise in our accelerometer and gyroscope data is stationary and ergodic, we need to evaluate if the mean and variance remain at a constant value throughout the observation period.

To achieve this, we used a sliding window approach, using MATLAB's `movmean` and `movvar` functions, to calculate the rolling mean and rolling variance over time.

By rolling mean, we mean that we applied a sliding window of approximately 0.5 seconds across the 70-seconds worth of data, calculating the average of the isolated linear acceleration and angular velocity noise data for each axis (x, y, and z) within each window. This "sliding" mean moves one data point forward each time, recalculating the mean as it goes. If the rolling mean stays consistent across all windows, it suggests that the average behavior of the data does not change over time, which is a key feature of being stationary.

Similarly, we calculated the rolling variance by applying the same sliding window technique, but instead of averaging, we assessed the variability of the data within each window. The rolling variance provides a continuous view of the data's fluctuation around the mean. If the variance also remains consistent across windows, it indicates stable variability, further supporting the stationary nature of the process.

We used a window length of 496 ms, approximating our desired window size of 500 ms. The calculation for the window length is as follows:

$$\text{window\_length} = \left\lfloor \frac{500\text{ ms}}{\Delta t} \right\rfloor = \left\lfloor \frac{500\text{ ms}}{8\text{ ms/sample}} \right\rfloor = 62\text{ samples}$$

where Δt is 8 ms, corresponding to the sampling rate as stated in section 3.1. By taking the floor of the desired window size divided by the sampling interval, we obtained a window length of 62 samples.

From the rolling mean and variance plots of the accelerometer shown in Figure 20, along with the exact means and standard deviations listed in Tables 6 and 7, we verified that the accelerometer noise is both stationary and ergodic. This conclusion is supported by the fact that both the mean and variance remain effectively constant over time, with minor standard deviations on the order of 10^{-3} for the rolling mean and 10^{-4} for the rolling variance.

Table 6 shows that the rolling mean has a standard deviation on the order of 10^{-3}. Figure 20 confirms this, illustrating that fluctuations in the rolling mean remain minimal, with the values consistently staying within this narrow range. Similarly, Table 7 demonstrates that the rolling variance has a mean and standard deviation both on the order of 10^{-4}. These consistent patterns in the rolling mean and variance over time validate the stationarity of the noise.

Furthermore, the time-averaged rolling statistics converge to their respective ensemble averages, confirming the noise process is ergodic as well as stationary. This alignment of long-term and statistical properties supports the robustness of the system's behavior.

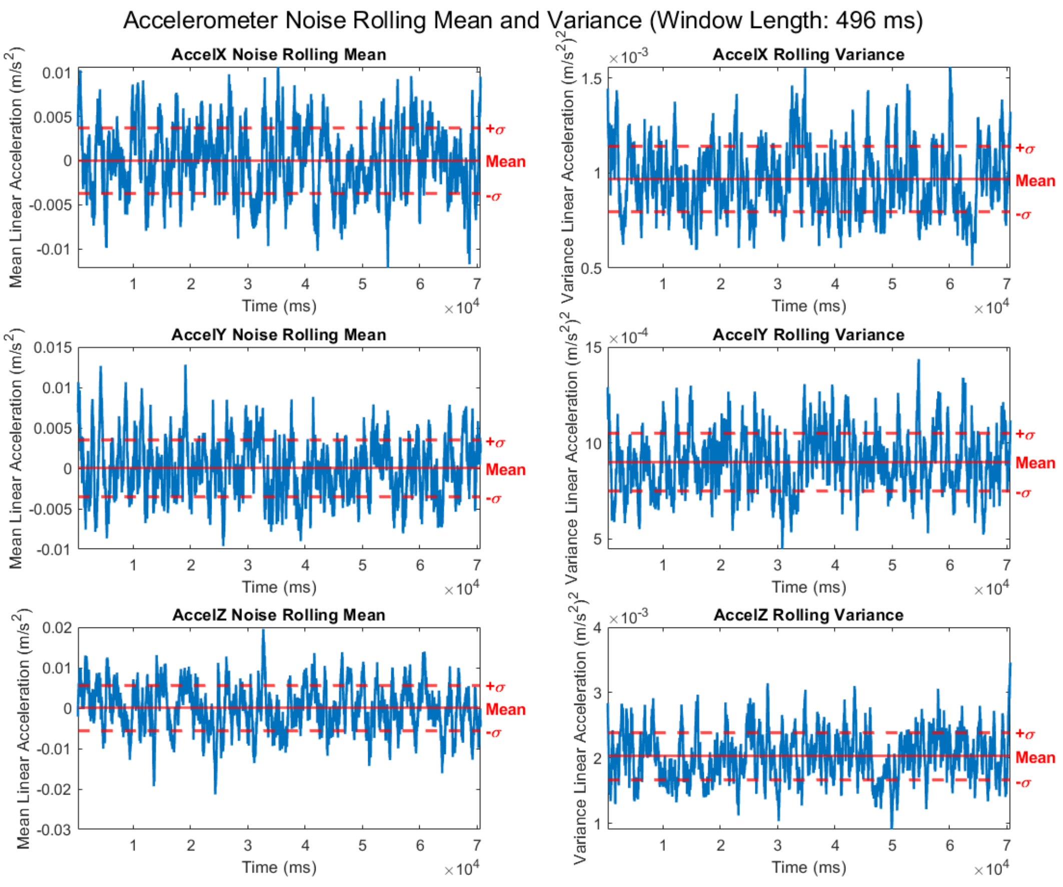

Figure 20: Accelerometer Noise Rolling Mean and Variance

From the rolling mean and variance plots of the gyroscope shown in Figure 21, along with the exact means and standard deviations listed in Tables 6 and 7, we verified that the gyroscope noise is both stationary and ergodic. This conclusion is supported by the fact that both the mean and variance remain effectively constant over time, with minor standard deviations on the order of 10^{-4} for the rolling mean and 10^{-7} for the rolling variance.

Table 7 shows that the mean variance value is stable at an order of 10^{-6}, with minor standard deviations at an order of 10^{-7}. Figure 21 confirms this, illustrating that fluctuations in the rolling variance remain within a narrow range over time. The consistent behavior of both the rolling mean and variance reinforces the conclusion that the noise is stationary.

Furthermore, the convergence of the time-averaged statistics (rolling mean and variance) to their respective ensemble averages confirms the ergodicity of the noise. This demonstrates that the long-term behavior of the gyroscope noise aligns with its statistical properties, validating both stationarity and ergodicity.

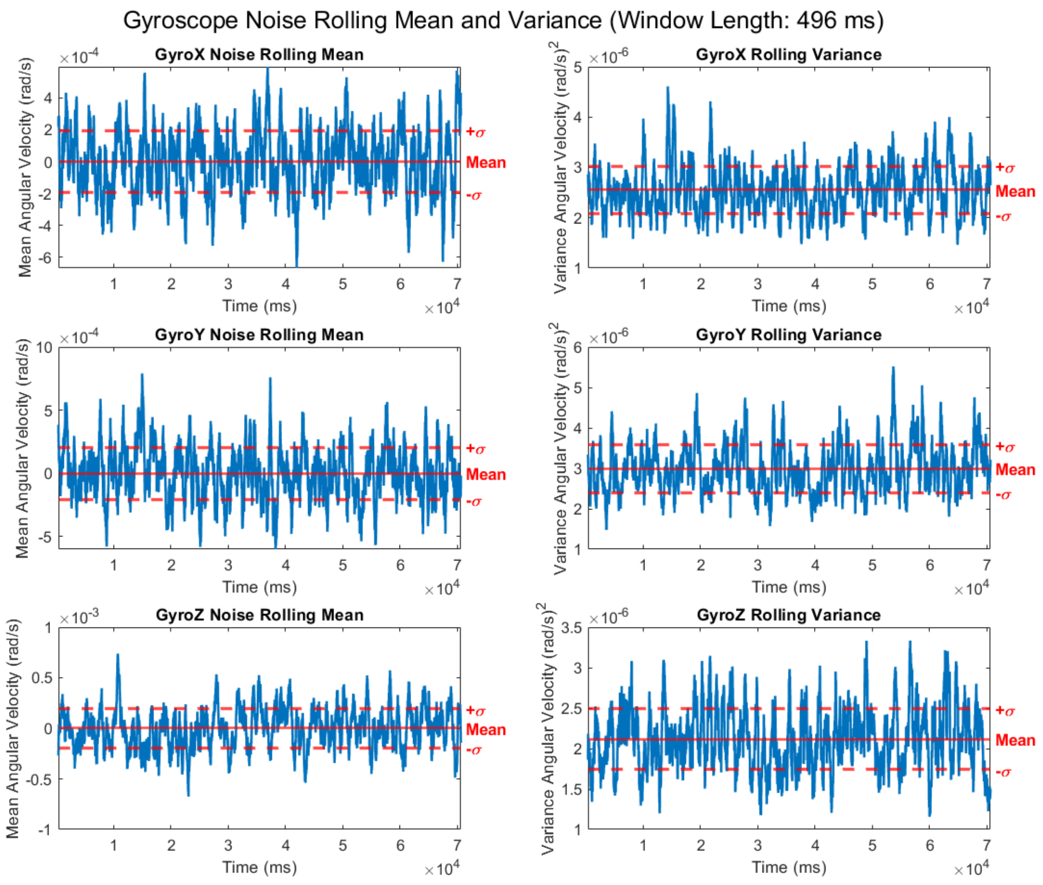

Figure 21: Gyroscope Noise Rolling Mean and Variance

Table 6: Rolling Mean Statistics

Sensor	Mean	Standard Deviation
Accelerometer X	6.569×10^{-7}	3.745×10^{-3}
Accelerometer Y	9.949×10^{-7}	3.501×10^{-3}
Accelerometer Z	7.008×10^{-6}	5.589×10^{-3}
Gyroscope X	1.444×10^{-7}	1.945×10^{-4}
Gyroscope Y	-3.080×10^{-7}	2.052×10^{-4}
Gyroscope Z	1.383×10^{-7}	1.949×10^{-4}

Table 7: Rolling Variance Statistics

Sensor	Mean	Standard Deviation
Accelerometer X	9.678×10^{-4}	1.729×10^{-4}
Accelerometer Y	9.025×10^{-4}	1.510×10^{-4}
Accelerometer Z	2.029×10^{-3}	3.637×10^{-4}
Gyroscope X	2.551×10^{-6}	4.725×10^{-7}
Gyroscope Y	2.991×10^{-6}	5.868×10^{-7}
Gyroscope Z	2.116×10^{-6}	3.777×10^{-7}

4.2.4 Analyzing Autocorrelation and Power Spectral Density

To better understand the noise characteristics of the sensor readings, we analyzed the autocorrelation and power spectral density (PSD) of the signals.

We estimated the autocorrelation of the noise signals using MATLAB's `xcorr` function with the unbiased estimation method. For both accelerometer and gyroscope noise, the autocorrelation exhibits a sharp peak at $\tau = 0$, indicating minimal correlation between values at different time lags. This behavior is consistent with the characteristics of white noise. The autocorrelation plots for the accelerometer and gyroscope are shown in Figures 22 and 23, respectively, and the mean square value of the noise is presented in Table 8.

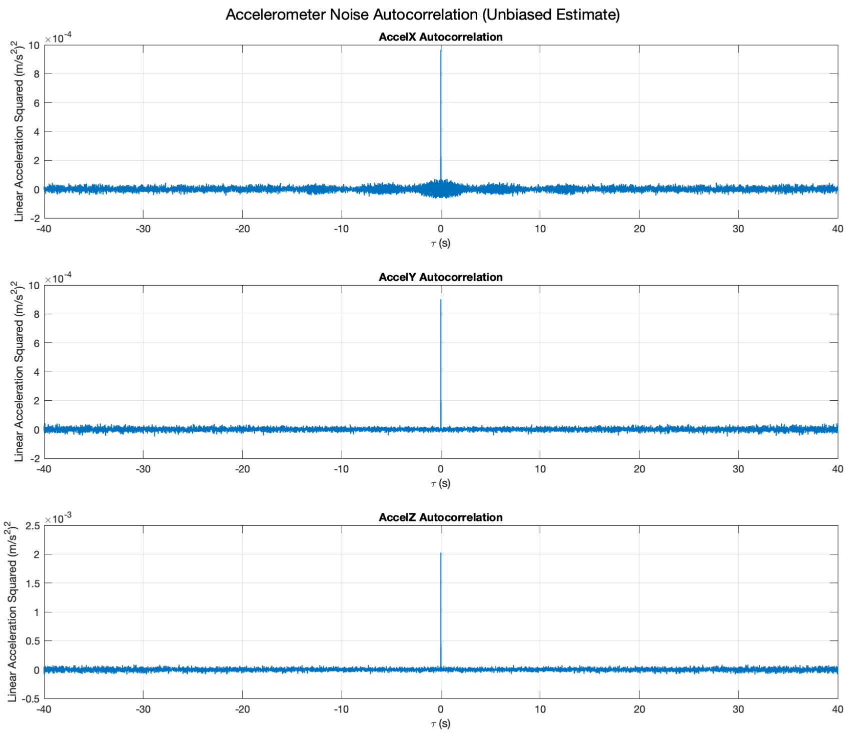

Figure 22: Autocorrelation of Accelerometer Noise (Unbiased Estimate)

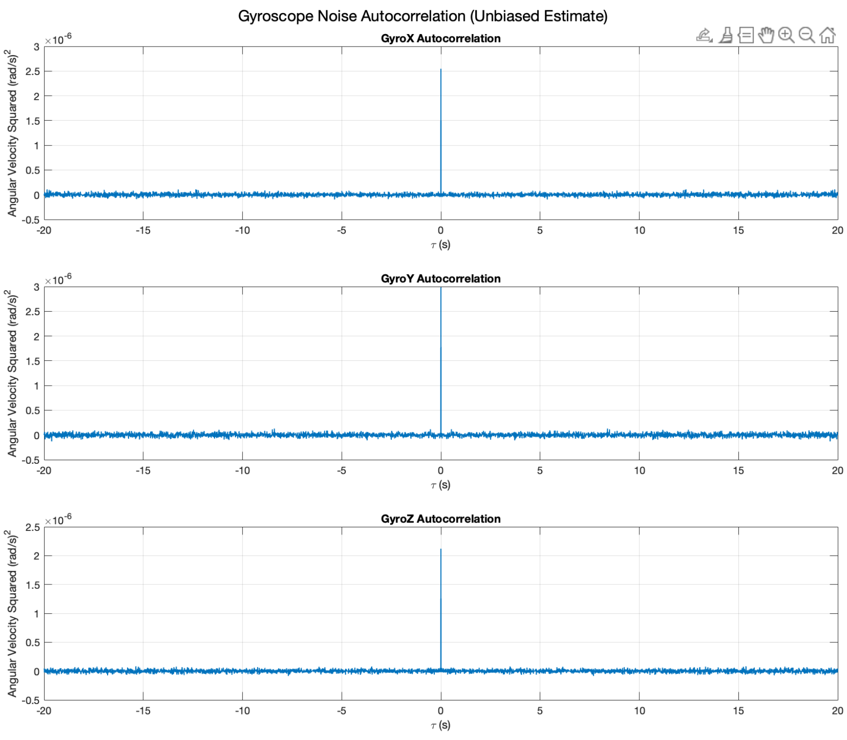

Figure 23: Autocorrelation of Gyroscope Noise (Unbiased Estimate)

Table 8: Mean Square Values for Gyroscope and Accelerometer Noise

Sensor	**Mean Square Value**
Accelerometer X	9.6641×10^{-4} (m/s^2)^2
Accelerometer Y	9.0030×10^{-4} (m/s^2)^2
Accelerometer Z	2.0000×10^{-3} (m/s^2)^2
Gyroscope X	2.5480×10^{-6} (rad/s)^2
Gyroscope Y	2.9847×10^{-6} (rad/s)^2
Gyroscope Z	2.1199×10^{-6} (rad/s)^2

For the accelerometer and gyroscope autocorrelation across all axes, as shown in Figures 22 and 23, the plots resemble a Dirac delta function with a pronounced impulse at $\tau = 0$. Beyond this central peak, the values remain close to zero across the τ-axis, indicating weak correlations at other time lags. This behavior is apparent across x, y, and z directions for each sensor.

Then, we then computed the power spectral density (PSD) using MATLAB's `periodogram` function. The PSD estimates, displayed in Figures 24 and 25, illustrate the power distribution across frequencies for the accelerometer and gyroscope noise. As expected for white noise, the PSD is nearly constant across all frequencies, confirming that the noise power is uniformly distributed. The mean of the PSD estimates are presented in Table 9.

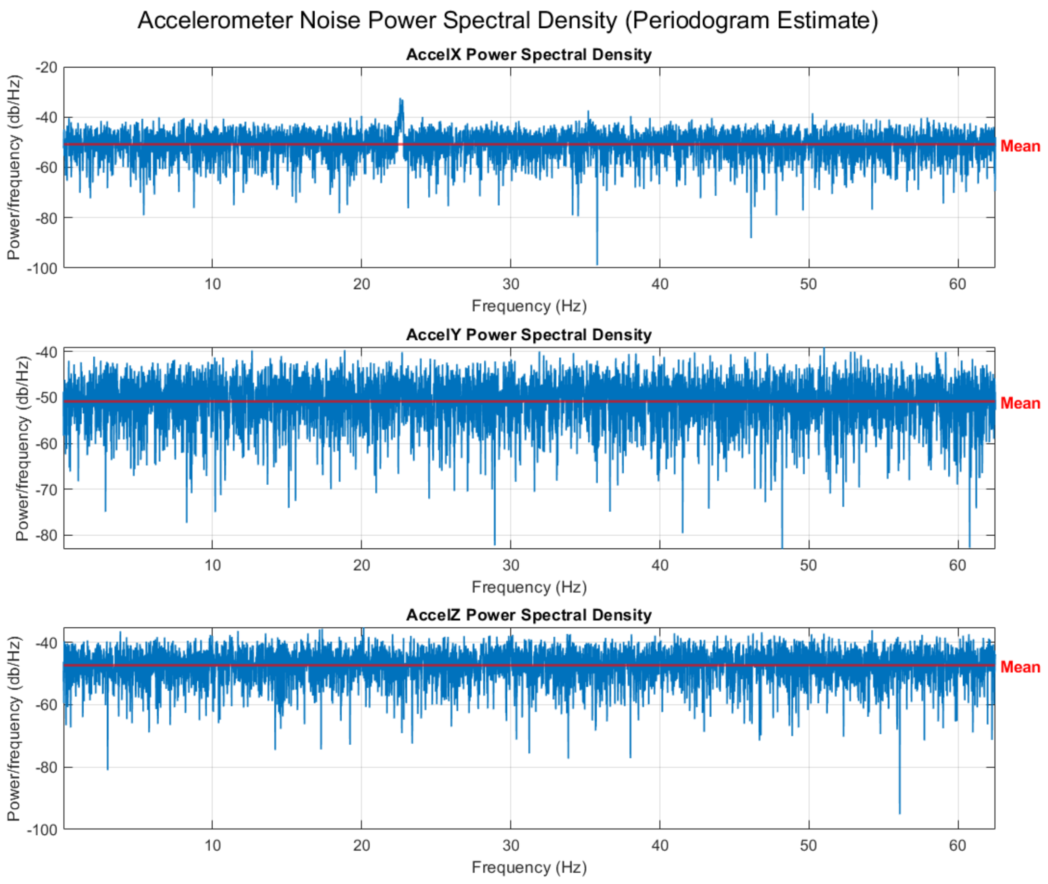

Figure 24: Power Spectral Density of Accelerometer Noise (Periodogram Estimate)

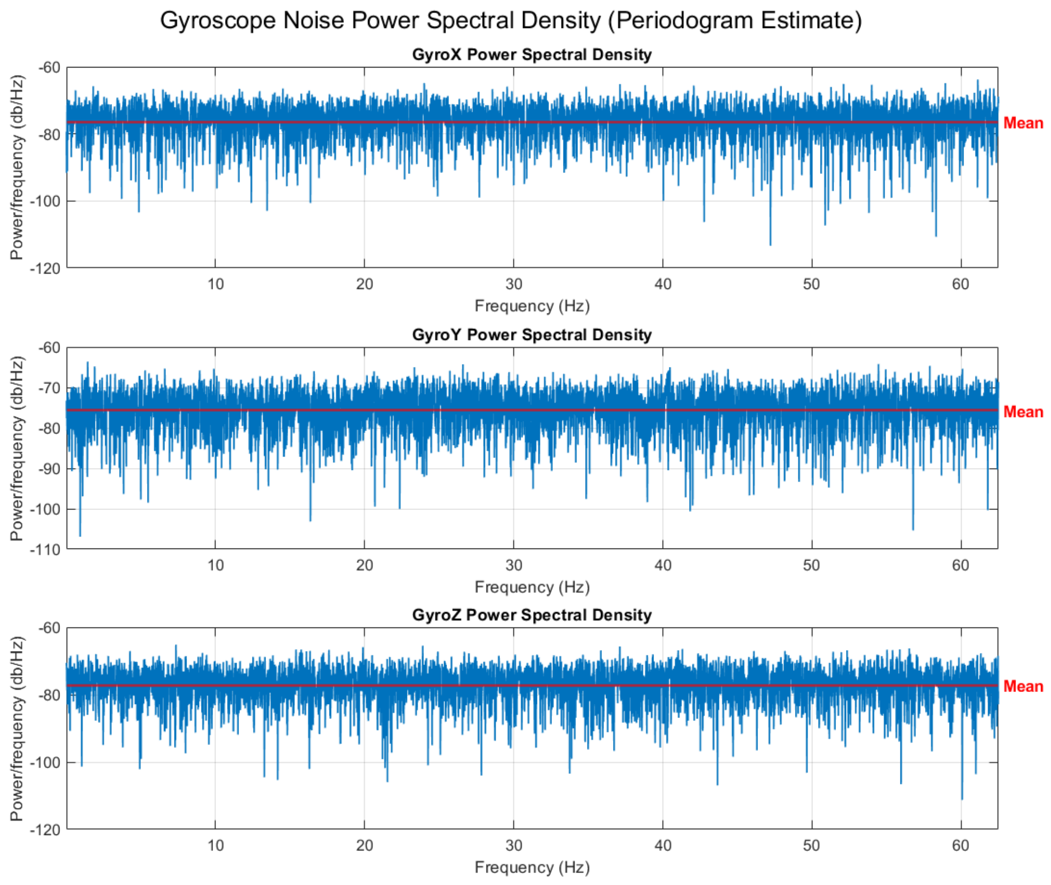

Figure 25: Power Spectral Density of Gyroscope Noise (Periodogram Estimate)

Table 9: Mean Power Spectral Density for Gyroscope and Accelerometer Noise

Sensor	**Mean Power Spectral Density**
Accelerometer X	−50.8819 (db/Hz)
Accelerometer Y	−50.9086 (db/Hz)
Accelerometer Z	−47.3922 (db/Hz)
Gyroscope X	−76.4164 (db/Hz)
Gyroscope Y	−75.6905 (db/Hz)
Gyroscope Z	−77.2178 (db/Hz)

Together, the sharp peak in the autocorrelation at zero lag and the flat power spectral density confirm that the noise in the sensor readings can be characterized as white noise.

4.2.5 Characterization

Based on the observations from the previous sections, the noise in the sensor readings can be characterized as Gaussian white noise for the following reasons:

- The noise is both stationary and ergodic, with a zero mean and constant variance for both the accelerometer and gyroscope readings.

- The power spectral density is flat and constant, indicating equal power distribution across frequencies.
- The autocorrelation function is delta-shaped, with a peak at zero lag and zero elsewhere.
- The distribution of noise amplitudes is Gaussian.

These properties collectively confirm that the noise behaves as a stochastic process with characteristics ideal for systems relying on Gaussian noise assumptions, such as Kalman filters.

5 State Space Model

5.1 Theoretical Model Derivation

Our pendulum is a 2D system where we are concerned with angular displacement θ and angular velocity $\dot{\theta}$. By working entirely in continuous time, we derive the dynamics based on the tangential linear acceleration in the z-axis, which arises due to gravity.

We first establish all the forces acting on the system. These include:

1. The drag force F_D, resulting from air resistance as the pendulum swings, introducing a damping coefficient b.
2. The tangential acceleration a_z, which is given by $a_z = g\sin(\theta)$, where g is the gravitational acceleration.

A few helpful symbols prior to derivation that will be our theoretical models and then simulated/fitted accordingly to the observed are the following:

- g - gravitational constant
- b - damping coefficient
- R - length of the pendulum string
- m - mass of the pendulum object

To model the system, we apply Newton's second law to describe the forces in the tangential z direction:

$$F_z = ma_z = m\ddot{z},$$

where m is the pendulum's mass and $\ddot{z}$ is its tangential acceleration.

The net force F_z is the sum of all forces acting along the z-direction. This yields:

$$m\ddot{z} = -mg\sin(\theta) - bR\dot{\theta},$$

where R is the pendulum's length.

To express $\ddot{z}$ in terms of angular variables, we leverage the relationship between the arc length z and the angular displacement θ:

$$z = R\theta, \quad \dot{z} = R\dot{\theta}, \quad \ddot{z} = R\ddot{\theta}.$$

Substituting $\ddot{z} = R\ddot{\theta}$ into the force equation, we obtain:

$$mR\ddot{\theta} = -mg\sin(\theta) - bR\dot{\theta}.$$

Simplifying for $\ddot{\theta}$:

$$\ddot{\theta} = -\frac{g\sin(\theta)}{R} - \frac{b}{m}\dot{\theta}.$$

To linearize the system (required for state-space modeling), we apply the small-angle approximation $\sin(\theta) \approx \theta$. This gives:

$$\ddot{\theta} = -\frac{g}{R}\theta - \frac{b}{m}\dot{\theta}.$$

With this, we can now construct the continuous-time state-space model. Defining the state vector, input, and output as:

$$\vec{x} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}, \quad \dot{\vec{x}} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix}, \quad y = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix},$$

we express the system dynamics as:

$$\dot{\vec{x}} = \mathbf{A}_c \vec{x} + \mathbf{B}_c u,$$
$$y = \mathbf{C}_c \vec{x} + \mathbf{D}_c u.$$

For our pendulum, the input u is zero since the motion is driven entirely by its initial conditions and gravitational forces. Thus:

$$\mathbf{B}_c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{D}_c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The system reduces to:

$$\dot{\vec{x}} = \mathbf{A}_c \vec{x}, \quad y = \mathbf{C}_c \vec{x}.$$

Using the linearized differential equation for $\ddot{\theta}$, the system matrix A_c becomes:

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 \\ -\frac{g}{R} & -\frac{b}{m} \end{bmatrix}.$$

Since the output y directly corresponds to the state vector $\vec{x}$, the output matrix C_c is the identity matrix:

$$\mathbf{C}_c = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

5.2 Theoretical Model Analysis

5.2.1 Linearity and Time-Invariance

Regarding the linearity and time-invariance of the system, the original dynamics were nonlinear due to the presence of trigonometric terms like $\sin(\theta)$ and $\cos(\theta)$. However, by employing the small-angle approximation, we successfully linearized the system. The main constraint on our system is the range of angular displacement, which we constrained experimentally to approximately $\pm 20°$ to ensure the validity of the small-angle approximation.

Additionally, the system is time-invariant, as time does not explicitly appear in the differential equations governing the dynamics. Regardless of when the pendulum motion begins, the outputs θ (angular displacement) and $\dot{\theta}$ (angular velocity) remain unaffected by any time shift, confirming time-invariance.

5.2.2 States

The system's states (the values which we aim to estimate) are defined as the angular displacement θ and angular velocity $\dot{\theta}$, which together form the state vector $\vec{x}$:

$$\vec{x} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

5.2.3 Inputs

This is an unforced system with no external control inputs. The system's motion is governed solely by its internal dynamics, which include:

- Gravitational torque
- Damping torque due to friction at the pivot
- Initial conditions (θ_0, $\dot{\theta}_0$)

5.2.4 Outputs

The outputs are measurable quantities that can be observed from the system. In this case, we are interested in observing both states:

$$\vec{y} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} = \vec{x}$$

This results in an identity output matrix $\mathbf{C}_c$ in our state-space representation, where the outputs directly correspond to the states.

5.2.5 Discrete vs Continuous-Time

The system was initially modeled in continuous time using state-space equations, yielding:

$$\dot{\vec{x}} = \mathbf{A}_c \vec{x}, \quad y = \mathbf{C}_c \vec{x}.$$

To discretize the system, we used the matrix exponential approach with a theoretical sampling rate denoted T_s. The discrete-time state-space matrix $\mathbf{A}$ was computed in MATLAB as:

$$\mathbf{A} = e^{\mathbf{A}_c T_s}$$

The output matrix $\mathbf{C}$ remains the same as the identity matrix during discretization, as the outputs directly correspond to the states. The input matrix $\mathbf{B}$ and feedthrough matrix $\mathbf{D}$ are excluded, as the system's motion is solely driven by its internal dynamics.

5.2.6 Stability

To assess system stability, we first analyze the continuous-time state matrix $\mathbf{A}_c$:

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 \\ -\frac{g}{R} & -\frac{b}{m} \end{bmatrix}$$

The characteristic equation is found by:

$$0 = |s\mathbf{I}_2 - \mathbf{A}_c|$$

Expanding the determinant:

$$0 = \begin{vmatrix} s & -1 \\ \frac{g}{R} & s + \frac{b}{m} \end{vmatrix}$$

Which yields the characteristic polynomial:

$$s^2 + \frac{b}{m}s + \frac{g}{R} = 0$$

Applying the Routh-Hurwitz Criterion [7], we observe that:

1. All coefficients are positive since $b, m, g, R > 0$
2. The coefficients appear in ascending order of s

Therefore, all roots have negative real parts, confirming the continuous-time system is strictly stable. Since stability in continuous-time implies stability in discrete-time, we can conclude that our discretized implementation is also stable, with eigenvalues lying within the unit circle.

5.2.7 Observability

We constructed the observability matrix using the discrete-time matrices $\mathbf{A}$ and $\mathbf{C}$. The observability matrix, denoted as $\mathcal{O}$, is given by:

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \end{bmatrix}$$

We also computed the discrete observability Gramian, denoted as $\mathcal{G}$ with the following equation:

$$\mathcal{G} = \mathcal{O}^T \mathcal{O}$$

where T represents taking the transpose of the matrix $\mathcal{O}$.

By evaluating the rank of the observability matrix $\mathcal{O}$ in MATLAB, we determined whether the system is fully observable. We confirmed that $\mathcal{O}$ is full rank, ensuring that the system is fully observable. Additionally, the Gramian $\mathcal{G}$ was used to further analyze the system's observability properties, validating our findings.

The sampling rate significantly impacts the system's observability. Lowering the sampling rate reduces the number of measurements, increasing the likelihood of missing critical events, thereby degrading observability. Conversely, increasing the sampling rate enhances observability by capturing more data, but this comes at the cost of additional computational and storage overhead.

Observability can be improved by redefining the system states using a transformation matrix $\mathbf{T}$. Specifically, we define a new state vector $\vec{Z}$ as:

$$\vec{Z} = \mathbf{T}\vec{x}$$

where $\vec{Z}$ represents the transformed states, designed to make the system more observable.

To find the transformation matrix $\mathbf{T}$, we normalize the observability Gramian $\mathcal{G}$ (associated with the original state $\vec{x}$) to become the identity matrix $\mathcal{G}_Z$ (associated with the new state $\vec{Z}$). This normalization ensures that the transformed system is fully observable.

The relationships between the transformed state-space matrices and the original ones are as follows:

$$\begin{aligned}\mathbf{A}_Z &= \mathbf{TAT}^{-1}\\ \mathbf{B}_Z &= \mathbf{TB}\\ \mathbf{C}_Z &= \mathbf{CT}^{-1}\\ \mathbf{D}_Z &= \mathbf{D}\end{aligned}$$

In our case, both $\mathbf{B}$ and $\mathbf{D}$ are zero, so their transformed counterparts $\mathbf{B}_Z$ and $\mathbf{D}_Z$ are also zero. Additionally, since $\mathbf{C}$ is the identity matrix, the transformation simplifies to:

$$\mathbf{C}_Z = \mathbf{T}^{-1}$$

The observability Gramian for the transformed state is defined as:

$$\mathcal{G}_Z = \sum_{m=0}^{n-1} \left(\mathbf{A}_Z^T\right)^m \mathbf{C}_p^T \mathbf{C}_Z \mathbf{A}_Z^m$$

which, under a similarity transformation, becomes:

$$\mathcal{G}_Z = \sum_{m=0}^{n-1} \left((\mathbf{TAT}^{-1})^T\right)^m \mathbf{T}^{-T}\mathbf{T}^{-1}(\mathbf{TAT}^{-1})^m$$

Simplifying using the transpose property $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$, we get:

$$\mathcal{G}_Z = \mathbf{T}^{-T}\left(\sum_{m=0}^{n-1}(\mathbf{A}^T)^m\mathbf{C}^T\mathbf{C}\mathbf{A}^m\right)\mathbf{T}^{-1}$$

Factoring out $\mathbf{T}^{-T}$ and $\mathbf{T}^{-1}$, we arrive at:

$$\mathcal{G}_Z = \mathbf{T}^{-T}\mathcal{G}\mathbf{T}^{-1}$$

where $\mathcal{G}$ is the observability Gramian in the original state.

To normalize $\mathcal{G}_Z$ to the identity matrix $\mathbf{I}$, we perform the spectral decomposition of $\mathcal{G}$:

$$\mathcal{G} = \mathbf{V\Lambda V}^T$$

where $\mathbf{V}$ contains the eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. Normalizing $\mathcal{G}_Z$ implies:

$$\mathbf{I} = \mathbf{T}^{-T}\mathcal{G}\mathbf{T}^{-1}$$

This leads to the following condition for $\mathbf{T}$:

$$\mathbf{TT}^T = \mathbf{V\Lambda}^{1/2}\mathbf{\Lambda}^{1/2}\mathbf{V}^T = \mathbf{V\Lambda V}^T$$

The transformation matrix $\mathbf{T}$ is therefore computed as:

$$\mathbf{T} = \mathbf{\Lambda}^{1/2}\mathbf{V}^T$$

Using $\mathbf{T}$, we can now compute the transformed matrices $\mathbf{A}_Z$ and $\mathbf{C}_Z$, which improve the system's observability.

5.3 Fitting Model to Visual Data

The state-space model is defined by four parameters, three of which are specified in the experimental procedure:

- $g = 9.80665$ m/s^2
- $R = 0.4064$ m
- $m = 0.073$ kg

The fourth parameter, the damping coefficient b, is challenging to determine theoretically. An initial guess of $b = 0.02$ was used. The following section describes the approach used to refine this parameter based on the experimental data.

5.3.1 Visual Data Processing

In the experimental procedure, a vision system was used to track the pendulum's position over time. This data serves as ground truth for state estimation, helping to validate the alignment between the theoretical model and experimental observations.

Although the vision system is not perfect and introduces its own noise and quantization errors due to the use of integer image coordinates, it provides a direct observation of the pendulum's position. This data is sufficiently accurate for the purpose of comparison with the IMU-based state estimation.

The vision data provides the pixel coordinates of the pendulum bob over time and an estimated position for the top of the pendulum. From this information, a vector of the pendulum's motion can be constructed, and the angle θ relative to the Y-axis can be calculated:

$$\vec{v} = \begin{bmatrix} \text{posX} \\ \text{posY} \end{bmatrix} - \begin{bmatrix} \text{clickPosX} \\ \text{clickPosY} \end{bmatrix}, \quad \theta = \text{atan2}(v_1, v_2), \quad \theta = \theta - \text{mean}(\theta)$$

Subtracting the mean ensures that the angle θ approaches zero when the pendulum reaches steady-state, accounting for inaccuracies in the user-provided pendulum top location. The angular velocity ω is then approximated using numerical differentiation:

$$\omega = \frac{d\theta}{dt} \approx \frac{\sum_i^{n-1} (\theta_{i+1} - \theta_i)}{dt}, \quad dt = \frac{1}{\text{FPS}}$$

Figure 26 shows the computed angle and angular velocity from the vision system.

5.3.2 Theoretical Model Simulation

The theoretical model was simulated using the parameters defined in the experimental procedure and an "eyeball approximation" of b. The initial state was derived from the visual data. Figure 27 shows the resulting simulation.

At first glance, the simulated model appears to closely match the experimental data, as shown in Figure 28. However, the squared error between the observed and simulated θ was calculated to be 15.2910, likely due to small inaccuracies in the experimental measurements.

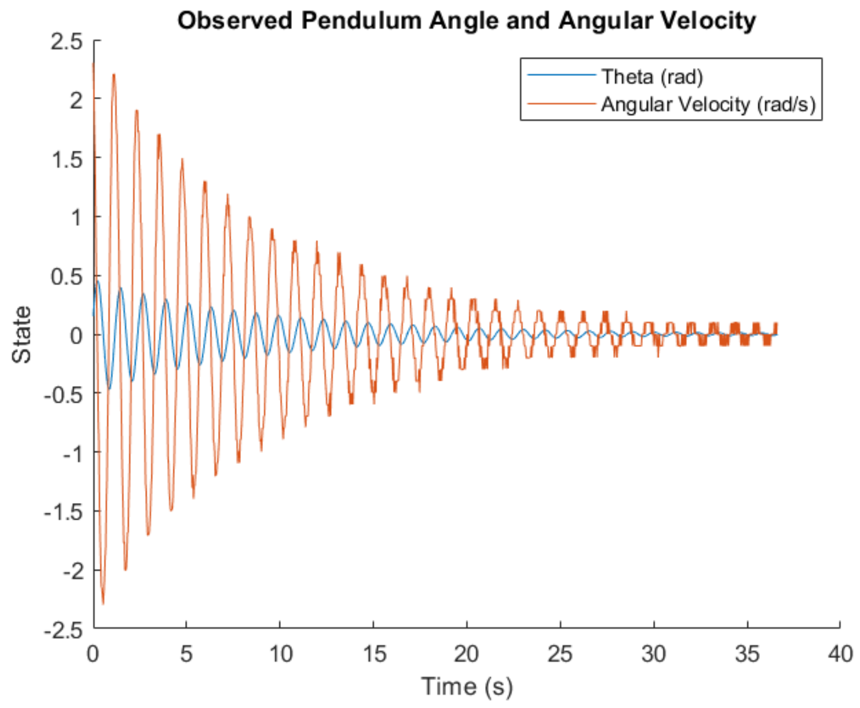

Figure 26: Observed pendulum angle and angular velocity from vision data.

5.3.3 Optimizing Model Parameters

To refine the model, a parameter optimization was performed to minimize the squared error between observed and simulated θ. The objective function is defined as:

$$\text{Squared Error} = \sum_{i}^{n} (\theta_{\text{observed},i} - \theta_{\text{simulated},i})^2$$

MATLAB's `fminsearch` function was used to optimize the R, m, and b parameters. The optimized values were:

$$R = 0.3628\,\text{m}, \quad m = 0.0837\,\text{kg}, \quad b = 0.0190$$

The resulting squared error was reduced to 0.5014. Figure 29 demonstrates the improved alignment between the model and experimental data.

The optimized model closely matches the pendulum's dynamics. However, it maintains a consistent period of oscillation, while the experimental data shows slight variations, likely caused by the pendulum swinging with a USB cable, introducing unmodeled nonlinear dynamics.

5.3.4 Optimized Model Parameters Discussion

Initially, the optimization process focused solely on estimating the unknown parameter, b, as it was challenging to observe directly. However, it quickly became apparent that the period of oscillation predicted by the theoretical model, based on the experimentally

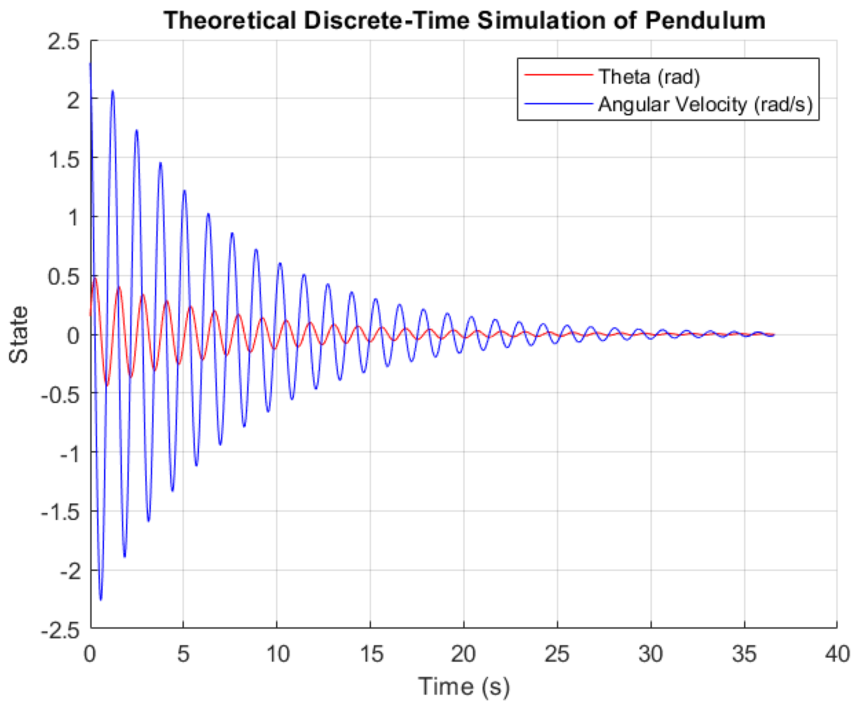

Figure 27: Theoretical pendulum dynamics with initial parameter guess

determined R and m, did not align well with the observed period from the experimental data. This discrepancy prompted us to also optimize R (pendulum length) and m (pendulum mass), in addition to b.

The optimized parameters differ slightly from the experimentally measured values:

- The pendulum length R was reduced by 4.36cm. This discrepancy is attributed to inaccuracies in estimating the pendulum's top location and the effect of the USB cable restricting motion.
- The mass m increased slightly, likely due to the additional weight of vision tracking equipment, jumper cables, or the USB cable.

The gravity constant (g) was not included in the optimization because it is well-established and does not vary in the experimental setup.

Remarkably, the initial guess for the damping coefficient b (0.02) was close to the optimized value (0.0190). These results suggest that the optimized parameters provide a realistic representation of the system.

5.4 State Observability Ellipse

Using the theoretical model parameters listed in Section 5.3 and the derivations outlined in Section 5.2.7, the state observability ellipse can be computed and visualized. The resulting ellipse is shown in Figure 30.

As depicted in Figure 30, the angular velocity $\dot{\theta}$ is more observable than the angular displacement θ. This is evident from the direction of the most observable vector, which is aligned more closely with $\dot{\theta}$, while the least observable vector is oriented along θ.

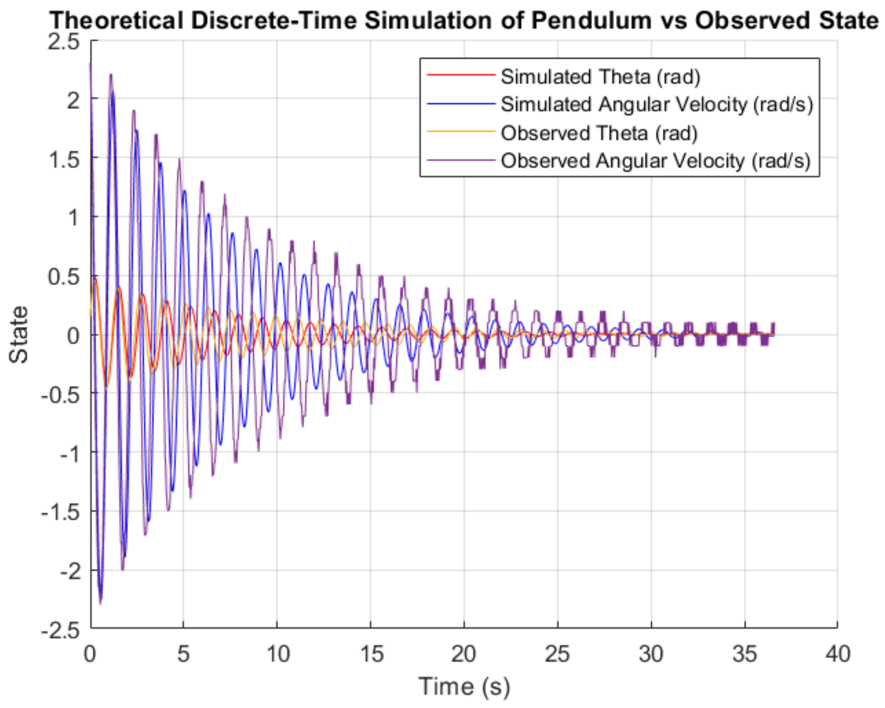

Figure 28: Comparison of observed and simulated states

To further analyze the system, a transformation to a new state space, denoted as $\vec{Z}$, can be performed, in which both states are equally observable. The observability ellipsoid for the transformed state space is shown in Figure 31.

In this transformed space, the observability ellipsoid is now represented as a unit circle, indicating that both θ and $\dot{\theta}$ are equally observable.

The transformation matrix used to achieve this is:

$$\mathbf{T} = \begin{bmatrix} -0.6636 & -1.0138 \\ -1.4726 & 0.9639 \end{bmatrix}$$

To verify that the optimal model parameters, as derived in Section 5.3, maintain the same level of observability as the theoretical model parameters, the observability ellipse is computed using the optimal parameters. The resulting ellipse is shown in Figure 32.

As shown in Figure 32, the observability ellipses generated using both the theoretical and optimal model parameters are identical. This indicates that the optimal parameters do not alter the system's observability, confirming that the system's observability remains consistent across the two sets of parameters.

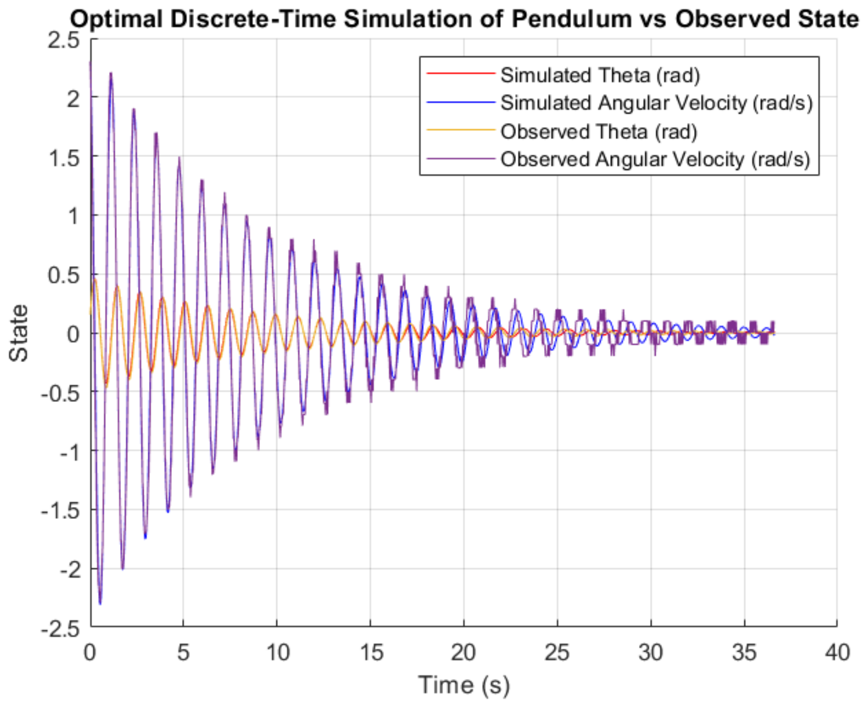

Figure 29: Comparison of observed and optimized theoretical model states

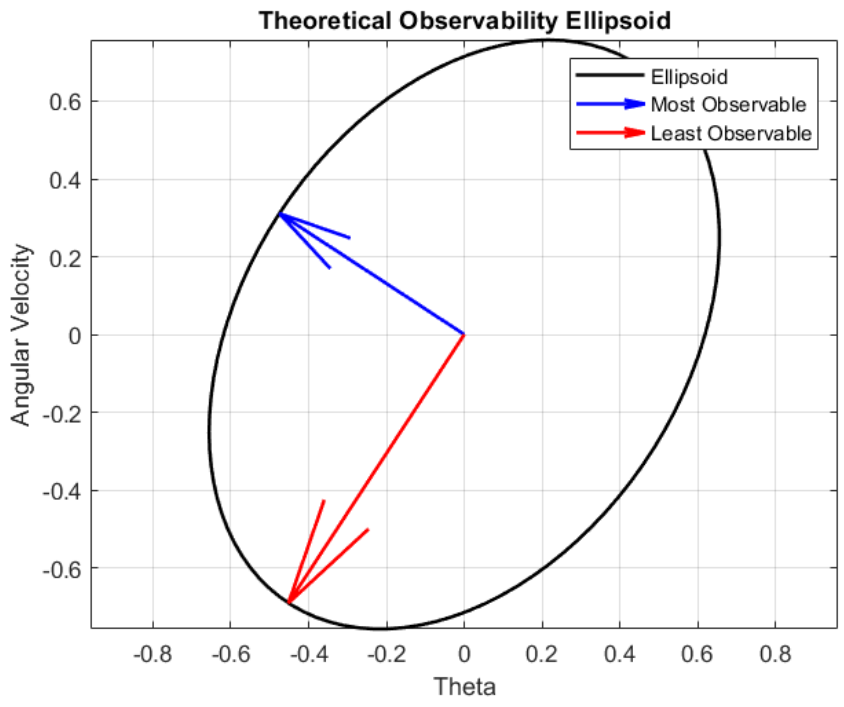

Figure 30: State Model Observability Ellipsoid Generated with Theoretical Parameters

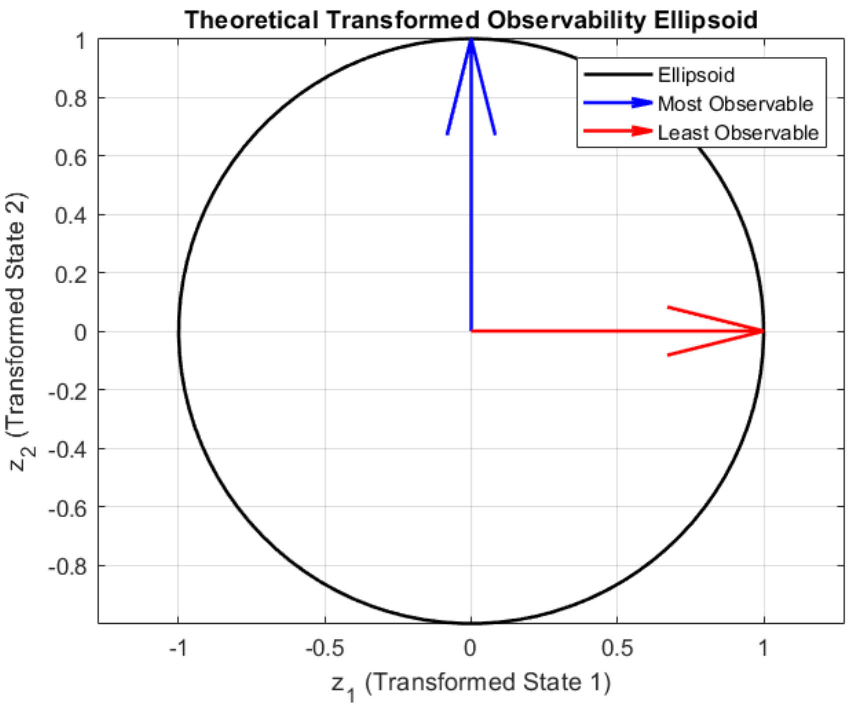

Figure 31: Transformed State Model Observability Ellipsoid

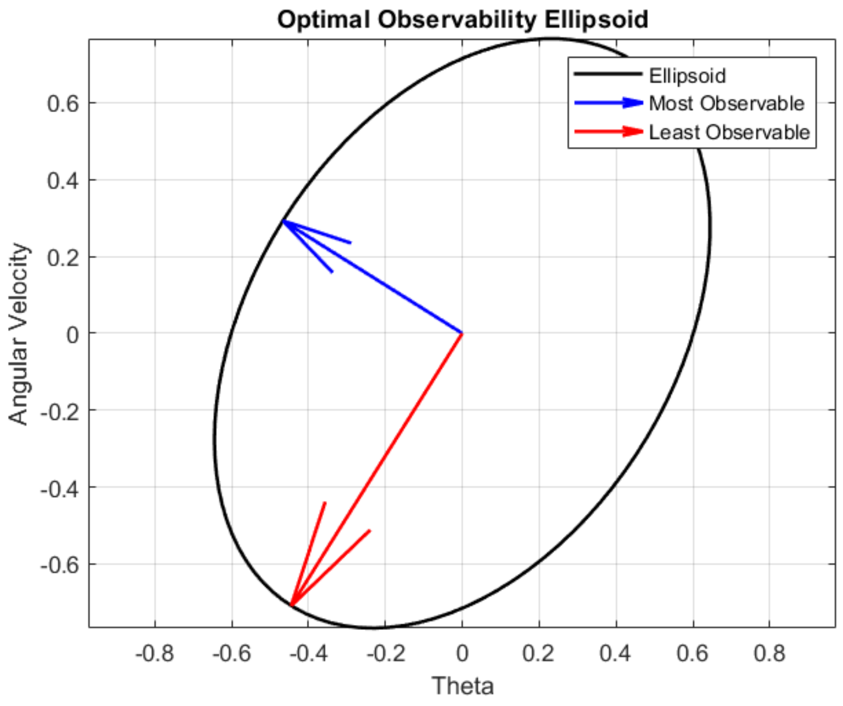

Figure 32: State Model Observability Ellipsoid Generated with Optimal Parameters

6 State Estimation

6.1 Kalman Filter

This section details the development of two Kalman filters that can estimate the pendulum's position from the data discussed in section 4.1 using the pendulum motion model and sensor fusion, respectively.

6.1.1 Position Estimation using Pendulum Motion Model

To create the process model for the Kalman filter, we use the discretized pendulum motion model as described in section 5.1 and add white Gaussian noise.

$$\vec{x}_{k+1} = \mathbf{A}\vec{x}_k + \vec{w}_k$$

where:

$$\mathbf{A} = e^{\mathbf{A}_c T_s} \text{ and } \mathbf{A}_c = \begin{bmatrix} 0 & 1 \\ \frac{-g}{r} & \frac{-b}{m} \end{bmatrix} \text{ (section 5.3)}$$

$$\vec{x}_k = \begin{bmatrix} \theta_k \\ \dot{\theta}_k \end{bmatrix}$$

$$\vec{w}_k = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix},\ w_1 \text{ and } w_2 \sim N(0, \sigma^2_{process}),\ \sigma^2_{process} = 1 \text{ rad}^2 \text{ and } 1 \frac{\text{rad}^2}{\text{s}^s} \text{ (manually tuned)}$$

Note that $\sigma^2_{process}$ is a manually tuned parameter, and the optimal model parameters found in section 5.3 are used to populate $\mathbf{A}_c$.

To create the measurement model for the Kalman filter, we use only gyroscope data as input.

$$z_k = \mathbf{C}\vec{x}_k + v_k$$

where:

$$\mathbf{C} = [0, 1]$$

$$z_k = \dot{\theta}_{gyro_k}$$

$$v_k \sim N(0, \sigma^2_{gyro}),\ \sigma^2_{gyro} = 2.5483 \times 10^{-6} \ \frac{\text{rad}^2}{\text{s}^2} \text{ (section 4.2)}$$

Note that angular velocity measurements from the X axis of the gyroscope ($\dot{\theta}_{gyro_k}$) is being used as it observes the plane of pendulum motion.

Next, we define the Kalman filter the measurement and process noise covariance matrix.

$$\mathbf{Q} = E[\vec{w}_k \vec{w}_k^T] = \begin{bmatrix} E[w_{1_k}^2] & E[w_{1_k} w_{2_k}] \\ E[w_{1_k} w_{2_k}] & E[w_{2_k}^2] \end{bmatrix} = \mathbf{I}_2$$

$$\mathbf{R} = E[v_k v_k^T] = E[v_k^2] = 2.5483 \times 10^{-6} \ \frac{\text{rad}^2}{\text{s}^2}$$

where:

$$Var[w_{1_k}] = E[w_{1_k}^2] - E[w_{1_k}]^2 = E[w_{1_k}^2] = 1 \text{ rad}^2$$

$$E[w_{1_k} w_{2_k}] = E[w_{1_k}]E[w_2] = 0$$

$$Var[w_{2_k}] = E[w_{2_k}^2] - E[w_{2_k}]^2 = E[w_{2_k}^2] = 1 \ \frac{\text{rad}^2}{\text{s}^2}$$

and:

$$Var[v_k] = E[v_k^2] - E[v_k]^2 = E[v_k^2] = 2.5483 \times 10^{-6} \ \frac{\text{rad}^2}{\text{s}^2}$$

Finally, to fully define the Kalman filter we define the priori state estimate and priori error covariance matrix.

$$\hat{\vec{x}}_0^- = 0_{2x1}$$

$$\mathbf{P}_0^- = \mathbf{I}_2 \text{ (manually tuned)}$$

The priori state estimate is defined as a 2x1 zero vector as this represents the pendulum at rest, and the priori error covariance matrix was manually tuned to the identity matrix which has provided acceptable results.

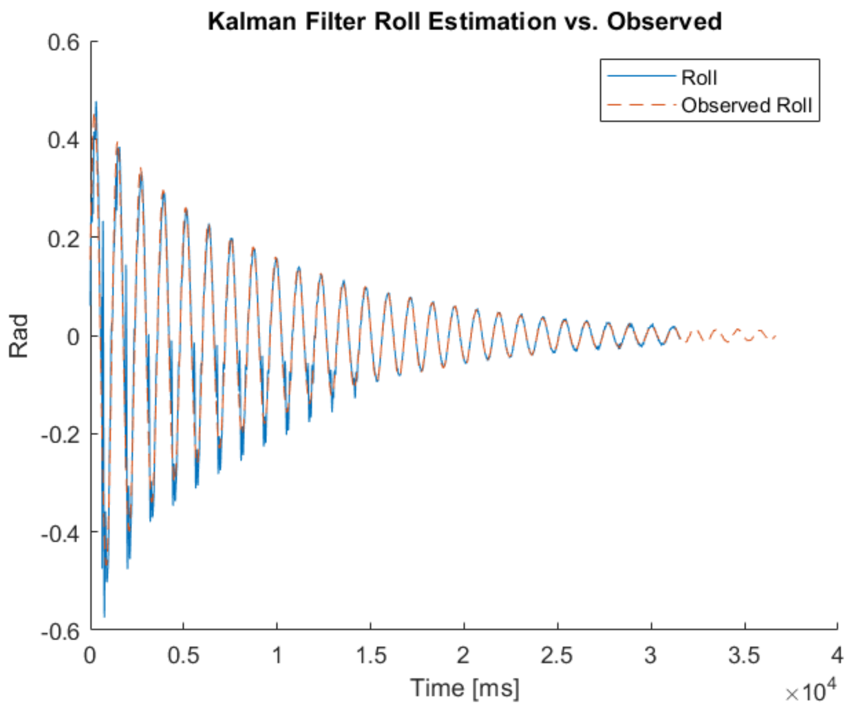

Figure 33: Roll Estimation using Pendulum Motion Model

As seen in Figure 33 through providing raw gyroscope measurements the Kalman filter is able to estimate the position using the pendulum motion model.

6.1.2 Position Estimation using Sensor Fusion

The Kalman filter in section 6.1.1 is able to estimate the position of the pendulum, but it can be greatly improved through using both the gyroscope and accelerometer measurements. With the inclusion of accelerometer data, we no longer have to rely on the pendulum motion model to estimate the current position but can compute it directly as the integration of the gyroscope's angular velocity measurements over time.

First, we define the gyroscope measurement model:

$$\dot{\theta}_{gyro_k} = \dot{\theta}_k + \beta_k + w_{\dot{\theta}_k}$$

$$w_{\dot{\theta}_k} \sim \mathcal{N}(0, \sigma^2_{gyro})$$

where β_k is the gyroscope's bias (a constant), $w_{\dot{\theta}_k}$ is the white Gaussian noise associated with the sensor, $\dot{\theta}_k$ is the true state, and $\dot{\theta}_{gyro_k}$ is the gyroscope measurement.

Then using this model we can model the angular position by integrating over the duration of one time step:

$$\theta_{gyro_k} = \theta_k + \beta_k T_s + w_{\theta_k}$$
$$w_{\theta_k} \sim \mathcal{N}(0, \sigma^2_{gyro} T_s)$$

This gyroscope measurement model can then be used for formulate a state space model defined as:

$$\vec{x}_{k+1} = \mathbf{A}\vec{x}_k + \mathbf{B}u_k + \vec{w}_k$$

where:

$$\mathbf{A} = \begin{bmatrix} 1, 0, -T_s \\ 0, 0, -1 \\ 0, 0, 1 \end{bmatrix} \text{ and } \vec{x}_k = \begin{bmatrix} \theta_k \\ \dot{\theta}_k \\ \beta_k \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} Ts \\ 1 \\ 0 \end{bmatrix} \text{ and } u_k = \dot{\theta}_{gyro_k}$$

$$\vec{w}_k = \begin{bmatrix} w_{\theta_k} \\ w_{\dot{\theta}_k} \\ 0 \end{bmatrix}$$

The state space model incorporates the gyroscope measurement model in matrix **A** which describes the unforced state transition, and **B** represents the forced state transition and is used to incorporate gyroscope measurements into the process. Since gyroscope measurements drive this process, the process noise $\vec{w}_k$ is the gyroscope sensor noise.

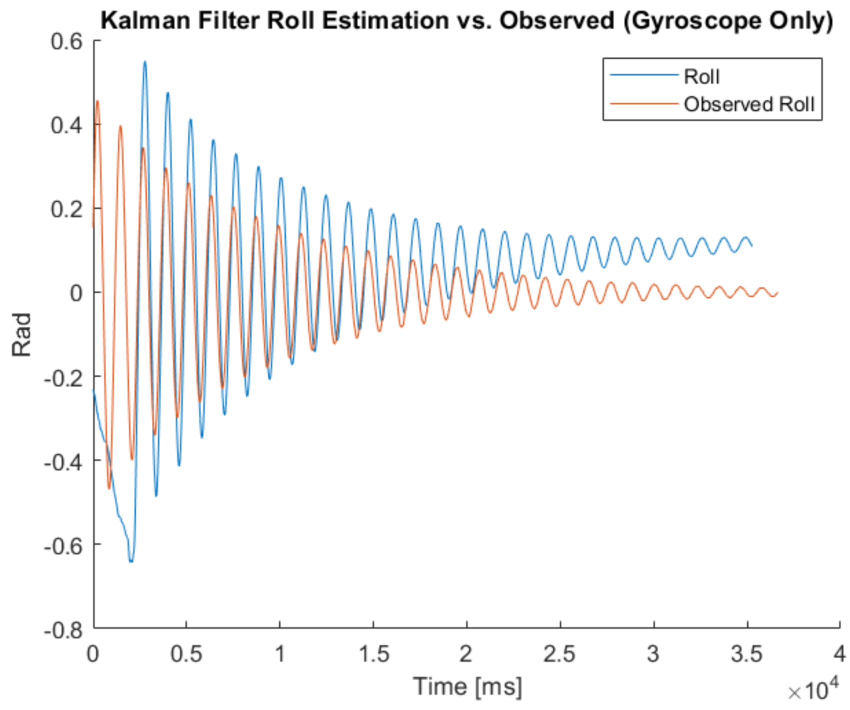

Figure 34: Position Estimation Using Gyroscope Model

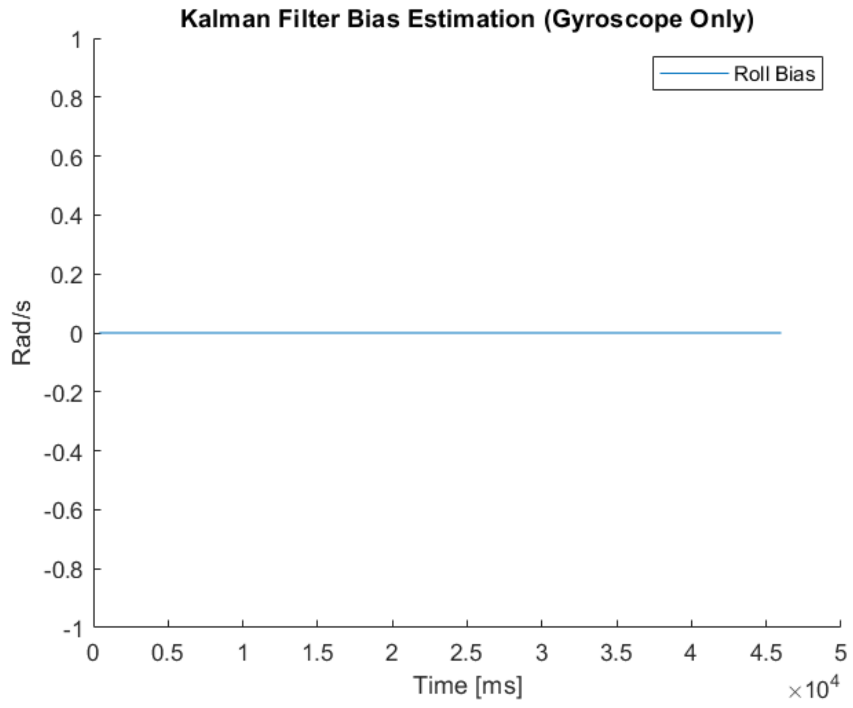

Figure 35: Bias Estimation Using Gyroscope Model

Figure 34 and 35 show the state estimation using this process model. The observed position values are derived from the visual analysis of the pendulum's motion (section 5.3), and the estimated position values have been synchronized with them along the time axis. As you can see, integrating the gyroscope measurements result in drift as there is not enough information available to estimate the gyroscope's bias.

Next, the measurement model is defined:

$$z_k = \mathbf{C}\vec{x}_k + v_k$$

where:

$$\mathbf{C} = [1, 0]$$

$$z_k = \theta_{accel_k} = \arctan \frac{a_{y_k}}{\sqrt{a_{x_k}^2 + a_{z_k}^2}}$$

$$v_k \sim \mathcal{N}(0, \sigma_{\theta_{accel}}^2),\ \sigma_{\theta_{accel}}^2 = 1.1068 \times 10^{-5}\ \text{rad}^2$$

The measurement model incorporates the accelerometer measurements, enabling the Kalman filtering algorithm to estimate the gyroscope bias and eliminate gyroscope drift. The value of $\sigma_{\theta_{accel}}^2$ was determined by computing θ_{accel_k} values and then their variance using acceleration data from the static data collection event described in section 4.2.

Next, we define the Kalman filter the measurement and process noise covariance matrix.

$$\mathbf{Q} = E[\vec{w}_k\vec{w}_k^T] = \begin{bmatrix} E[w_{1_k}^2] & E[w_{1_k}w_{2_k}] & E[w_{1_k} \times 0] \\ E[w_{1_k}w_{2_k}] & E[w_{2_k}^2] & E[w_{2_k} \times 0] \\ E[w_{1_k} \times 0] & E[w_2 \times 0] & E[0 \times 0] \end{bmatrix} = \begin{bmatrix} \sigma_{gyro}^2 T_s & 0 & 0 \\ 0 & \sigma_{gyro}^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R} = E[v_k v_k^T] = E[v_k^2] = \sigma_{\theta_{accel}}^2$$

where:

$$Var[w_{1_k}] = E[w_{1_k}^2] - E[w_{1_k}]^2 = E[w_{1_k}^2] = \sigma_{gyro}^2 T_s$$

$$E[w_{1_k}w_{2_k}] = E[w_{1_k}]E[w_2] = 0$$

$$Var[w_{2_k}] = E[w_{2_k}^2] - E[w_{2_k}]^2 = E[w_{2_k}^2] = \sigma_{gyro}^2$$

and:

$$Var[v_k] = E[v_k^2] - E[v_k]^2 = E[v_k^2] = \sigma_{\theta_{accel}}^2$$

Finally, to fully define the Kalman filter we define the priori state estimate and priori error covariance matrix.

$$\hat{\vec{x}}_0^- = 0_{3x1}$$

$$\mathbf{P}_0^- = \mathbf{I}_3 \text{ (manually tuned)}$$

The priori state estimate is expanded to a 3x1 zero vector, which assumes no gyroscope bias. The priori error covariance matrix was manually tuned to the identity matrix just as with the previous Kalman filter in section 6.1.1.

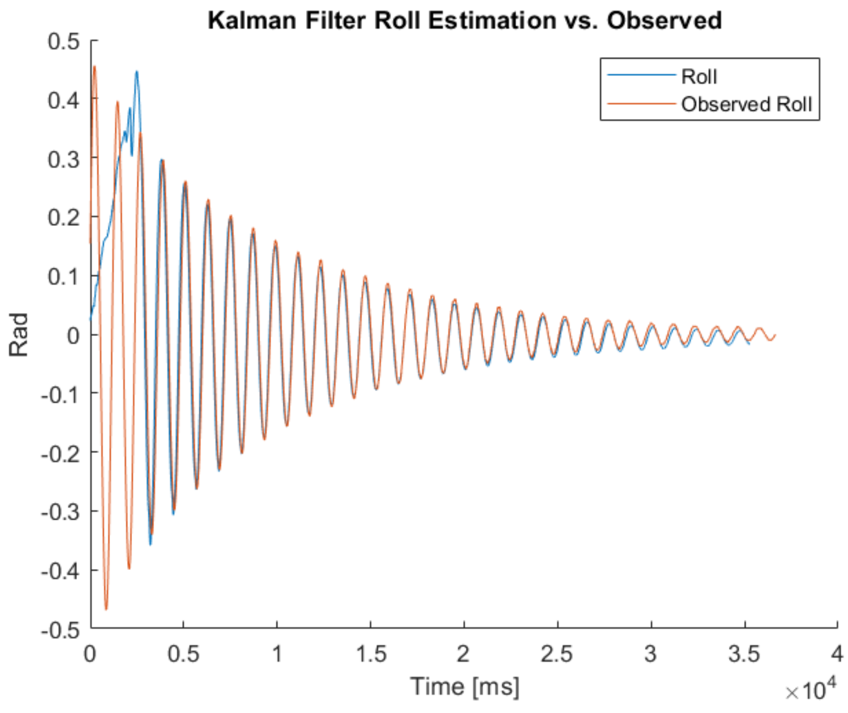

Figure 36: Position Estimation Including Measurement Model

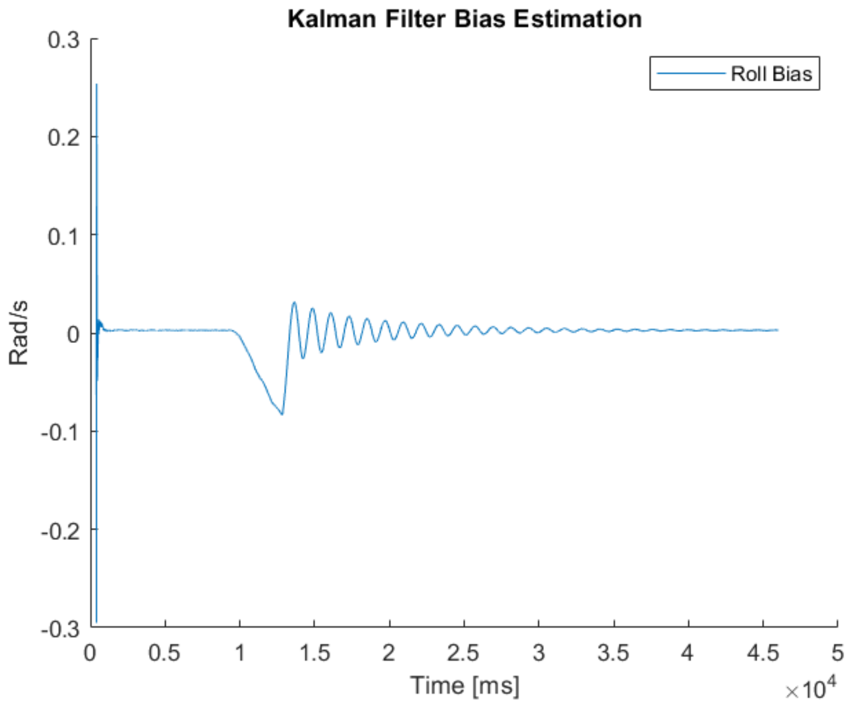

Figure 37: Bias Estimation Including Measurement Model

Figure 36 and 37 shows the state estimation using data from both the gyroscope and accelerometer. As you can see, the addition of the gyroscope data enables the filter to estimate the gyroscope's bias and produce accurate position estimates.

6.2 Complementary Filter

In our pendulum system, we use a complementary filter to estimate the angular displacement θ by fusing measurements from the MPU6050's accelerometer and gyroscope sensors.

To determine the optimal gain α for our system, we develop a theoretical framework based on the complementary filter equation shown in Figure 4. For simplicity, we omit the Laplace transform variable s and define the following variables:

- θ_k: The true angular displacement
- $\hat{\theta}_k$: The estimated angular displacement
- $w_{\theta_k} \sim \mathcal{N}(0, \sigma^2_{gyro} T_s)$, noise from the gyroscope measurement for θ_{gryo_k}
- $v_k \sim \mathcal{N}(0, \sigma^2_{\theta_{accel}})$, noise from the accelerometer measurement for θ_{accel_k}
- $e = \hat{\theta}_k - \theta_k$: The estimation error

The angular displacement estimate can be expressed as:

$$\hat{\theta}_k = \theta_k + v_k(1-\alpha) + w_{\theta_k}\alpha$$

The estimation error is therefore:

$$e = v_k(1-\alpha) + w_{\theta_k}\alpha$$

To minimize this error, we calculate the mean squared error (MSE):

$$E[e^2] = E\left[(v_k(1-\alpha) + w_{\theta_k}\alpha)^2\right]$$

Since the noise terms are independent ($E[v_k w_{\theta_k}] = 0$), this expands to:

$$\text{MSE} = \sigma^2_{\theta_{accel}}(1-\alpha)^2 + \sigma^2_{gyro} T_s \alpha^2$$

Differentiating with respect to α and setting it to zero:

$$\frac{\partial \text{MSE}}{\partial \alpha} = -2\sigma^2_{\theta_{accel}}(1-\alpha) + 2\sigma^2_{gyro} T_s \alpha = 0$$

yields the theoretical optimal α:

$$\alpha = \frac{\sigma^2_{\theta_{accel}}}{\sigma^2_{\theta_{accel}} + \sigma^2_{gyro} T_s}$$

For the variances derived earlier, this calculation gives $\alpha = 0.998$, suggesting a stronger reliance on the gyroscope.

To validate this, we conducted an empirical test, where we evaluated α values from 0.1 to 1.0 in increments of 0.1 and documented the resultant uncertainty in Table 10. The test indicates that the computed $\alpha = 0.998$ does achieve a minimum uncertainty. This value of α emphasizes a greater reliance on the gyroscope than the accelerometer.

Using the validated optimal value of $\alpha = 0.998$, we compared our complementary filter estimates against ground truth measurements from the vision system. As shown in Figure 38, the estimated angular displacements demonstrate strong alignment with the actual measurements, with only minor drift.

Table 10: Comparison of Final Uncertainty for Different α Values

Alpha	**Variance (rad^2/s^2)**
0.10	3.0092e-2
0.20	2.7163e-2
0.30	2.4412e-2
0.40	2.1779e-2
0.50	1.9207e-3
0.60	1.6634e-3
0.70	1.39763e-3
0.80	1.1091e-3
0.90	7.6368e-4
0.998*	**2.333e-4**
1.00	3.1957e-3

Note: * indicates the theoretically calculated optimal α.

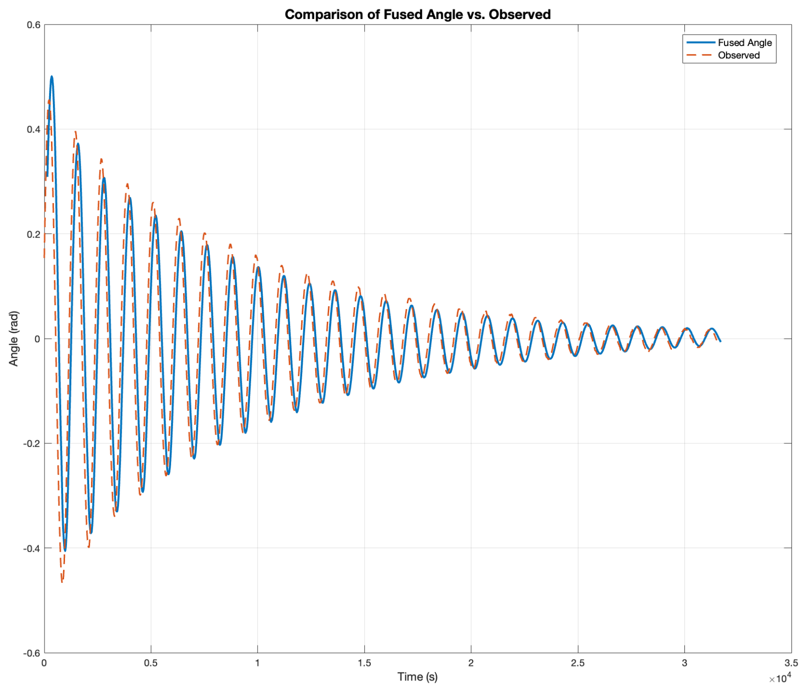

Figure 38: Comparison of Angular Position (θ) between Complementary Filter Estimation ($\alpha = 0.998$) and Vision System Measurements.

6.3 Performance Comparison

For comparing the estimators, we used our ground truth data from the vision system to calculate the mean square error (MSE) to evaluate the accuracy of each estimation method. The MSE quantifies the average squared deviation between the estimated state ($\hat{x}$) and the true state (x) from our vision system:

$$\text{MSE} = E[(\hat{x} - x)^2] = \frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i - x_i)^2 \tag{2}$$

The results for both angular position and angular velocity estimation are shown in Tables 11 and 12.

Table 11: Comparison of Angular Position (θ) Estimation Methods

Method	**MSE (rad^2)**
Kalman Filter 1	1.2e-3
Kalman Filter 2	1.32e-2
Kalman Filter 2 (converged)	**3.7573e-4**
Complementary Filter ($\alpha = 0.998$)	9.73e-4

Table 12: Comparison of Angular Position (θ) Estimation Error Variance

Method	**Variance (rad^2/s^2)**
Kalman Filter 1	5.1364
Kalman Filter 2	**4.67e-7**
Complementary Filter ($\alpha = 0.998$)	2.333e-4

Among the three filtering methods tested, Kalman filter 2 achieved the lowest converged Mean Square Error (MSE) of 3.753e-4 rad^2, computed using data from the 2.5-second mark onwards, after the filter had effectively adapted to the pendulum's motion. Initially, Kalman filter 2 exhibited a higher MSE of 1.32e-3 rad^2 when evaluated against the ground truth using all available data. From the plot, it is evident that the filter took a few oscillations of the pendulum to adapt and begin accurately predicting the pendulum's motion. In comparison, Kalman filter 1 initially showed an MSE of 1.2e-3 rad^2, which was slightly better than Kalman filter 2's initial MSE, but the performance of Kalman filter 1 did not improve significantly over time. The complementary filter, with an optimal $\alpha = 0.998$, achieved an MSE of 9.73e-4 rad^2, performing marginally better than the initial results of both Kalman filters. However, Kalman filter 2 ultimately proved superior, with its final converged MSE of 3.753e-4 rad^2 representing the best performance among all tested methods.

7 Conclusion

This research focused on estimating the state of a pendulum system using sensor fusion techniques, leveraging a 6DOF Inertial Measurement Unit (IMU) based on the MPU6050 sensor, interfaced with an Arduino UNO. The project utilized a linear state-space model to estimate the pendulum's angular position and velocity, validated against ground truth data obtained from a camera-based tracking system with visual markers. Through this process, sensor noise was characterized, which laid the groundwork for comparing two sensor fusion techniques: the Kalman filter and the complementary filter.

Two approaches were employed to develop the Kalman filter. The first Kalman filter used a state-space model fit to ground truth data, which incorporated gyroscope measurements to estimate the pendulum's angular displacement. The second approach involved a more sophisticated state-space model, where the gyroscope data was modeled with both an unforced and a forced term, alongside an additional hidden state to estimate sensor bias. This model allowed for the integration of both accelerometer and gyroscope data, leading to more accurate estimates of the pendulum's angular displacement. In contrast, the complementary filter serves as a simpler alternative to the Kalman filter. The optimal blending factor α was computed from sensor noise variances, enabling the fusion of accelerometer and gyroscope data to estimate the angular position.

The results demonstrated that the second Kalman filter outperformed the complementary filter in accurately estimating the pendulum's states, especially after it had effectively adapted to the pendulum's motion. By effectively fusing the accelerometer and gyroscope data, the second Kalman filter was able to provide precise estimates of both angular position, making it the preferred method for sensor fusion in this project.

Despite the success of the Kalman filter, challenges remained, particularly in addressing the inherent non-linearity of the pendulum system. For example, we needed to make small angle approximations for this system to remain linear. Future work will explore advanced methods, such as the extended Kalman filter (EKF), to better handle these non-linearities and further improve the accuracy of state estimation. The project highlights the importance of robust sensor fusion techniques and careful consideration of model constraints when dealing with real-world systems, and serves as a foundation for more sophisticated approaches in the future.

8 Reflection

8.1 Braedan Kennedy

I decided to enroll in this course for several reasons; the first being it is highly valued by my superiors at the research corporation I work at. The second being stochastic processes play a large role in my thesis involving deep reinforcement learning. And the third and most important being the topic of Kalman filters and sensor fusion has intrigued me since high school. Professor McKell once asked the class if we had in our possession and IMU sensor prior to starting this course, and I should have answered – yes, this IMU has been sitting in my closet for five years waiting for this moment. Several times on my own I had tried to work through articles explaining how Kalman filters work, but eventually I would get discouraged by the foreign concepts that underpin its workings. When I enrolled in this course I knew it was going to be challenging for me, but I also knew that this time I was going to put in the work.

The course began with concepts I was familiar with having taken STAT 350, but quickly took them to a new level. In particular, I remember writing code in the first homework assignment to compute the likelihood of a circuit board percolating that I thought was an interesting application of Monte Carlo simulation. Then the course quickly transitioned into utilizing concepts from linear algebra I was quite rusty with, having taken my last math course (MATH 244) during my freshman year. For example, in the second homework assignment I had to review eigenvalues and eigenvectors to show that a real symmetric matrix is positive definite iff all its eigenvalues are strictly positive. However, as soon as the fourth homework assignment I was completely out of my depth working with foreign concepts such as Fourier and Laplace transforms, convolution, and basically anything in the continuous time or frequency domain. My coursework as a CPE student involved taking EE 327, the discrete time signals and systems class, in which several of these topics were taught in the discrete case but not well and not at the depth required to enable me to compute the PSD of the output of an LTI system given a certain frequency response and driven by a random process with a certain PSD.

From there on, every homework assignment turned into an opportunity for me to teach myself the concepts I was lacking in. It was difficult, and I had to drop my other graduate level course to make time for it, but through diligent study and showing up to office hours almost every week I was able to keep up with the pace of the class and ultimately design a Kalman filter that accurately estimates orientation by fusing data from both the gyroscope and an accelerometer sensor present in our IMU.

Major topics that I learned in this course are utilized in sensor noise characterization, which involves understanding the autocorrelation, power spectral density, stationarity, and ergodicity of the noise. Additionally, I found the topics involved in state space modelling particularly interesting. Knowing how to model a system, analyze its stability and observability, and discretize a continuous time model are powerful concepts that I can see myself utilizing in future projects.

8.2 Luis Garcia

Starting EE 525, I was actually quite nervous. In terms of academic preparation, I knew this course would involve linear algebra and differential equations again, but what truly intimidated me was the STAT 350 material on probability and random variables. I didn't perform particularly well in STAT 350, but that's precisely why I found it so challenging and rewarding—it became something I wanted to improve on. Taking Stochastic Processes allowed me to do just that.

From the start, I found myself grateful for the probability refresher and the concept of supports, particularly how they enable assumptions to hold. I was reminded of expectations and now firmly associate them with averages, as they fundamentally represent just that. The progression toward understanding the Kalman filter was well worth the effort. You brought back my continuous-time signal (EE 228) knowledge, which I worked hard to grasp, especially when defining system models and then the new knowledge came with discretizing them using the Van Loan method.

None of this would have been possible without delving into noise analysis, both in our project and classwork, breaking it down to establish stationarity and ergodicity. This involved analyzing the mean and variance over time then using autocorrelation and power spectral density to confirm convergence and time invariance. Some of the most memorable takeaways for me were the words Gaussian, Gaussian, Gaussian. Your constant reminder that a Gaussian process input to a system yields a Gaussian output is now ingrained in me.

Regarding the project, it has already become a conversation starter in interviews. This is because it taught me more about collaboration and how complex a pendulum swing experiment can be. In simulations, it's always portrayed so simply. However, even the initial setup took six attempts, and collecting usable data was no easy process. Finally, we could start analyzing the noise, to hopefully find that it was Gaussian white noise, which it was. The homework assignments were instrumental in building my technical abilities. However, when it came to setting up the theoretical state model, I often found myself brainstorming with Braedan and scratching my head, confusing states and measurements. Eventually, I realized that measurements are outputs, while states are what we aim to estimate. The Gramian showed me how observability can change over time, which was a critical breakthrough for me.

This class has been a fantastic way to conclude my master's degree at Cal Poly and has brought lots of nostalgia. Though control systems were never my primary focus, in my career I always find myself near them. Even now at my current job where I'm technically part of the control systems team but focus on testing, this course has given me valuable insights into the systems we create.

References

[1] B. Siepert and I. Wellish, "Mpu6050 6-dof accelerometer and gyro," Nov 2019. [Online]. Available: https://learn.adafruit.com/mpu6050-6-dof-accelerometer-and-gyro/overview

[2] B. Siepert, "Adafruit mpu6050," Jun 2023. [Online]. Available: https://github.com/adafruit/Adafruit_MPU6050

[3] "Mpu-6000 and mpu-6050 product specification revision 3.4 mpu-6000/mpu-6050 product specification," Aug 2013. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[4] "Mpu-6000 and mpu-6050 register map and descriptions revision 4.2," 2015, accessed: 2024-10-11. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

[5] B. P. Lathi and R. Green, *Signal Processing and Linear Systems*, 2nd ed. New York, NY: Oxford University Press, 2021.

[6] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 4th ed. Hoboken, NJ: John Wiley & Sons, 2012.

[7] "Routh-Hurwitz Criterion - an overview | ScienceDirect Topics." [Online]. Available: https://www.sciencedirect.com/topics/engineering/routh-hurwitz-criterion

[8] Stan, "Arduino uno and the invensense mpu6050 6dof imu," https://42bots.com/tutorials/arduino-uno-and-the-invensense-mpu-6050-6dof-imu/, Apr 2014, [Online; accessed 22-Nov-2024].

[9] V. Pesce, P. Hermosin, A. Rivolta, S. Bhaskaran, S. Silvestrini, and A. Colagrossi, "Chapter nine - navigation," in *Modern Spacecraft Guidance, Navigation, and Control*, V. Pesce, A. Colagrossi, and S. Silvestrini, Eds. Elsevier, 2023, pp. 441–542. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323909167000093

Appendix A Author Contributions

A.1 Braedan Kennedy Contributions

- Led the integration of the Arduino microcontroller with the MPU6050 sensor and managed the Git repository.
- Led the analysis of noise power spectral density and autocorrelation.
- Led the visual analysis of pendulum motion.
- Led the development of the Kalman filter.

A.2 Luis D. Garcia Contributions

- Contributed to the Git repository and developed scripts for data parsing and logging to CSV format.
- Led the analysis of noise stationarity and ergodicity.
- Led the development of pendulum motion model.
- Led the development of the complementary filter.

Appendix B Code

The complete code used for data acquisition and analysis is in our GitHub repository: `https://github.com/kennedyengineering/EE525_Final_Project`

Appendix C Hardware and Software Requirements

Table 13: Hardware and Software Requirements

Requirement Type	Description
Hardware	x64-Based PC or Laptop
	Arduino UNO
	6-Foot USB Cable for Arduino UNO
	Breadboard
	MPU6050 Sensor
	Jumper Wires (5 Wires)
	Tape
	2X4 Wooden Beam
	Battery Powered LEDs (4 LEDs, White)
	Green Expo Marker
	Video Camera
Software	Linux/macOS Operating System
	Data Logging Script (Appendix B)
	Visual Analysis Tool (Appendix B)
	Arduino IDE
	Arduino Firmware (Appendix B)

Appendix D Raw Swing Data Plots Overlayed

D.1 Overlaid Accelerometer Data

Figure 39 presents the overlaid x, y, and z acceleration measurements from the pendulum experiment. The acceleration is measured in $\frac{m}{s^2}$ and plotted against time in milliseconds.

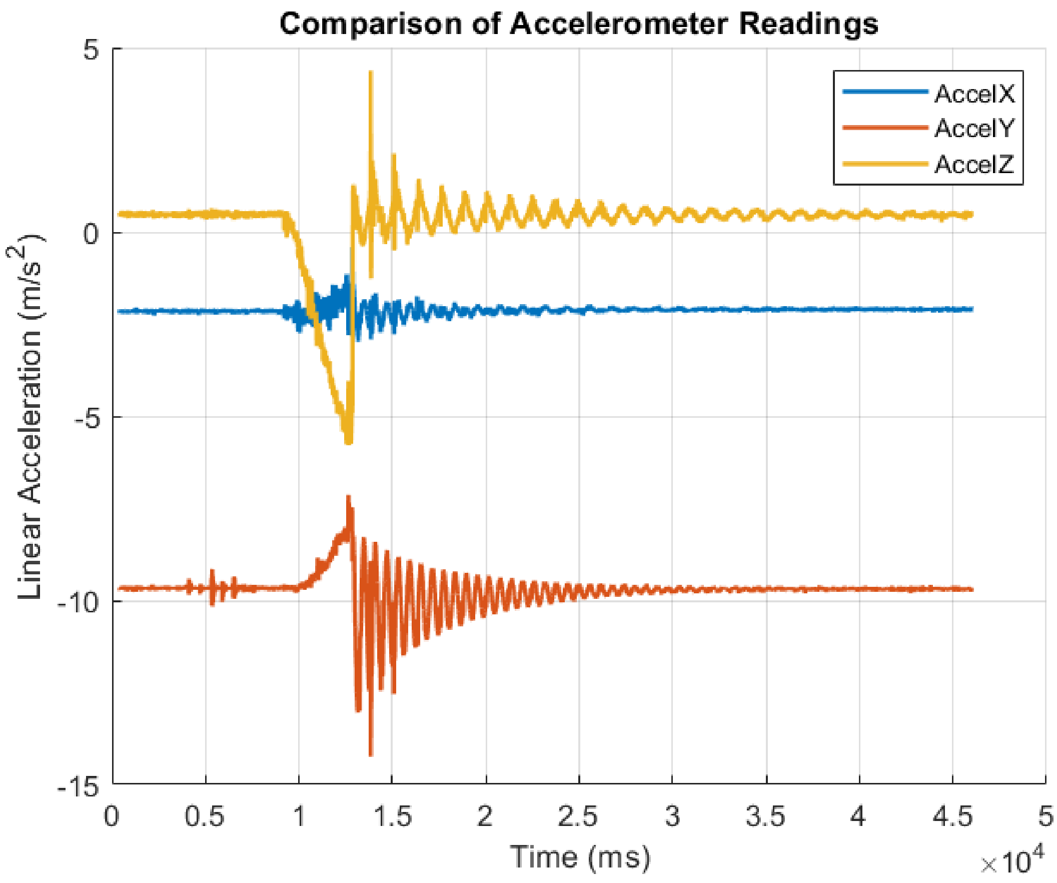

Figure 39: Overlaid Tri-Axial Accelerometer Measurements in $\frac{m}{s^2}$ During Pendulum Swing

D.2 Overlaid Gyroscope Data

Figure 40 shows the overlaid x, y, and z angular velocity measurements from the gyroscope. The angular velocity is measured in $\frac{rad}{s}$ and plotted against time in milliseconds.

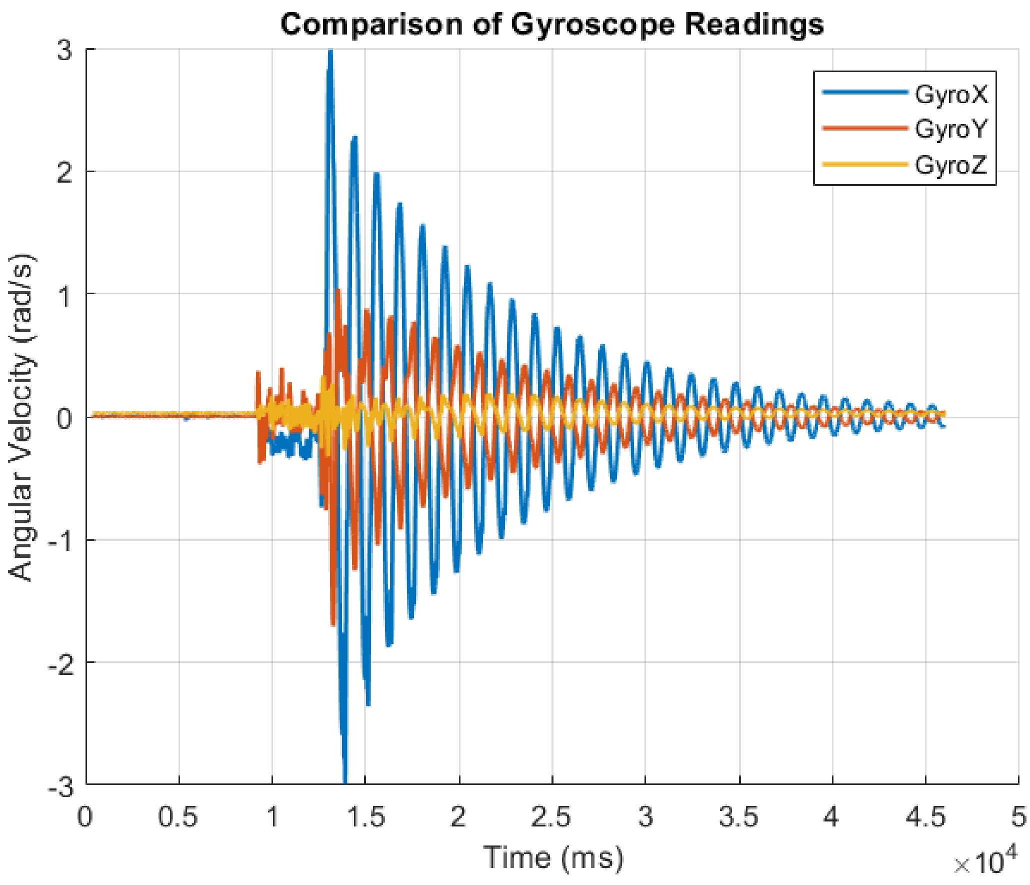

Figure 40: Overlaid Tri-Axial Gyroscope Measurements in $\frac{rad}{s}$ During Pendulum Swing

Appendix E Raw Static Data Plots Overlayed

E.1 Overlaid Accelerometer Data

Figure 41 presents the overlaid x, y, and z acceleration measurements from the accelerometer, when the pendulum is at rest to help characterize the noise. The acceleration is measured in $\frac{m}{s^2}$ and plotted against time in milliseconds.

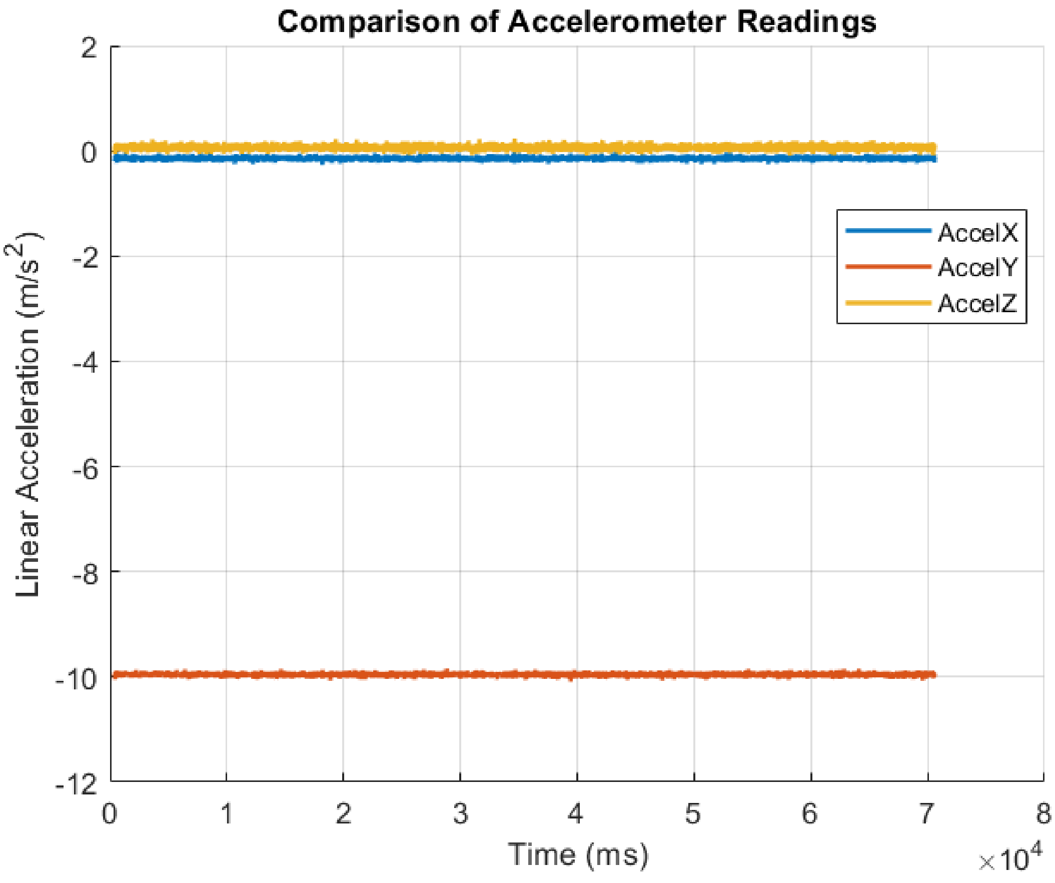

Figure 41: Overlaid Tri-Axial Accelerometer Measurements in $\frac{m}{s^2}$ at Pendulum Rest

E.2 Overlaid Gyroscope Data

Figure 42 shows the overlaid x, y, and z angular velocity measurements from the gyroscope, when the pendulum is at rest to help characterize the noise. The angular velocity is measured in $\frac{rad}{s}$ and plotted against time in milliseconds.

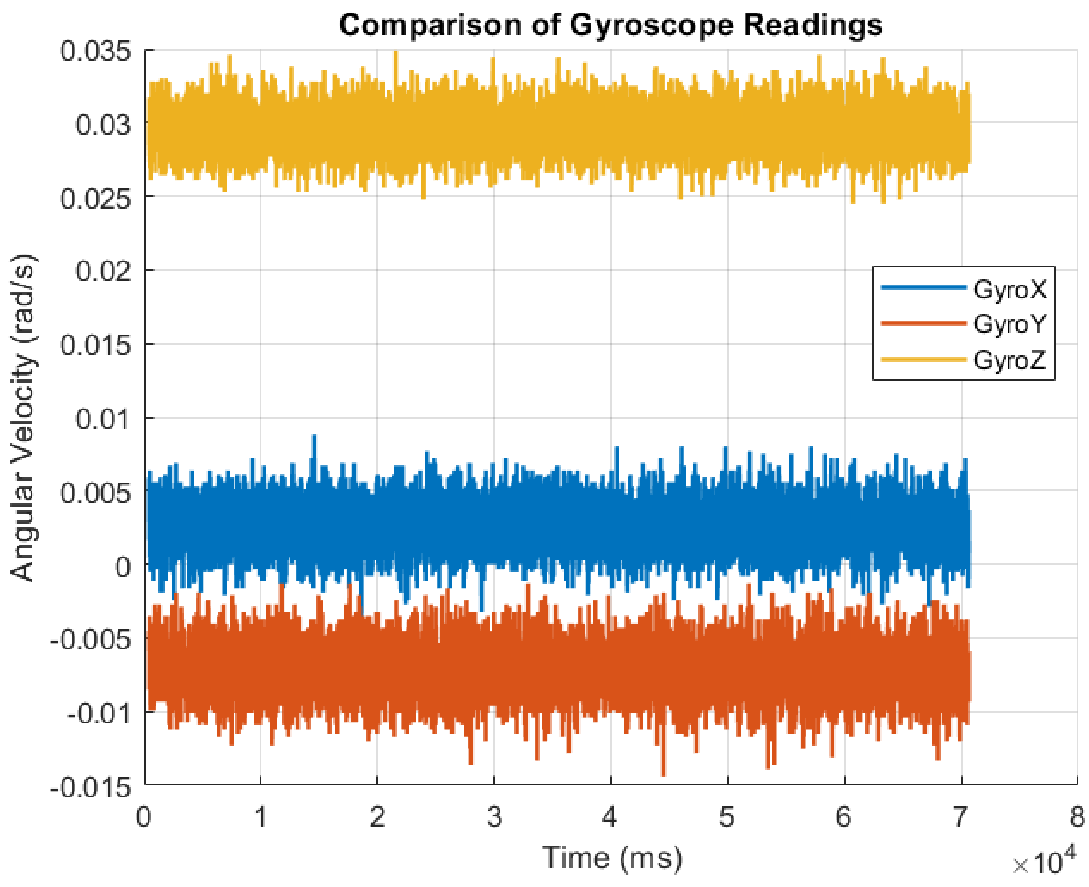

Figure 42: Overlaid Tri-Axial Gyroscope Measurements in $\frac{rad}{s}$ at Pendulum Rest