Bluetooth Low Energy and Inertial Navigation Sensor Fusion for Indoor Localisation on Embedded Devices

Nikhil J. Babani

3rd Year Project Final Report

Department of Electronic &
Electrical Engineering

UCL

Supervisor: Professor John E. Mitchell

Friday 19th April, 2024

I have read and understood UCL's and the Department's statements and guidelines concerning plagiarism.

I declare that all material described in this report is my own work except where explicitly and individually indicated in the text. This includes ideas described in the text, figures and computer programs.

I acknowledge the use of Chat-GPT-3.5 as a debugging resource for Python, Arduino, and MATLAB code. The publisher is OpenAI and it is accessible at the following URL: [https://chat.openai.com].

This report contains 71 pages (excluding this page and the appendices) and 11603 words.

Signed: [illegible] Date: Friday 19th April, 2024

Abstract

Stochastic variation in radio-frequency waves in indoor environments gives rise to challenges in indoor localisation. However, with the advancements in embedded devices and utilising sensor fusion, we can achieve positional inference for a given target. Our system leverages Bluetooth Low Energy Received Signal Strength Indicator trilateration with accelerometer & gyroscope fusion through multithreading. Our proposed solution uses a Particle Filter with 5000 particles and we achieved average errors of $\mu_\varepsilon = 0.9952$ **and standard deviation of** $\sigma_\varepsilon = 0.5575$**. Our system was robust as we maintained error consistency with a 75th percentile value of 1.3311 m.**

Bluetooth Low Energy and Inertial Navigation Sensor Fusion for Indoor Localisation on Embedded Devices

Nikhil J. Babani

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AHRS	Attitude and Heading Reference System
AoA	Angle of Arrival
AWGN	Additive White Gaussian Noise
BLE	Bluetooth Low Energy
BLET	Bluetooth Low Energy Trilateration
BPM	Beats Per Minute
CDF	Cumulative Distribution Function
CI	Confidence Interval
DCM	Direction Cosine Matrix
DMP	Digital Motion Processor
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
ENU	East-North-Up
FC	Foster Court
FIR	Finite Impulse Response
GNSS	Global Navigation Satellite System
IDE	Integrated Development Environment
IIR	Infinite Impulse Response
IMU	Inertial Measurement Unit
IPS	Indoor Positioning System
IQR	Interquartile Range

KF	Kalman Filter
LoS	Line of Sight
LSQ	Linear Least Squares
MARG	Magnetic, Angular Rate, and Gravity
MCU	Micro Controller Unit
MPEB	Malet Place Engineering Building
NLoS	Non-Line of Sight
PDF	Probability Density Function
PDR	Pedestrian Dead Reckoning
PF	Particle Filter
RSSI	Received Signal Strength Indicator
RTOS	Real-Time Operating System
SKPF	Sigma-point Kalman Particle Filter
Std Dev	Standard Deviation
TP	Torrington Place
UKF	Unscented Kalman Filter
UUID	Universally Unique Identifier
UWB	Ultra-Wideband

Chapter 1. Introduction

1.1 Background & Motivation

For most physical systems, sensors are deployed to obtain precise measurements of various quantities of interest. The numerical data obtained is vital for a system to dynamically respond to environmental changes through informed decision-making. However, real-world sensors are imperfect as they exhibit noise in their readings; obtaining an error-free measurement of any quantity is an impossible task in practice. A common approach to mitigate uncertainties is through sensor fusion, wherein different sensor information is combined algorithmically to give a more robust and accurate prediction of the state of a given system [1].

Sensor fusion algorithms like the Kalman Filter (KF), introduced by Rudolf E. Kalman in 1960 [2], have become more pertinent due to substantial increases in computational power for most digital systems. Modern technological advancements have facilitated the use of extensive signal processing particularly concerning resource-constrained embedded systems. Embedded systems can now execute complex tasks with reduced computational limitations. Our project is therefore interested in identifying a non-trivial problem where sensor fusion on an embedded system would be a possible solution.

Due to the project supervisor, Professor John E. Mitchell, having extensive expertise in the field of wireless communications, the selected field of research was indoor localisation as indoor positioning solutions typically involve the deployment of wireless technologies. This choice was motivated by the inherent challenges that come with indoor environments. Traditional methods that use a Global Navigation Satellite System (GNSS) cannot accurately determine indoor position. Whilst GNSS is suitable for outdoor positioning, it cannot be used reliably indoors due to heavy signal attenuation [3]. Adding to this, indoor localisation does not have a standardised method due to numerous practical challenges like multipathing effects and Non-Line of Sight (NLoS) issues [4]. Shadowing is also prevalent when objects are between the transmitter and receiver [5].

Indoor localisation is an ongoing topic of research with numerous applications such as asset tracking for warehouses [6], patient tracking [7], autonomous mobile robotics [8], and smartphone indoor navigation [9]. A project within this field would therefore have numerous translatable impact across a wide range of applications.

1.2 Project Goals

This project aims to design and implement an Indoor Positioning System (IPS) with a chosen application in indoor pedestrian navigation. We will leverage sensor fusion techniques to achieve $x-y$ positioning without the need for expensive nor complex hardware. We can characterise the system's success with the following goals:

1. **Accuracy:** To achieve an IPS that has greater accuracy when employing sensor fusion as compared to individual sensor measurements
2. **Robustness:** To ensure reliable performance when faced with issues like multipathing and signal attenuation
3. **Latency:** To provide real-time positioning information with reduced delay
4. **Scalability:** To have a system that can be easily deployed whilst remaining economically viable during large-scale deployment
5. **Power Consumption:** To ensure solution can be deployed in battery-powered applications for long durations

These project goals will allow us to assess the performance of the IPS to ensure that it is versatile and efficient. In line with the project motivation and goals, we will implement the IPS on an embedded device and conduct all positioning computations exclusively on the device.

1.3 Literature Review

When consulting relevant research literature, the focus was divided twofold: identifying an optimal measurement method that prioritised simplicity and selecting an appropriate sensor for fusion. The initial literature review centered on determining an appropriate wireless communication technology and measurement technique that was within the scope of the project. The second part focused on an accompanying sensor like Inertial Measurement Units (IMUs) or optical sensors that could be fused with the wireless technology for greater indoor positioning accuracy.

A review on indoor localisation methods emphasised the exploration of IEEE 802.11 (Wi-Fi), Ultra-Wideband (UWB), Bluetooth Low Energy (BLE), and IMU data due to their widespread presence in smartphones [10]. Other wireless technologies like ZigBee and RFID lack availability or scalability for deployment across a building floor [11]. Therefore, a large emphasis of the literature review was focused on UWB, BLE and Wi-Fi for wireless technologies shown in Table 1.

We have selected BLE due to its low cost, low energy, and high scalability. Using BLE, we can leverage the Received Signal Strength Indicator (RSSI) values which vary with distance. As BLE v5.0 has shown greater signal stability [12] and greater range [13], we have chosen a Micro Controller Unit (MCU) that supports BLE v5.0 as our embedded device.

Visible Light Communication via Light Emitting Diodes suffers from NLoS and acoustic sensors utilising microphones require additional infrastructure in the form of reference nodes making these sensors not ideal for our localisation goals [14]. A common alternative utilises IMUs where a Pedestrian Dead Reckoning (PDR) model will characterise the user's dynamics to obtain their position. We therefore identified IMUs as cheap and widely available sensors for our fusion approach. We conclude that a fusion between BLE and IMU devices would be suitable for this project.

After identifying BLE and IMUs as appropriate sensors for the project, the literature review shifted towards finding common positioning techniques for both technologies shown in Table 2. For BLE, researchers have incorporated machine learning techniques in methods like RSSI fingerprinting where a unique set of RSSI values are measured and stored in a database during an offline phase and a positioning estimate is then determined from this during the online phase. However, we intend to investigate the efficacy of our sensor fusion algorithm therefore we will not consider any machine learning models for this project. RSSI lateration is also a common technique in determining position as the complexity is low and it avoids an extensive offline phase that RSSI fingerprinting would induce. RSSI lateration involves taking RSSI measurements typically between a mobile receiver and a fixed reference transmitter, then converting into distance measurements to then find Cartesian position. We will therefore select RSSI lateration as our BLE measurement technique.

Typically, we refer to 9 Degrees of Freedom (DOF) IMUs as Magnetic, Angular Rate, and Gravity (MARG) sensors but we will simply denote both sensors as IMUs for simplicity. For IMUs, a common PDR model involves a step detection and step estimation process as this avoids acceleration integration which would induce accumulated errors [15]. The accelerometer can be used to infer the distance travelled and a fusion between the accelerometer and either a gyroscope and/or a magnetometer will allow for heading estimation. It is noted that magnetometers suffer from hard iron and soft iron distortions so the heading estimations can be perturbed by this distortion as the IMU travels around in a given space [16]. However, the magnetic north vector can be used as a reference frame for the quaternion orientation for more accurate and error tolerant Attitude and Heading Reference System (AHRS). This could result in less drift in yaw readings when obtaining the rotation quaternions from the fusion algorithm as drift is corrected through magnetometer readings [17]. In summary, we will use only 6-DOF from our IMU, which consists of just the accelerometer and gyroscope. This choice was justified as accounting for the potential disturbances in the magnetic field whilst considering RSSI inconsistencies would make the analysis of the empirical data non-trivial.

The research papers in Table 2 also highlighted the different fusion that were implemented including algorithms like the KF and the Particle Filter (PF). The Sigma-point Kalman Particle Filter (SKPF) simply combines the Unscented Kalman Filter (UKF) with the PF for reduced computational time. Both the SKPF and the PF rely on Sequential Monte Carlo methods that improve the filter's ability to estimate via increasing the number of sampled particles, with the drawback of higher computational cost [18]. The SKPF therefore lowers computation time by using the UKF to generate sigma points for the PF to sample from, effectively lowering the number of particles needed for accurate localisation.

Table 2 also highlights different measurement techniques for BLE where Angle of Arrival (AoA) has been used with a step length PDR model. Whilst the addition of the Constant Tone Extension for BLE v5.1 allows for AoA localisation, this adds further signal processing complexity and is not currently supported by mainstream hardware. However, AoA does allow for a corrective measurement to be taken for the step length and heading angle when applying the fusion algorithm. Therefore, the use of an Extended Kalman Filter (EKF) with AoA is justified for their proposed system.

From the literature, a large number of chosen fusion algorithms have been concerned with the standard Kalman Filter and Particle Filters. Therefore, a large portion of the project's efforts will be concerned with implementing and comparing these two algorithms for localisation.

Table 1: Concise overview of different wireless communication technologies and their indoor positioning techniques

Technology	Measurement method	Cost	$\mu_\varepsilon{}^1 \pm \sigma_\varepsilon{}^2$ [m]	Pros	Cons	References
Wi-Fi	RSSI[3] Lateration	Low	1.589 ± 0.714	- Low complexity	- Attenuation issues - Suffers from multipathing	[19]
	RTT[4] Lateration	Low	1.31 ± 0.90	- Widely available on consumer devices	- NLoS degradation - Multipathing issues	[20]
	CSI[5] Fingerprinting	Low	1.171 ± 0.587	- Multipath characterisation	- Needs specialised hardware/firmware	[21]
UWB	TW-ToF[6] Lateration	High	0.35 ± 0.2	- Excellent localisation	- Not widely available	[22]
	TDoA[7] Multilateration	High	0.49 ± 0.23	- Multipath characterisation	- Not widely available	[23]
BLE	RSSI Lateration	Low	1.149 ± 0.698	- Low complexity	- Attenuation issues - Suffers from multipathing	[24]
	RSSI Fingerprinting	Low	1.1 ± 0.15	- No additional infrastructure needed	- Intensive offline phase	[25]

[1]Average accuracy error in LOS situations
[2]Standard deviation of accuracy error in LOS situations
[3]Received Signal Strength Indicator
[4]Round-Trip Time
[5]Channel State Information
[6]Two-Way Time of Flight
[7]Time Difference of Arrival

Table 2: BLE with IMU sensors and their respective fusion techniques

Measurement method	PDR Method	Fusion Technique	Utilised IMU Sensors	Features	References
RSSI Lateration	Step Length	KF[8]	- Accelerometer - Magnetometer	- Avoids integration errors - Works in dense BLE setting	[26]
RSSI Fingerprinting	Step Length	PF[9]	- Accelerometer - Magnetometer - Gyroscope	- Landmark-based measurement corrections	[27]
RSSI Fingerprinting	Step Length	SKPF[10]	- Accelerometer - Gyroscope	- High accuracy - More efficient than PF	[28]
RSSI Lateration	Step Length	KF	- Accelerometer - Magnetometer - Gyroscope	- Considers complex motion - Works in dense BLE setting	[29]
AoA[11]	Step Length	EKF[12]	- Accelerometer - Gyroscope	- Heading correction - Step length correction	[30]

[8]Kalman Filter
[9]Particle Filter
[10]Sigma-point Kalman Particle Filter
[11]Angle of Arrival
[12]Extended Kalman Filter

Chapter 2. Indoor Positioning System Overview

2.1 System Objectives

Concluding the literature review, we can further refine our project goals into objectives by defining key metrics of a good IPS. We also based the metrics on statistical results provided by the literature wherein different median and 90$^{\text{th}}$ percentile values were provided. After considering different empirical results obtained from various research papers, we can define the project objectives as follows:

1. **Accuracy:** Average localisation error: $\mu_\varepsilon < 3$ m
2. **Robustness:** Standard deviation in average error: $\sigma_\varepsilon < 2$ m
3. **Latency:** Computation time: $t_{comp} < 2$ s
4. **Scalability:** Coverage Area: Area > 10x10 m^2
5. **Low Power Consumption:** Power draw: $P_{out} < 2$ W

The accuracy and robustness metric was determined by observing the Cumulative Distribution Functions (CDFs) in localisation errors across different papers [31][32], where the average and standard deviation values ($\mu_\varepsilon \pm \sigma_\varepsilon = 3 \pm 2$ m) were chosen in the presence of statistical outliers. Computational time was determined from observing different latency measurements quoted by authors when provided. The computation time describes the time taken for the IPS to compute position. The scalability metric was self defined in which the IPS should be able to localise in. The power draw specifies the output power for the IPS. We assumed 3 Energizer E91 1.5 V AA battery are used in series where each battery has 2800 mAh capacity for a discharge current of 25 mA [33]. The net capacity therefore is: $E = 1.5\text{V} \times 3 \times 2.8\text{Ah} = 12.6$ Wh. Therefore, a power draw of $P_{out} = 2$ W results in a battery life of 6.3 hours.

2.2 Hardware Choices

Given the project objectives, we now need to choose the different hardware components for our IPS. Certain MCUs were chosen due to certain demands for higher memory or faster clock speeds in light of the requirements for real time localisation. MCUs were chosen with BLE functionality with support for BLE v5.0. The IMU chosen contains a Digital Motion Processor (DMP) for quaternion orientation data resulting from sensor fusion.

1. **Arduino Nano ESP32:** The Arduino Nano ESP32 was chosen as our mobile BLE receiver. According to the datasheet provided by Arduino [34], it is based on the ESP32-S3 and features a 32-bit Dual-core Xtensa LX7 Microprocessor with clock speeds up to 240 MHz. It has 512 kB SRAM and 16 MB of external flash memory. This MCU is therefore adequate in performing real time sensor fusion and handling large code libraries for Bluetooth and IMU purposes. The u-blox NORA-W106-10B module provides the necessary BLE v5.0 functionality which was deemed necessary for the project.

2. **Raspberry Pi Pico W:** The Raspberry Pi Pico W was chosen as the fixed BLE transmitter reference nodes. From the datasheet provided by Raspberry Pi Ltd [35], the RP2040 features a 32-bit Dual-core ARM Cortex-M0+ Microprocessor with clock speeds up to 133 MHz. It has 2MB flash memory and an Infineon CYW43439 for Bluetooth v5.2 functionality. It was chosen as a cost effective BLE transmitter and only requires at least 3.3V to 5V input voltage.

3. **Adafruit BNO085:** The Adafruit BNO085 is a 9-DOF IMU with triaxial accelerometer, gyroscope, and magnetometer. From the datasheet provide by Ceva Inc [36], it contains a a 32-bit ARM Cortex-M0+ with SH-2 firmware that provides sensor fusion algorithms for quaternion data. The data provided by the DMP will be used in determining heading for our PDR solution. This means that the accelerometer and gyroscope fusion will be offloaded from the Arduino ESP32 to the IMU, reducing computational load placed on the primary MCU.

2.3 System Architecture

Figure 1 shows the higher level architecture for the IPS. It highlights the usage of the 6-DOF provided by the BNO085 through the usage of the accelerometer and gyroscope. The RSSI values are measured between each Raspberry Pico transmitter and the mobile Arduino Nano ESP32 receiver. We will therefore combine the PDR system with the Bluetooth Low Energy Trilateration (BLET) system through sensor fusion.

For the BLET system, we first measure the raw RSSI values between 3 fixed transmitter reference nodes and a singular mobile receiver. We reduce the noise found in the RSSI values using a moving average filter that considers the previous and the current RSSI value. Euclidean distances are then calculated from RSSI values using an empirical log-distance path loss model. If there are valid intersections between the radial distances computed between each transmitter and the receiver, then the linear least squares trilateration technique is performed with outlier removal to obtain an estimated position. However, this estimated position would be infrequent & distorted by indoor propagation effects.

For the PDR system, it uses a step length model for a given pedestrian and uses a heading reference from the quaternion data obtained from the fusion taken place in the DMP. Utilising a step length detector & estimator with a heading reference is a common method for PDR as it avoids accumulated acceleration error due to integration. The step detector would simply check if a valid step was taken based on certain criterion. If a valid step was taken, the user's position would be propagated based on the evaluated step length and heading. However, there would still be positional drift as the gyroscope integral values accumulate errors so the estimated position would eventually deviate from the ground truth.

To mitigate the weaknesses of both the BLET and the PDR systems, we can employ sensor fusion. The algorithm would leverage the short term accuracy of the PDR model, then correct the accumulated drift through the BLET system. The trilateration system would function as a measurement update to prevent the errors from growing unbounded. Therefore, the fading and shadowing effects are mitigated due to infrequent measurement updates. The fusion algorithm chosen, either the KF or the PF, will be tuned to either favour the prediction model or the measurement model. As both fusion algorithms employ different mathematical techniques, the method for tuning each of them will vary and will therefore be discussed in the later sections.

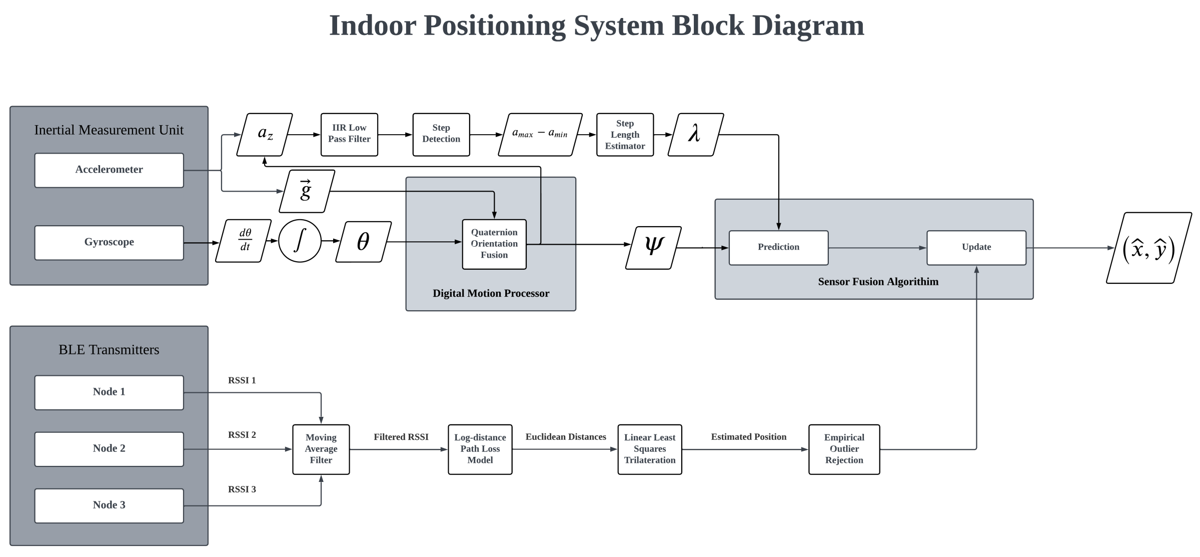


Figure 1: Combined positioning system using IMU and BLE sensor fusion

Chapter 3. Bluetooth Low Energy Trilateration

3.1 Bluetooth Low Energy

BLE is a wireless technology designed to function in power constrained systems whilst being inexpensive to implement for short range communication [37]. BLE can send packets of information during an advertisement process where an advertising device, known as a peripheral device, will broadcast a message containing a payload. This payload can carry information like the state of a sensor and this advertised package can then be received by a scanning device, known as a central device. In our project we have configured the fixed BLE transmitter to be the BLE peripheral device and the mobile BLE receiver to be the BLE central device. As mentioned in the Hardware Choices section, the Raspberry Pico will act as the advertising peripheral and the Arduino Nano ESP32 will act as the scanning central, shown in Figure 2.

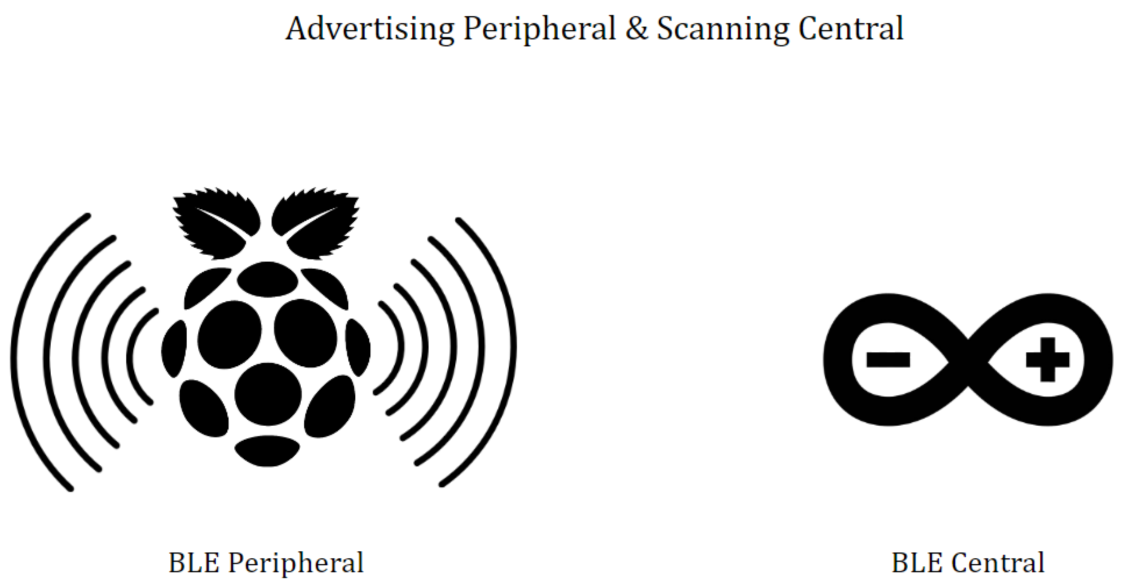


Figure 2: Fixed BLE transmitter advertising BLE signals to mobile BLE receiver

BLE, like classic Bluetooth, operates in the 2.4 GHz radio frequency band utilising 40 channels with 2 MHz spacing. This frequency operation is also shared by Wi-Fi as shown in Figure 3 which was inspired by the work done in a paper by Faragher et al. [38]. Channels 37, 38, and 39 are used for advertising to prevent minimum interference with Wi-Fi channels. For data messages, the scanning central device can initiate a connection with the peripheral and conduct channel hopping over the 37 data channels where a data payload is exchanged over these 37 channels. For our application, we are only concerned with the advertising channels as the advertising payload contains important information like the device's Universally Unique Identifier (UUID).

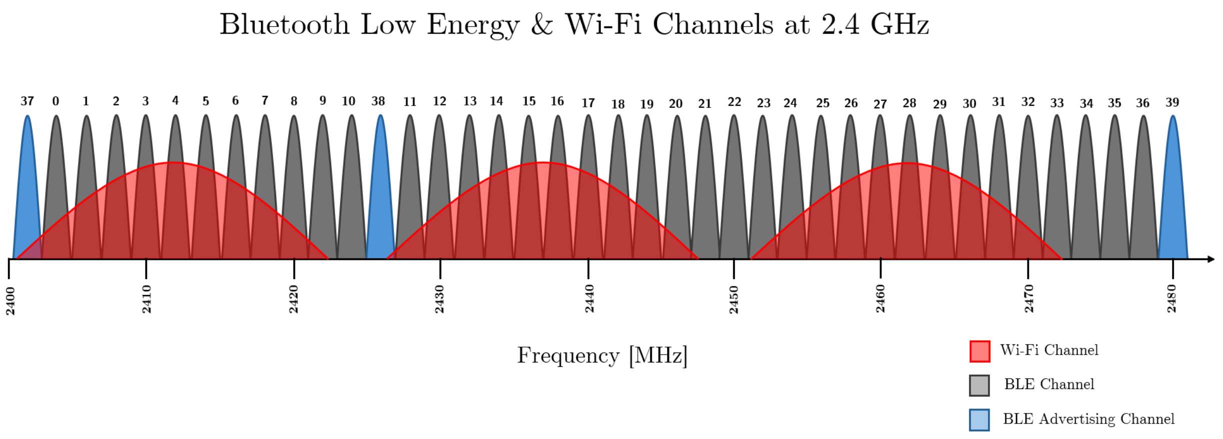


Figure 3: All 40 Bluetooth Low Energy channels

The Bluetooth Special Interest Group specified certain hexadecimal bit sequences that correspond to different service and characteristic UUIDs [39]. A service is simply a collection of characteristics where characteristics specify the type of data being received. For example, an environmental sensor could have a corresponding service UUID and the characteristics could be temperature and humidity with their own corresponding characteristic UUIDs. This is relevant to our project as we can distinguish between the 3 advertising nodes.

We can then configure our peripherals to advertise at a high frequency of 50 Hz. This value was chosen as the advertising interval must be between 20 ms and 10.24 s [40] wherein a 20 ms advertising interval would correspond to 50 Hz. A low advertising interval would result in more frequent positional updates for localisation. We can therefore configure the transmitter nodes with specific parameters for advertising, as shown in Table 3. We have also chosen the transmit (Tx) power to be equal to the default Tx power obtained from the CYW43439 datasheet [41]. The characteristic UUID of 0x2AF9 corresponds to generic level characteristic UUID.

Table 3: Transmitter node specifications

Node	Tx Power [dBm]	Service UUID	Characteristic UUID	Advertising Interval [ms]
1	8.5	0x1701	0x2AF9	20
2	8.5	0x1702	0x2AF9	20
3	8.5	0x1703	0x2AF9	20

Algorithm 1 shows the pseudocode for the advertising procedure. This was adapted from an example of using the Raspberry Pico as a BLE temperature sensor [42]. For each of the devices, we specified different service UUIDs and then uploaded the code to each transmitter using MicroPython. We also signified to the user that the peripheral was advertising by flashing the onboard LED.

Algorithm 1: BLE Advertising Process

```
Input  : Device name: $\mathcal{D}$
         Service UUID: $\mathcal{U}_s$
         Characteristic UUID: $\mathcal{U}_c$
         Advertising Interval: $I$
Output: Advertising payload: $\mathcal{P}$ at interval $I$

Function generatePayload($\mathcal{D}$, $\mathcal{U}_s$, $\mathcal{U}_c$):
    $\mathcal{P} \leftarrow \emptyset$;                          // Initialise the payload as empty
    if $\mathcal{D} \neq \emptyset$ then
        $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathcal{D}\}$;              // Include device name in the payload
    end if
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathcal{U}_s, \mathcal{U}_c\}$;                      // Add UUIDs to the payload
end
Function advertise($\mathcal{P}$, $I$):
    Start BLE advertising with payload $\mathcal{P}$ at intervals $I$;
end
Function main():
    $\mathcal{P} \leftarrow$ generatePayload ($\mathcal{D}$,$\mathcal{U}_s$,$\mathcal{U}_c$);
    advertise($\mathcal{P}$, $I$);
    while True do
        Toggle the LED;               // Indicate advertising by flashing LED
        Sleep for 1000 ms;                    // Flash LED every 1 second
    end while
end
```

3.2 Received Signal Strength Indicator

When a peripheral and central communicate, the strength of the signal received by the receiver is interpreted as the RSSI values. However, fading and shadowing affect the propagation of radio frequency signals which can impact the RSSI values indoors. Fading is a phenomenon where the strength of a wireless signal varies due to its interaction with the environment. This variation could lead to potential degradation in communication quality and reliability as the signal is impacted over time and space for a given frequency.

Fading can be divided into small and large scale fading. Small scale fading like multipath fading occurs when a signal can travel multiple paths due to reflections, leading to interference at the receiver. Large scale fading like shadowing occurs due to obstacles obstructing the line of sight between the two devices, and the attenuation of signal strength with increasing distance follows a logarithmic relationship [43].

3.2.1 Log-distance Path Loss Model

A common mathematical model for shadowing is the log-distance path loss model which describes how wireless signals attenuate during propagation over a given medium. From numerous papers [26][44][45], Equation 1 is then described as follows.

$$\Gamma(d) = \Gamma(d_0) - 10\gamma \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \tag{1}$$

Where d is the distance between the transmitter and receiver, $\Gamma(\cdot)$ is the log-distance path loss function for RSSI values, $\Gamma(d)$ represents the RSSI measurements taken at distance d, $\Gamma(d_0)$ represents the RSSI measurements taken at a fixed distance $d_0 = 1\text{m}$, γ is the path loss exponent, and X_σ represents the Additive White Gaussian Noise (AWGN) with zero mean and variance σ^2, denoted as $X_\sigma \sim \mathcal{N}(0, \sigma^2)$. We note that $\Gamma : \mathbb{R}^+ \to \mathbb{R}^-$ for $d \in \mathbb{R}^+$. When obtaining the RSSI values, we typically take the average RSSI values across the BLE advertising channels 37, 38, and 39. The inclusion of the AWGN term also reflects this averaged RSSI value.

3.2.2 Weighted Average Filter

The variations attributed to the AWGN term can be reduced through filtering. A common technique utilises a moving average filter which is a type of convolution that aims to smooth noisy signals. It aims to take the average of all the values within a given window size. Ibrahim et al. [46] states that using a filter window size that is too large would be unsuitable for real time applications. Therefore, Huang et al. [26] used a filter window of 2 terms that depends on the current RSSI and previous RSSI. We can then define the moving average filter for the i-th node, where $i \in \{1, 2, 3\}$, in Equation 2.

$$\Gamma_t^{(i)}(d) = \eta \Gamma_t^{(i)}(d) + (1 - \eta)\Gamma_{t-1}^{(i)}(d) \tag{2}$$

Where η is a constant coefficient, t denotes the continuous time step variable in which the RSSI value was observed, $t-1$ denotes the time step for the prior RSSI variable. The coefficient value selected was $\eta = 0.2$, allowing for greater noise tolerance. Choosing a lower value for η would sacrifice responsiveness for robustness as the filter would favour the prior value more. We can now empirically verify the efficacy of the filter by taking RSSI measurements.

In order to receive RSSI values, a scanning interval of 40 ms, scanning window of 20 ms, and a scan duration of 1 second was chosen. Algorithm 2 describes the process for conducting a central scan and then performing the moving average filter.

Algorithm 2: BLE Scanning & Moving Average Filter

```
Input  : Targeted UUID for $i$-th node: $\mathcal{U}_i$
Output: Filtered RSSI value for $i$-th node: $\Gamma_t^{(i)}(d)$ where $\Gamma : \mathbb{R}^+ \to \mathbb{R}^-$
        Current RSSI Measurement for $i$-th node: $\mathcal{R}^{(i)}$ where $\mathcal{R}^{(i)} :\in \mathbb{R}^-$
        BLE scan results: $\mathcal{B}_{scan}$
Initialise:
    $\eta = 0.20$;                          // Weight for moving average filter
    $\Gamma_{t-1}^{(i)}(d) = 0$;                    // Set initial previous RSSI value to 0
    $\mathcal{R}^{(i)} = 0$;   // Initialise array $\mathcal{R}^{(i)}$ to store current RSSI measurements
Function scanBLE():
    Set scan active, interval, and window;
    $\mathcal{B}_{scan} \leftarrow$ Append BLE scan results to $\mathcal{B}_{scan}$;
    return $\mathcal{B}_{scan}$;             // Results are used for moving average filter
end
Function movingAverageFilter($\mathcal{B}_{scan}$, $\eta$):
    for each device in $\mathcal{B}_{scan}$ do
        for each $i$ in $\{1, 2, 3\}$ do
            if (device has service UUID == True) $\wedge$ (service UUID == $\mathcal{U}_i$) then
                $\mathcal{R}^{(i)} \leftarrow$ Read RSSI value of device;
                if $\Gamma_{t-1}^{(i)}(d) == 0$ then
                    $\Gamma_{t-1}^{(i)}(d) \leftarrow \mathcal{R}^{(i)}$;             // First measurement taken
                end if
                else
                    $\Gamma_t^{(i)}(d) \leftarrow \eta\Gamma_t^{(i)}(d) + (1-\eta)\Gamma_{t-1}^{(i)}(d)$;     // Apply moving average
                end if
            end if
        end for
        return $\mathcal{R}^{(i)}$ and $\Gamma_t^{(i)}(d)$;
    end for
end
```

For our measurements, we conducted the experiment in Malet Place Engineering Building (MPEB) 6.02, which is the Electronic and Electrical Engineering Laboratory in UCL. The experiment was conducted under Line of Sight (LoS) conditions without any obstacle obstructions between the transmitter and receiver. Both the Raspberry Pico and Arduino ESP32 were connected to a breadboard and left flat on the table. The chip antenna direction was assumed to be in the same direction as the microprocessor chips on the respective boards. Hence, we assume the antennas are perpendicular to the surface of the table, therefore pointed vertically upwards. We measured a distance $d = 1$ m using a metre rule for testing. Surrounding electronic devices like phones and Bluetooth mice were disabled during testing to reduce signal interference.

The BLE receiver was fixed and connected to the laptop as RSSI readings were taken and processed as a comma delimited list. This was done by using `Serial.print` commands in the Arduino Integrated Development Environment (IDE). Python code for reading the serial data uses an adaptation of Michael Wrona's example for accelerometer calibration [47] but modified for RSSI measurements. The RSSI values are then appended to a data list and then processed in MATLAB. We took 50 paired measurements for the raw and filtered RSSI. The time between each measurement was 1.0091 s. The large duration between measurements was due to a suboptimal code library that lowered the overall throughput. This was rectified in later experiments.

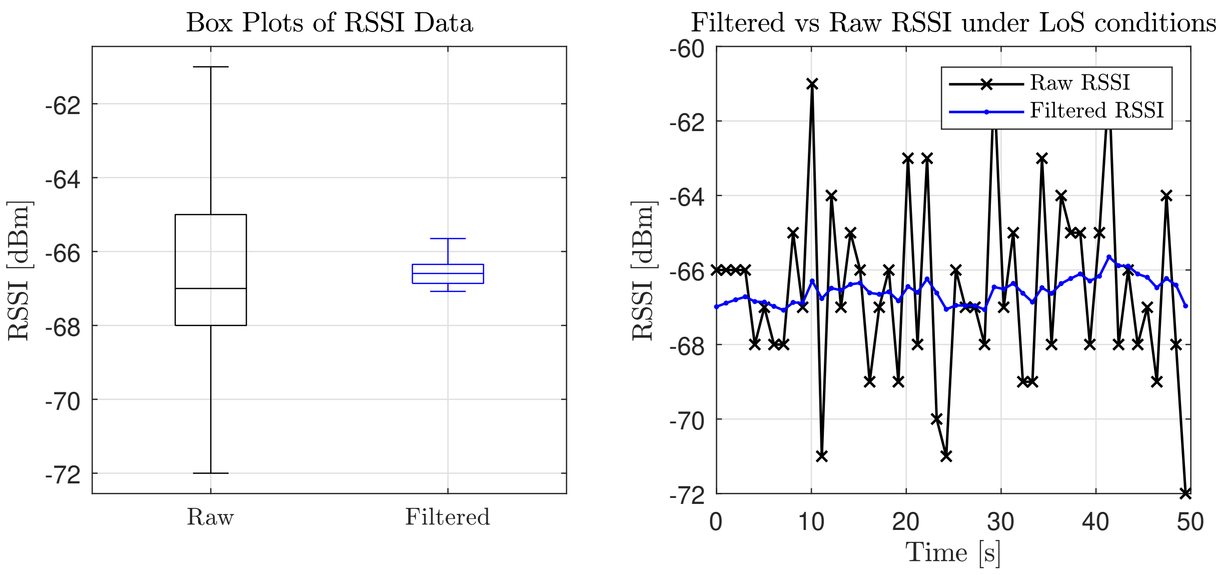


Figure 4: Filtered vs raw RSSI measurements taken in MPEB 6.02 at $d = 1$ m

The plot in Figure 4 shows a reduction in deviated values. Visually, the maximum and minimum values lie close to the expected value for that distance measurement. This suggests greater noise tolerance and a significant reduction in the AWGN present in the system. The box plot shows the different quartiles and the maximum and minimum values. We can employ statistical measures to further analyse the data. We used central tendency metrics like the sample mean and standard deviation (Std Dev) and dispersion metrics like percentiles and the Interquartile Range (IQR), as shown in Table 4.

Table 4: Statistical measures of moving average RSSI data

Measurement	Central Tendency [dBm]					Percentiles [dBm]				
	Mean	Std Dev	Min	Max	Range	25th	50th	75th	90th	IQR
Raw	-66.54	2.48	-72.00	-61.00	11.00	-68.00	-67.00	-65.00	-63.00	3.00
Filtered	-66.56	1.43	-67.08	-65.65	0.3398	-66.86	-66.59	-66.35	-66.14	0.51

Observing the central tendency, we notice a large reduction in the standard deviation and range for the filtered RSSI values. The mean values for the RSSI is also similar which is expected as the AWGN has zero mean. For the dispersion, the median values, represented by the 50th percentile, are almost identical to the mean values for both the raw and filtered RSSI values. This suggests a lack of skewness and a symmetrical distribution of RSSI values at $d = 1$ m. The reduction in the IQR for the filtered data reaffirms that the filtering technique is functioning as intended.

3.2.3 Path Loss Exponent

An important characteristic of the log-distance path loss model is the path loss exponent, γ, which is dependent on the type of environment and the antennas of the transmitter and receiver. Typical γ values under indoor LoS conditions are $\gamma \in [1.6, 1.8] \subset \mathbb{R}^+$ [48]. The γ value influences the shape of the log plot where higher γ values result in a flatter curve. Smaller γ values result in a faster decline in RSSI values for larger d values. Therefore, this necessitates γ calibration.

Using Algorithm 2, we can then plot the RSSI values at different distances for all 3 nodes. We have taken the measurements at Torrington Place (TP) UCL in room B08. To ensure proper LoS, we deployed tripods for all 3 transmitters and orientated the chip antennas to face the receiver. The tripods clamped onto each breadboard containing the battery powered Raspberry Pico. The measurements were conducted in the same positions for all 3 nodes to prevent environmental factors from distorting the measurements. We have taken 50 paired measurements for raw and filtered RSSI data for 41 distance points and plotted the results as shown in Figure 5.

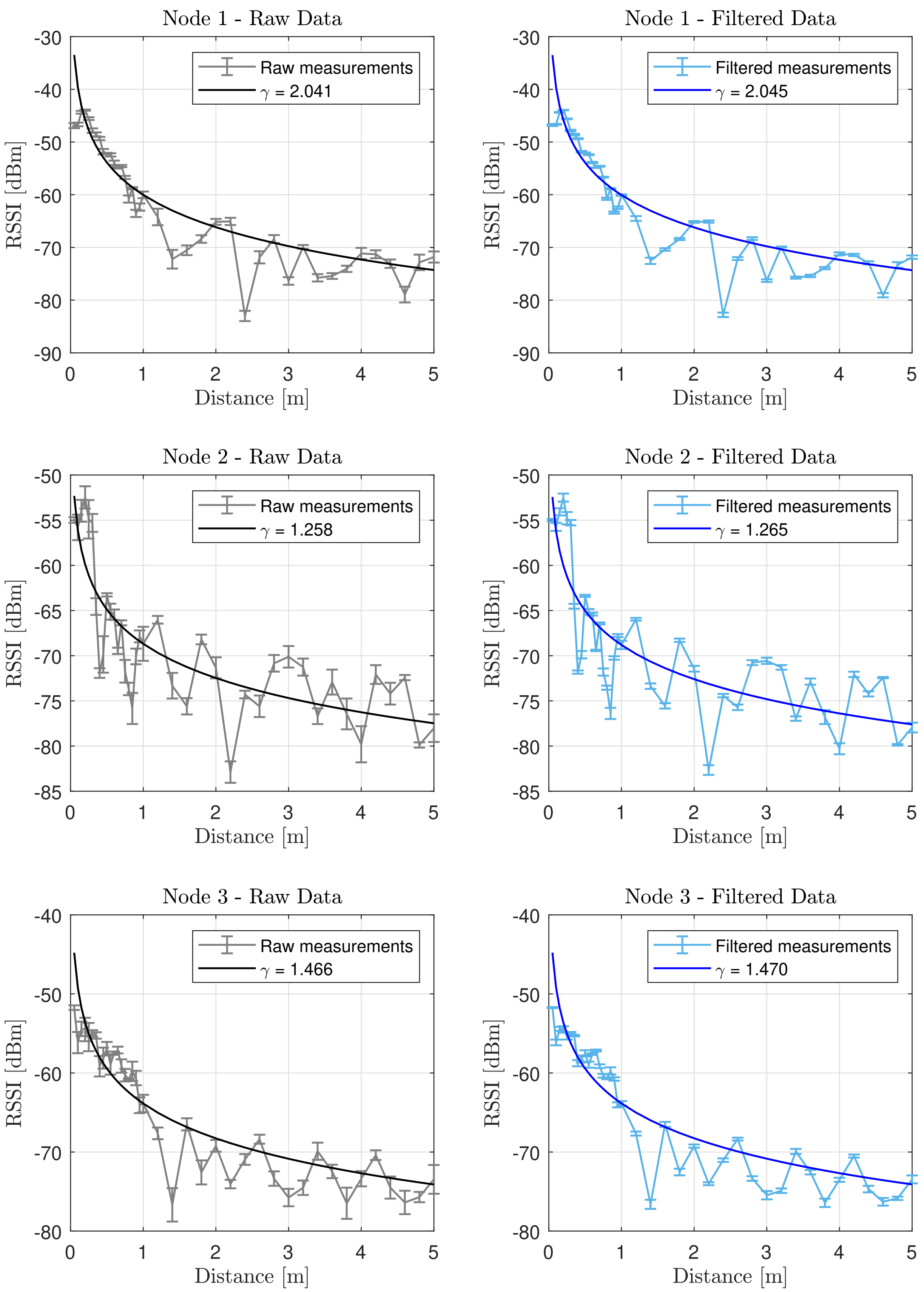


Figure 5: RSSI and γ values for all nodes with 95% CI from TP-B08

Figure 5 shows the average values at each distance with a 95% Confidence Interval (CI) error bar for each node. A $\Gamma(\cdot)$ function with a fitted γ value was added to model the RSSI variation over distance. The optimal γ value for each node was found through performing curve fitting by minimising the sum of squared residuals as shown below.

$$\min_{\gamma \in \mathbb{R}^+} \left\| \Gamma(d_i) - \bar{\Gamma}(d_i) \right\|_2^2 \tag{3}$$

Where $\bar{\Gamma}(\cdot)$ is the mean RSSI value and $\|\cdot\|_2$ is the Euclidean norm. The optimal curve plotted does not contain the AWGN term X_σ as this is a theoretical curve. Observing the CI for both the raw and filtered data, we see that the moving average filter has reduced the small scale fading due to multipathing. However, shadowing effects are still prevalent as seen by the large differences in measured values compared to the optimised curve. It is suspected that nearby obstacles like chairs could cause reflections or scatter the signal, therefore interfering with the main propagating wave that travels directly from the transmitter to receiver.

The results also suggest a possible correlation between RSSI measurements and certain distances across all nodes as they experience similar environmental effects as the experiment maintained the same room layout. However, this is difficult to verify as it could be due to the stochastic nature in RSSI measurements. We can then analyse the error statistics in Table 5.

Table 5: Statistical measures of the errors in RSSI data vs. optimal curve for 3 nodes

Node	Measurement	γ	Central Tendency [dBm]					Percentiles [dBm]				
			μ_ε	σ_ε	ε_{min}	ε_{max}	ε_{range}	25th	50th	75th	90th	IQR
1	Raw	2.04	-1.58	4.32	-15.22	2.56	17.78	-3.72	0.25	1.54	2.23	5.25
	Filtered	2.04	-1.56	4.30	-14.97	2.56	17.54	-3.80	0.35	1.60	2.24	5.39
2	Raw	1.26	-0.18	4.20	-9.89	7.40	17.29	-2.74	0.15	3.57	4.58	6.30
	Filtered	1.26	-0.10	4.21	-9.53	7.51	17.05	-2.64	0.21	3.70	4.65	6.34
3	Raw	1.47	-0.87	3.22	-10.68	4.17	14.69	-2.72	-0.50	1.69	2.57	4.42
	Filtered	1.47	-0.91	3.18	-10.63	3.94	14.57	-2.69	-0.53	1.66	2.45	4.36

Table 5 shows the statistical data for the errors between the measurements and the optimised curve for the different nodes. Most notably, the presence of outliers when taking the RSSI measurements have affected nodes 1 and 3 wherein the mean values exhibit greater errors than compared to node 2. This is inline with the plots from Figure 5 that show points of significant signal attenuation for nodes 1 and 3. Whilst this is also present in node 2, this effect is counteracted by points of greater received power perhaps due to constructive interference as shown by the largest IQR indicating a broader dispersion within the dataset. We can analyse the CDF for the absolute errors in Figure 6.

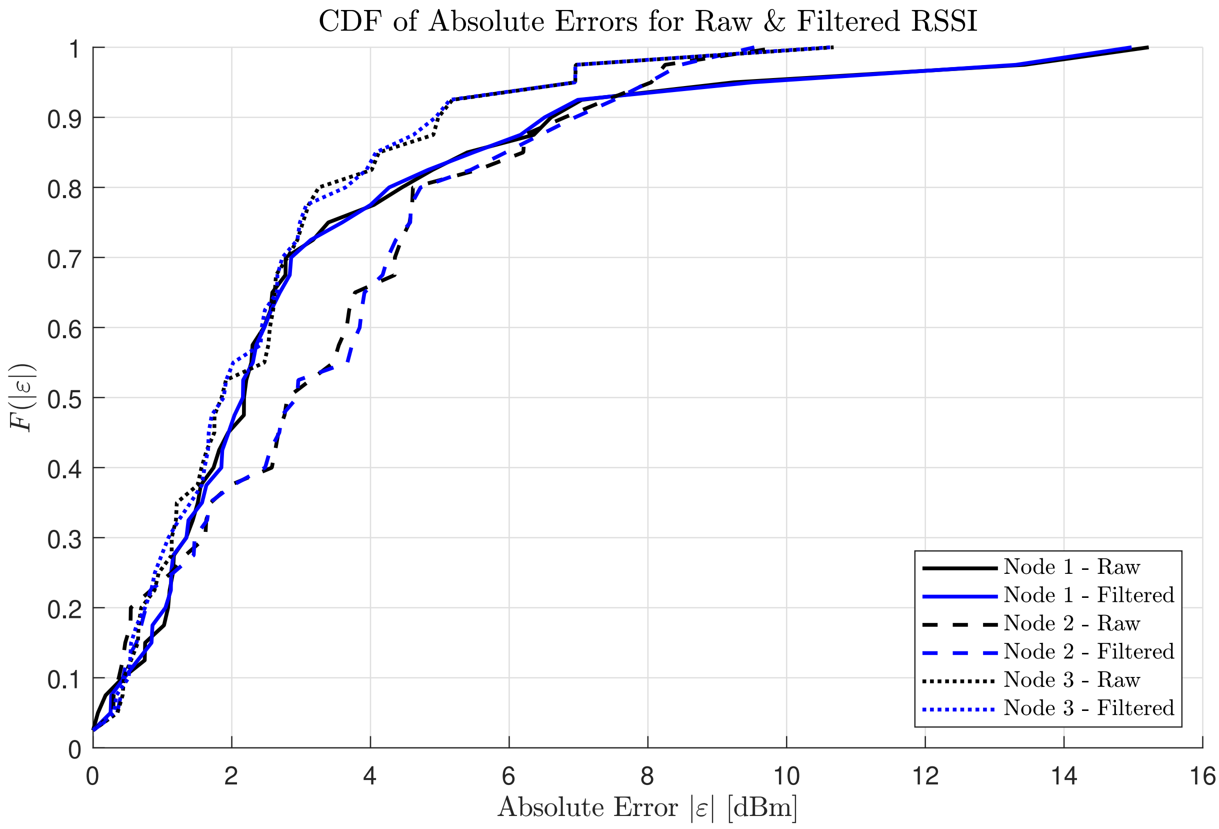


Figure 6: CDF of absolute errors for all 3 nodes from TP-B08

The absolute error CDF, $F(|\varepsilon|)$, gives greater insight to the distribution of the errors in the RSSI data. From our analysis it is evident that despite removing small scale fading from the moving average filter, large scale fading is still a significant factor creating large inaccuracies in RSSI measurements at a given distance. Particularly node 2's upper quartile signified greater errors than nodes 1 & 2 which could be attributed to a systematic error like a different transmitter orientation resulting in non-optimal alignment with the receiver. As we intend to determine position, we must consider the potential inaccuracies of the RSSI measurements. In summary, our work thus far has reduced the AWGN and we can therefore find the distance d using the $\Gamma(\cdot)$ function.

$$d = 10^{\frac{\Gamma(d_0) - \Gamma(d)}{10\gamma}} \tag{4}$$

3.3 Linear Least Squares Trilateration

We intend to compute the true Cartesian position of the unknown receiver, denoted by (x, y). Let us consider the Euclidean distance between each node and the receiver, for N nodes. For a node i, with known position (x_i, y_i), we define the measured Euclidean distance from (x_i, y_i) to (x, y) as follows.

$$\begin{aligned} d_1^2 &= (x - x_1)^2 + (y - y_1)^2 \\ d_2^2 &= (x - x_2)^2 + (y - y_2)^2 \\ &\vdots \\ d_i^2 &= (x - x_i)^2 + (y - y_i)^2 \end{aligned} \tag{5}$$

However, as we intend to find two unknowns, (x, y) , the set of N possible equations has led to an over-determined system wherein a unique solution may not exist. Therefore, we can attempt to select a set of estimated $(\hat{x}, \hat{y})$ that results in a minimisation of an objective function describing our estimated errors. The objective function defined in the paper by Yang et al. [49] is a loss function for the sum of squared residuals.

$$\mathcal{L}(x, y) = \sum_{i=1}^{N} \left[\sqrt{(x - x_i)^2 + (y - y_i)^2} - d_i \right]^2 \tag{6}$$

Where the loss function $\mathcal{L}(\cdot)$ has mapping $\mathcal{L} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$. We can therefore obtain $(\hat{x}, \hat{y})$ via minimisation of $\mathcal{L}(\cdot)$ as follows.

$$(\hat{x}, \hat{y}) = \underset{x,y \in \mathbb{R}}{\arg\min} \sum_{i=1}^{N} \left[\sqrt{(x - x_i)^2 + (y - y_i)^2} - d_i \right]^2 \tag{7}$$

The act of minimising the sum of the squared residuals can be approached as a linear least squares (LSQ) problem. We can therefore characterise the LSQ solution as follows.

$$\min_{\mathbf{x} \in \mathbb{R}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \tag{8}$$

Where $\mathbf{A} \in \mathbb{R}^{m \times n}$ for $m > n$ and $\mathbf{b} \in \mathbb{R}$. The Least Squares Method book by Björck [50] outlines a solution for $\mathbf{x} \in \mathbf{R}$ by utilising the following theorem.

$$\mathbf{A}^{\mathbf{T}} (\mathbf{A}\mathbf{x} - \mathbf{b}) = \mathbf{0} \tag{9}$$

We can therefore rearrange this to obtain a useful form for $\hat{\mathbf{x}}$ as follows.

$$\mathbf{x} = \left(\mathbf{A}^{\mathbf{T}}\mathbf{A}\right)^{-\mathbf{1}}\mathbf{A}^{\mathbf{T}}\mathbf{b} \tag{10}$$

This assumes that $\mathbf{A}$ is of full column rank, therefore $\mathbf{A}^{\mathbf{T}}\mathbf{A}$ is invertible. If this assumption is met, the LSQ solution produces a unique solution for $\mathbf{x}$. To apply this to our proposed trilateration method, we can leverage the intersections of 3 radial distance measurements. When we obtain RSSI measurements, the $\hat{d}$ value describes the noisy radial distance wherein possible $(\hat{x}, \hat{y})$ positions lie on the inscribed circle. However, the common intersections of these 3 circles at a singular point may not be feasible due to fading induced measurement uncertainty. We can represent the residual errors with $\tilde{r} \in \mathbb{R}$. The paper by Cantón et al. [51] provided inspiration for Figure 7 which shows the proposed linear LSQ trilateration method.

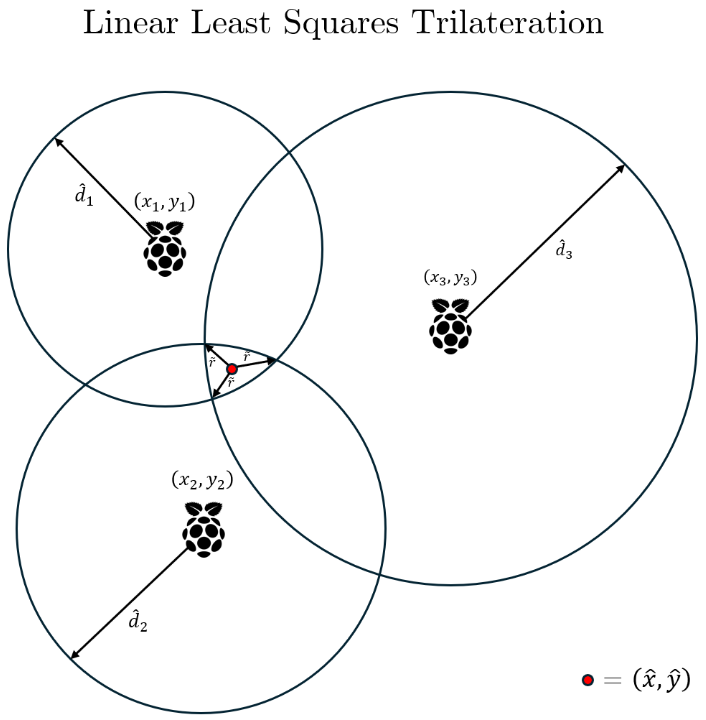


Figure 7: Linear least squares trilateration with measurement residuals for $(\hat{x}, \hat{y})$ position

By referring to Equation 5, we can represent our 3 radial distances as follows.

$$\begin{aligned}\hat{d}_1^2 &= (\hat{x} - x_1)^2 + (\hat{y} - y_1)^2 \\ \hat{d}_2^2 &= (\hat{x} - x_2)^2 + (\hat{y} - y_2)^2 \\ \hat{d}_3^2 &= (\hat{x} - x_3)^2 + (\hat{y} - y_3)^2\end{aligned} \tag{11}$$

We can subtract $\hat{d}_3^2$ from $\hat{d}_1^2$ and $\hat{d}_2^2$ to obtain the following equation.

$$\begin{bmatrix} 2\left(x_1 - x_3\right) & 2\left(y_1 - y_3\right) \\ 2\left(x_2 - x_3\right) & 2\left(y_2 - y_3\right) \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \begin{bmatrix} x_1^2 - x_3^2 + y_1^2 - y_3^2 + \hat{d}_3^2 - \hat{d}_1^2 \\ x_2^2 - x_3^2 + y_2^2 - y_3^2 + \hat{d}_3^2 - \hat{d}_2^2 \end{bmatrix} \tag{12}$$

We can express Equation 12 as a linear matrix in the form as shown below.

$$\mathbf{A}\hat{\mathbf{x}} = \mathbf{b} \tag{13}$$

As this is in the form required for the LSQ problem, we can apply Equation 10 to find a solution for $\hat{\mathbf{x}}$.

$$\begin{aligned} \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \left(\begin{bmatrix} 2(x_1 - x_3) & 2(y_1 - y_3) \\ 2(x_2 - x_3) & 2(y_2 - y_3) \end{bmatrix}^T \begin{bmatrix} 2(x_1 - x_3) & 2(y_1 - y_3) \\ 2(x_2 - x_3) & 2(y_2 - y_3) \end{bmatrix} \right)^{-1} \times \\ \left(\begin{bmatrix} 2(x_1 - x_3) & 2(y_1 - y_3) \\ 2(x_2 - x_3) & 2(y_2 - y_3) \end{bmatrix}^T \begin{bmatrix} x_1^2 - x_3^2 + y_1^2 - y_3^2 + \hat{d}_3^2 - \hat{d}_1^2 \\ x_2^2 - x_3^2 + y_2^2 - y_3^2 + \hat{d}_3^2 - \hat{d}_2^2 \end{bmatrix} \right) \end{aligned} \tag{14}$$

This LSQ solution assumes that $\mathbf{A^T A}$ is well conditioned to prevent numerical inaccuracies. For a singular matrix, the condition number is infinite. Cline et al. [52] describes the computation of the condition number of a matrix $\mathbf{M}$ as $\text{cond}(\mathbf{M}) = \|\mathbf{M}\| \cdot \|\mathbf{M^{-1}}\|$. Therefore, assuming that $\mathbf{A^T A}$ is non-singular, we require the condition number to be less than 10^p where p is the digits of precision. In Arduino IDE, the `float` data type can hold 6-7 decimal digits of precision so we can set our threshold as follows.

$$\text{cond}(\mathbf{A^T A}) = \left\|\mathbf{A^T A}\right\|_2 \cdot \left\| \left[\mathbf{A^T A}\right]^{\mathbf{-1}} \right\|_2 \leq 10^3 \tag{15}$$

It is clear that this position estimation requires 3 intersections from all circles so we can establish a conditional statement that checks if the sum of the measured distances for two nodes is greater than the sum of the distances between the two nodes. Therefore, we can express this condition for all nodes as follows.

$$\forall (i,j) \in \{1,2,3\},\ i \neq j : (\hat{d}_i + \hat{d}_j) \geq \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{16}$$

We can now express the logic governing the trilateration system. It employs a flag based system to verify all target UUIDs are being advertised with valid intersecting circles.

3.3.1 Empirical Outlier Rejection

Despite the LSQ procedure providing optimal solutions, the presence of outliers could skew our measurement distribution. Therefore, we propose an outlier rejection method based on an empirical procedure. After reference node deployment, you can perform the trilateration experiments using the RSSI to distance Equation 4. Observing the results for x and y, we calculate the empirical CDF $F_{i,x}$ for x and $F_{i,y}$ for y for i-th experiment. Then the 75th percentile is $F_{i,x}^{-1}(0.75)$ and $F_{i,y}^{-1}(0.75)$. If we repeated the trilateration experiment N times, we can search for the maximum $F^{-1}(0.75)$ value then compute the floor function, $\lfloor \cdot \rfloor$, to obtain an outlier boundary $\mathbf{\Omega}$.

$$\mathbf{\Omega} = \begin{bmatrix} \Omega_x \\ \Omega_y \end{bmatrix} = \begin{bmatrix} \lfloor \max_{i \in \{1,2,\cdots N\}} F_{i,x}^{-1}(0.75) \rfloor \\ \lfloor \max_{i \in \{1,2,\cdots N\}} F_{i,y}^{-1}(0.75) \rfloor \end{bmatrix} \tag{17}$$

The rationale behind this was motivated by the fact that the upper quartile would generally be more robust to outliers. By establishing a boundary at $F^{-1}(0.75)$, we discard the extremely noisy measurements without losing a significant proportion of our data. By searching for the maximum $F^{-1}(0.75)$ for all experiments, we ensure for the "worst case" that we do not discard a large proportion of the data. The $\lfloor \cdot \rfloor$ function was chosen to help further reduce outlier presence. To implement this practically, we describe the logical condition as follows.

$$\forall (x, y) \in \mathbb{R} : (|x| < \Omega_x) \land (|y| < \Omega_y) \tag{18}$$

We can therefore describe the complete trilateration method using Algorithm 3. To remain computationally efficient, we computed $\left(\mathbf{A^T A}\right)^{\mathbf{-1}} \mathbf{A^T}$ with relevant condition number checks inside the `void setup()` function in Arduino IDE. This is because $\left(\mathbf{A^T A}\right)^{\mathbf{-1}} \mathbf{A^T}$ is only dependent on the coordinates of the reference nodes at their relative distances.

Algorithm 3: BLE Localization: RSSI to Distance Conversion and Trilateration

```
Input  : RSSI Measurements for all nodes: $\Gamma^{(i)}(d)$ for $i \in \{1,2,3\}$
         Path Loss Exponents for all nodes: $\gamma^{(i)}$ for $i \in \{1,2,3\}$
         Reference RSSI at 1m: $\Gamma^{(i)}(d_0)$ for $i \in \{1,2,3\}$
         Reference node coordinates for all nodes: $(x_i, y_i)$ for $i \in \{1,2,3\}$
         Reference distances between each node: $d_i$ for $i \in \{1,2,3\}$
Output: Estimated position of the receiver: $(\hat{x}, \hat{y})$
Initialise:
    $\left(\mathbf{A^T A}\right)^{-1} \mathbf{A^T}$;                // Computed in setup for efficiency
Function RSSIToDistance($\Gamma^{(i)}(d)$, $\gamma^{(i)}$, $\Gamma^{(i)}(d_0)$):
    for $i \in \{1,2,3\}$ do
        if all UUIDs detected then
            $d^{(i)} \leftarrow 10^{\left(\frac{\Gamma^{(i)}(d_0)-\Gamma^{(i)}(d)}{10\gamma^{(i)}}\right)}$;
        end if
    end for
    return $d^{(i)}$;    // Computed distance between each node & receiver
end
Function Trilateration($d^{(i)}$, $x_i$, $y_i$):
    $\hat{\mathbf{x}} = \left(\mathbf{A^T A}\right)^{-1} \mathbf{A^T b}$;                // Compute LSQ trilateration
    return $(\hat{x}, \hat{y})$;
end
```

For our trilateration experiments, we wrote 200 paired (x, y) measurements onto an SD card and we conducted our tests in Foster Court (FC) in room 132. Unfortunately, the original Arduino MCU chip was burnt and with limited room availability, we had to assume an average γ value of 1.6 for all nodes. We radially calibrated for $\Gamma(d_0)$ where radial calibration prevented incorrect $\Gamma(d_0)$ values arising from localised RSSI anomalies. Due to the aforementioned challenges and time constraints, re-calibration for γ values was not feasible. To obtain localisation error metrics, we played an in-ear metronome at 110 beats per minute (BPM) to maintain a fixed cadence. This accompanied with fixed tape measurements at 0.7 m spacing allowed for a ground truth reference. The path taken was a 3.50×4.60 m^2 rectangular grid. The LSQ trilateration with & without outlier rejection is shown in Figure 8.

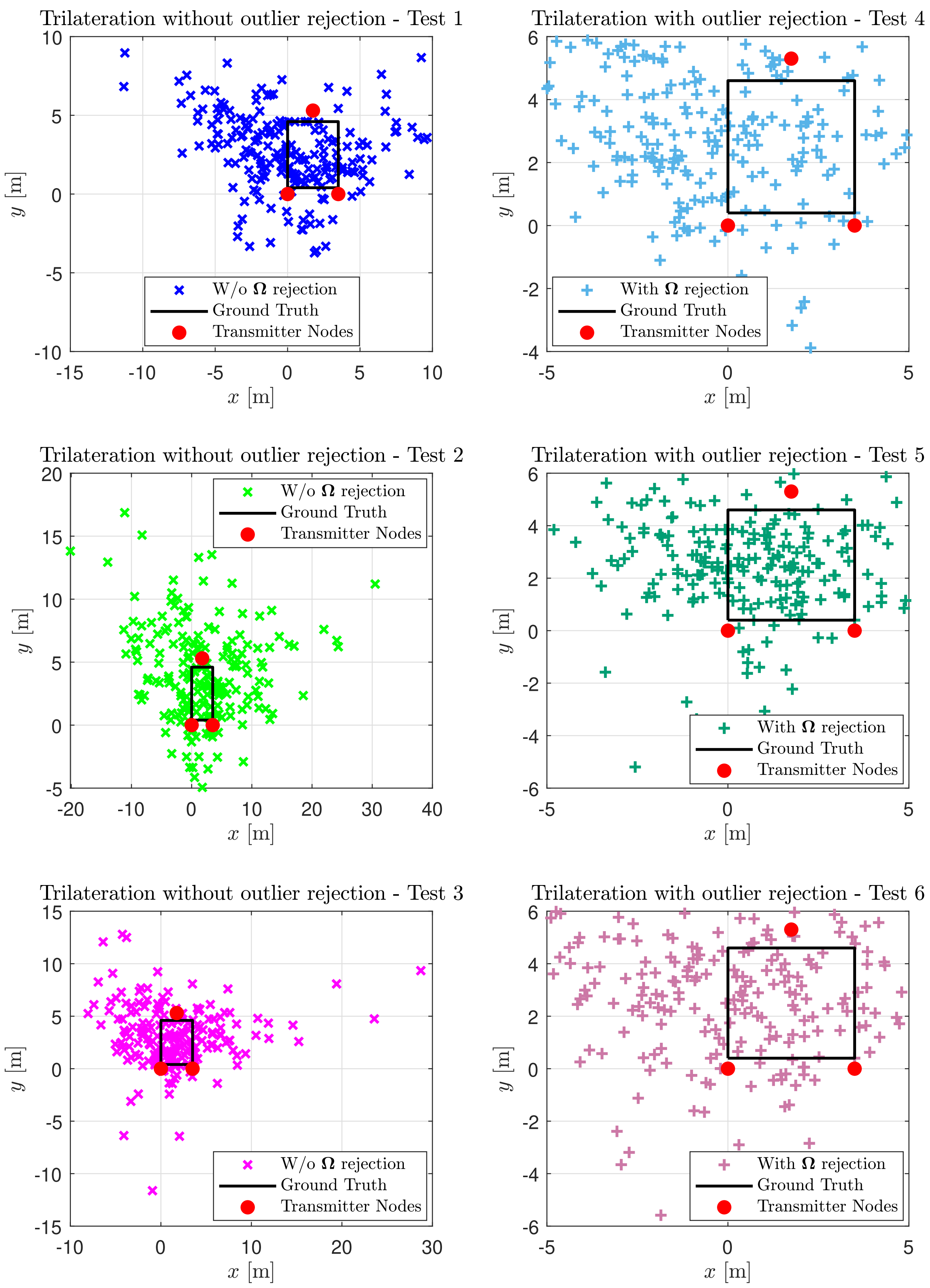


Figure 8: Least squares trilateration with & without outlier rejection taken in FC-132

Upon obtaining the quartile information for the unfiltered data, we obtained $\Omega_x = 6.0$ & $\Omega_y = 5.0$ and this was plotted alongside the unfiltered data as seen in Figure 8. Despite the general clustering of LSQ solutions near the ground truth, the unfiltered data exhibited outliers as expected. We can observe that the $\boldsymbol{\Omega}$ rejection method has removed outliers but data points are still almost randomly distributed about the ground truth. However, the rejection data shows an almost biased clustering towards the top left corner. Observing the unfiltered data, this is also inline with our observation where outlier presence seems prominent in the upper-left corner. Several factors could have caused this.

The lack of calibration for this new room and the use of a different receiver could change the log-distance relationship model via γ. Although $\gamma = 1.6$ was based on the average γ values obtained during TP-B08 calibration and typical indoor LoS values [48], changes in the testing environment and the usage of different antennas would affect the measured RSSI. However, a general expectation for a severe lack of calibration would be a much larger spread far exceeding the ground truth values. If we observe test 5, results are clustered centrally around the grid which would not necessarily occur if our assumed γ and measured $\Gamma(d_0)$ deviated greatly from their true values. Whilst γ calibration is important, $\Gamma(d_0)$ mathematically would have a greater influence on the log-distance model for $d < 5$ m & assuming $\gamma \in [1.6, 1.8]$.

Another factor to consider is the presence of obstacles blocking LoS. Whilst chairs were present in the room, their NLoS effects were mitigated by ensuring that the transmitters were orientated slightly upwards and pointed towards the general walking path. Throughout the path, no obstacles obstructed LoS throughout testing.

A plausible reason could be that for both upper left & right corners, they are the furthest points away from the transmitters. For the bottom corners, the two transmitters are at a close proximity with the receiver throughout the bottom path hence distance measurements would be more accurate. This observed behaviour could be due to the $\Gamma(\cdot)$ function where at closer distances, changes in RSSI values would not lead to large changes in the computed distance. Therefore due to beacon deployment, the upper corners could exhibit slight changes in RSSI leading to larger variations in distance measurements.

We compute the total error as: $\varepsilon = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2}$. We defined a MATLAB trajectory with the same path at time steps of $\frac{60}{110}$ s to match the user's cadence. We search for the time step where this closely matches the time at $(\hat{x}, \hat{y})$ then we compute the errors. Table 6 shows the statistics for each test.

Table 6: Statistical measures of the errors in Least Squares Trilateration

Test	Central Tendency [m]					Percentiles [m]				
	μ_ε	σ_ε	ε_{min}	ε_{max}	ε_{range}	25th	50th	75th	90th	IQR
1	4.5968	2.3652	0	14.2397	14.2397	2.9656	4.1640	6.0566	7.5917	3.0910
2	6.6397	4.5177	0	29.1500	29.1500	3.5748	5.8000	8.9133	12.3306	5.3385
3	5.7018	3.7009	0	30.0793	30.0793	3.4662	4.9200	7.0047	9.7070	3.5384
4	4.4774	1.8857	0	9.8723	9.8723	3.2035	4.7048	5.6944	6.6564	2.4909
5	3.3628	1.7101	0	8.2486	8.2486	2.0154	3.2604	4.5144	5.6133	2.4990
6	3.8550	2.0748	0	10.2851	10.2851	2.2760	3.4617	5.3394	6.8503	3.0634

The ε_{min} value was zero as we saved the measurement at the same starting point. For the unfiltered data, the $F^{-1}(0.90)$ with a large ε_{max} verify the presence of outliers. For the $\mathbf{\Omega}$ rejection tests, the significant reduction in ε_{max} and ε_{range} indicates the successful removal of large outliers. The general reduction in μ_ε for $\mathbf{\Omega}$ tests is expected due to its sensitivity to outliers. Figure 9 contains the $F(\varepsilon)$ plot for all tests which reaffirms our analysis & observations of successful outlier rejection.

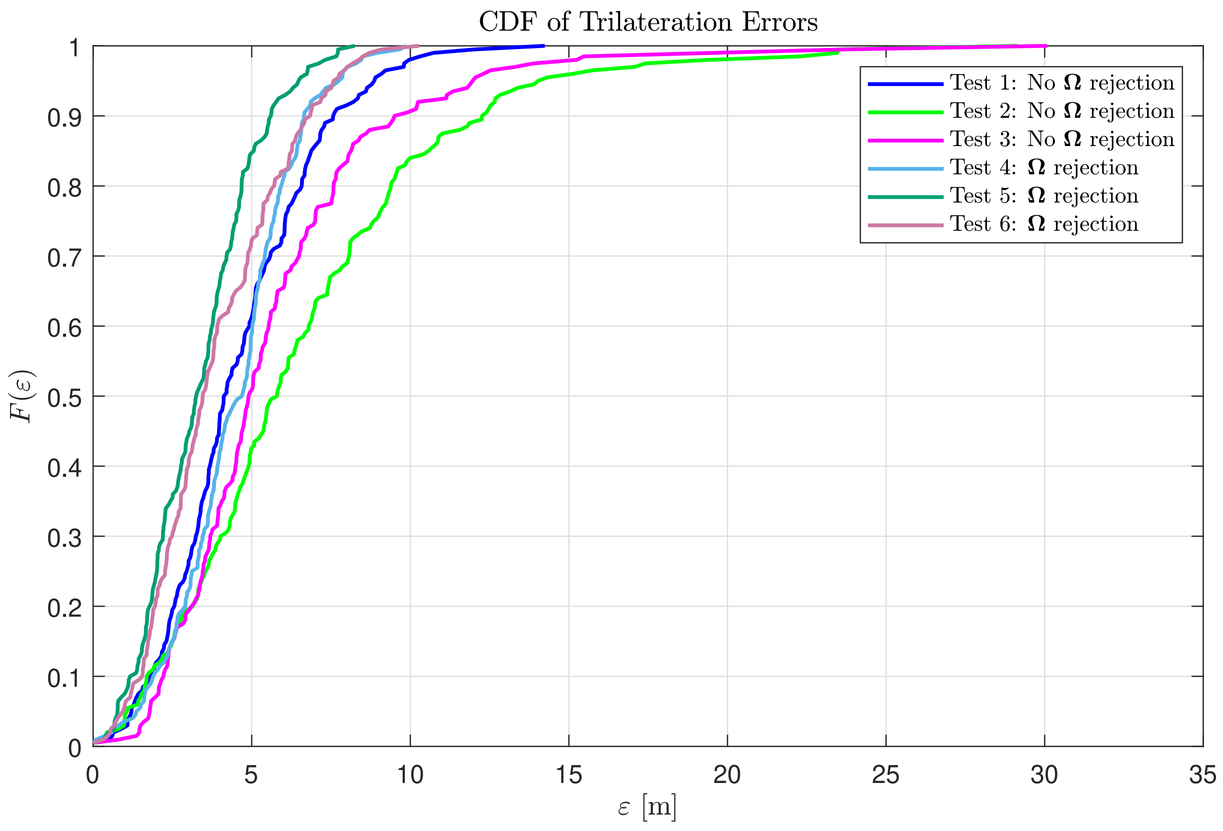


Figure 9: CDF of trilateration errors for tests with & without outlier rejection

Chapter 4. Inertial Navigation

4.1 Inertial Measurement Unit Calibration

The short term updates provided by the PDR solution can be obtained from the IMU. The accelerometer has a rate of 125 Hz with a range of $\pm 8g$. The gyroscope has a rate of 100 Hz with a range of $\pm 2000°/\text{s}$ [36]. For both sensors, Lv et al. [53] describes the mathematical equation that relates raw measurements to calibrated readings. Equation 19 describes the accelerometer calibration.

$$\begin{bmatrix} a_{x_{cal}} \\ a_{y_{cal}} \\ a_{z_{cal}} \end{bmatrix} = \begin{bmatrix} S_{a_x} & 0 & 0 \\ 0 & S_{a_y} & 0 \\ 0 & 0 & S_{a_z} \end{bmatrix} \begin{bmatrix} a_{x_{raw}} - b_{a_x} \\ a_{y_{raw}} - b_{a_y} \\ a_{z_{raw}} - b_{a_z} \end{bmatrix} \tag{19}$$

Which can be rewritten as shown in Equation 20.

$$\vec{\boldsymbol{a}}_{cal} = \mathbf{S}_a \left(\vec{\boldsymbol{a}}_{raw} - \vec{\boldsymbol{b}}_a \right) \tag{20}$$

Where $\vec{\boldsymbol{a}}_{cal}$ is the calibrated acceleration matrix, $\mathbf{S}_a$ is the accelerometer scaling factor matrix, $\vec{\boldsymbol{a}}_{raw}$ is the raw acceleration matrix, and $\vec{\boldsymbol{b}}_a$ is the zero-g offset bias matrix. We have conducted a static calibration procedure where the accelerometer measurements are taken in 6 orientations. The MathWorks documentation [54] provides a good visualisation of the different orientations needed for static calibration as shown in Figure 10. We note that for each axis direction $\pm\hat{\boldsymbol{x}}$, $\pm\hat{\boldsymbol{y}}$, and $\pm\hat{\boldsymbol{z}}$, we require the gravity vector, denoted $\vec{\boldsymbol{g}}$, to be in line with each axis direction hence 6 orientations.

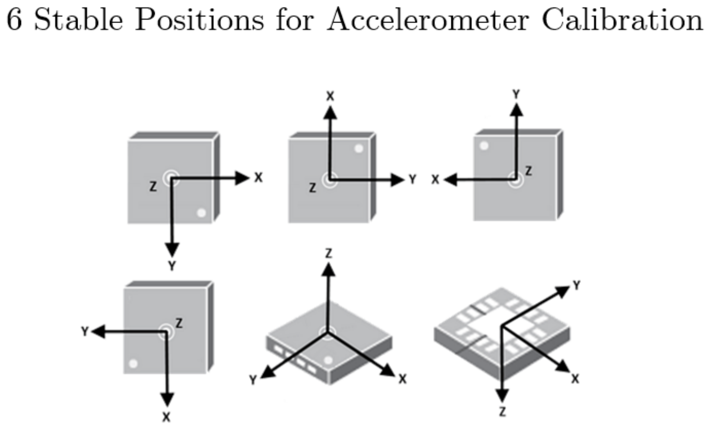
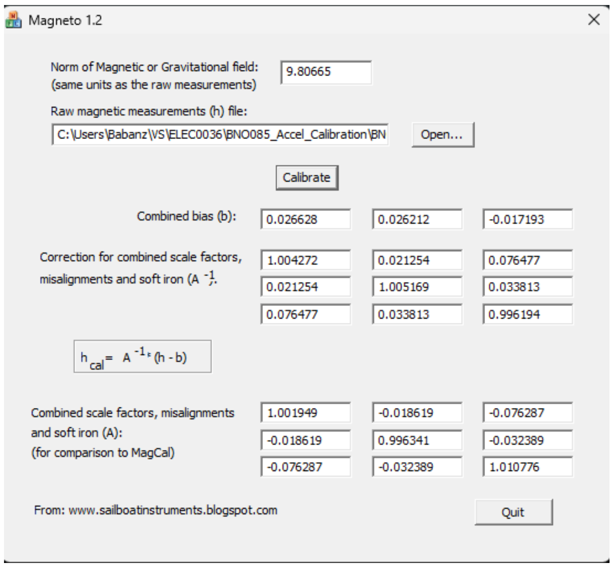


Figure 10: Static calibration for accelerometer using 6 orientations

For calibration we mounted the Arduino onto a cuboid shaped device for stability and we took measurements using similar code as the RSSI measurements. In order to ensure that $\vec{\boldsymbol{g}}$ was parallel to the axis of measurement, the cuboid device was also placed next to a wall for a perpendicular reference to ground.

We utilised the open source Magneto v1.2 software [55] for calibration. We uploaded the tab-delimited file containing the acceleration measurements and specified the norm of the gravitational field to be $\|\vec{\boldsymbol{g}}\|_2 = 9.80665 \text{ m s}^{-2}$. Figure 11 shows a combined bias result corresponding to $\vec{\boldsymbol{b}}_a = \begin{bmatrix} 0.026628 & 0.026212 & -0.017193 \end{bmatrix}^T$. The combined scale factors result approximates to $\mathbf{S}_a \approx \mathbf{I}$ where $\mathbf{I}$ is identity matrix. These are sensible results as we expect an offset shift for acceleration readings but there should not be any scale factor in the applied to the axes like we would see in a magnetometer.


Magneto 1.2

Norm of Magnetic or Gravitational field: (same units as the raw measurements) 9.80665

Raw magnetic measurements (h) file: C:\Users\Babanz\VS\ELEC0036\BNO085_Accel_Calibration\BN

Open...

Calibrate

Combined bias (b):	0.026628	0.026212	-0.017193
Correction for combined scale factors, misalignments and soft iron (A $^{-1}$):	1.004272	0.021254	0.076477
	0.021254	1.005169	0.033813
	0.076477	0.033813	0.996194

$h_{cal} = A^{-1} * (h - b)$

Combined scale factors, misalignments and soft iron (A): (for comparison to MagCal)	1.001949	-0.018619	-0.076287
	-0.018619	0.996341	-0.032389
	-0.076287	-0.032389	1.010776

From: www.sailboatinstruments.blogspot.com

Quit

Figure 11: Magneto v1.2 calibration results

To verify calibration, we compare the linear acceleration of the raw and calibrated data when Arduino is flat on a table. We define this as the difference between the Euclidean norm of the acceleration vector and the $\vec{\boldsymbol{g}}$ vector. This is described mathematically as shown in Equation 21.

$$a_{linear} = \|\vec{\boldsymbol{a}}\|_2 - \|\vec{\boldsymbol{g}}\|_2 \tag{21}$$

Figure 12 shows the linear acceleration data obtained for both raw and calibrated accelerometer readings, taking 1000 measurements for both. We expect linear acceleration to be close to zero as we have also removed the gravity component from our readings using Equation 21. From the Magneto results, the calibration data consisted of applying the bias correction matrix whilst assuming identity scaling factor.

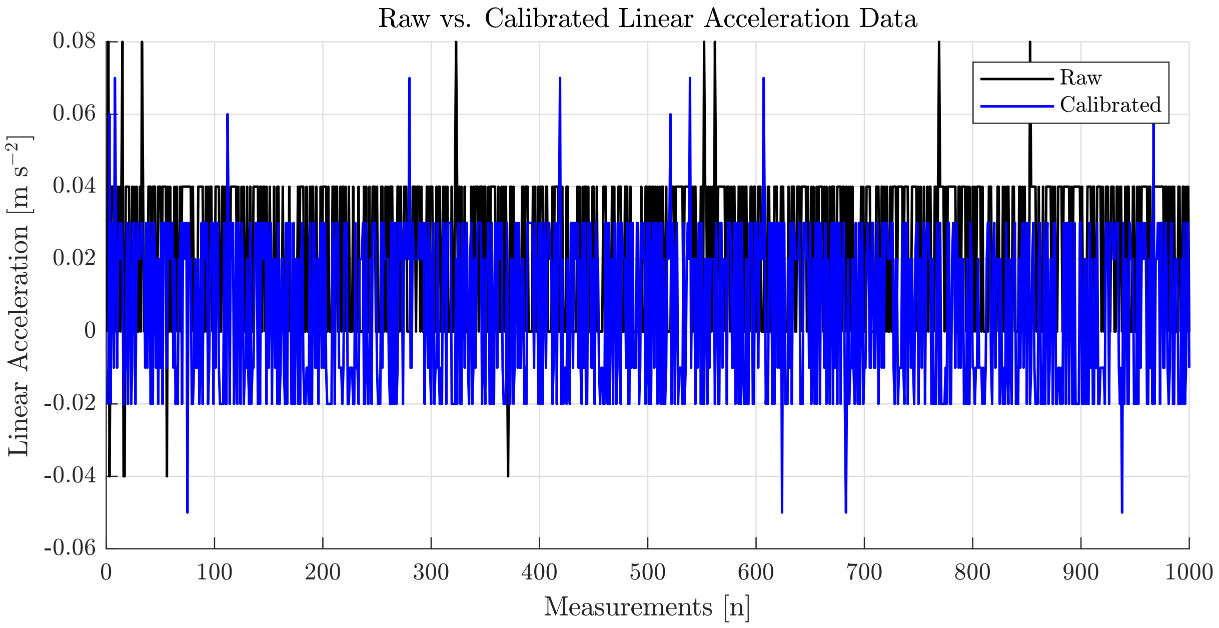


Figure 12: Calibrated vs raw linear acceleration measurements taken when stationary

We can numerically analyse the efficacy of the calibration by observing the statistics and the root mean square error (RMSE) shown in Table 7. We observe from the mean values that the calibration data was more closely centered at zero than the raw values. From the RMSE, the reduction in error for the calibrated results indicate a corrected bias alignment. From our data, it is clear that the bias correction simply removes some systematic error in the linear acceleration data which is the intended behaviour, concluding successful zero-g bias correction calibration.

Table 7: Statistical measures of accelerometer data

Measurement	Central Tendency [m s^{-2}]						Dispersion [m s^{-2}]				
	Mean	Std Dev	Min	Max	Range	RMSE	25th	50th	75th	90th	IQR
Raw	0.0209	0.0210	-0.0400	0.0800	0.1200	0.0296	0	0.0400	0.0400	0.0400	0.0400
Calibrated	0.0066	0.0223	-0.0500	0.0700	0.1200	0.0232	-0.0200	0.0200	0.0300	0.0300	0.0500

For the gyroscope, the BNO085 provides calibrated and uncalibrated angular velocity outputs in rad s^{-1}. We shall refer to the our proposed calibration results as the calibration results, whilst the IMU calibrated results are referred to as IMU calibration results. The equation for the gyroscope is shown in Equation 22.

$$\vec{\boldsymbol{\omega}}_{cal} = \mathbf{S}_{\omega}\left(\vec{\boldsymbol{\omega}}_{raw} - \vec{\boldsymbol{b}}_{\omega}\right) \tag{22}$$

As we lack proper calibration apparatus, we assume that: $\mathbf{S}_\omega \approx \mathbf{I}$. We placed the Arduino on a flat surface taking 1000 averaged measurements for each axis. This average value is the offset bias for the axis and we would then subtract this from the raw axis measurements to obtain the proposed calibration results. For the IMU calibrated method, [36] when the IMU detects minimal angular velocity, the DMP regards the IMU as stationary and dynamically computes the zero-rate offset bias. As the sensor data was subject to significant noise, a filter was applied after data collection to smooth the results for easier visualisation and analysis. A Gaussian-weighted moving average filter with window size 40 was applied in MATLAB show in Figure 13.

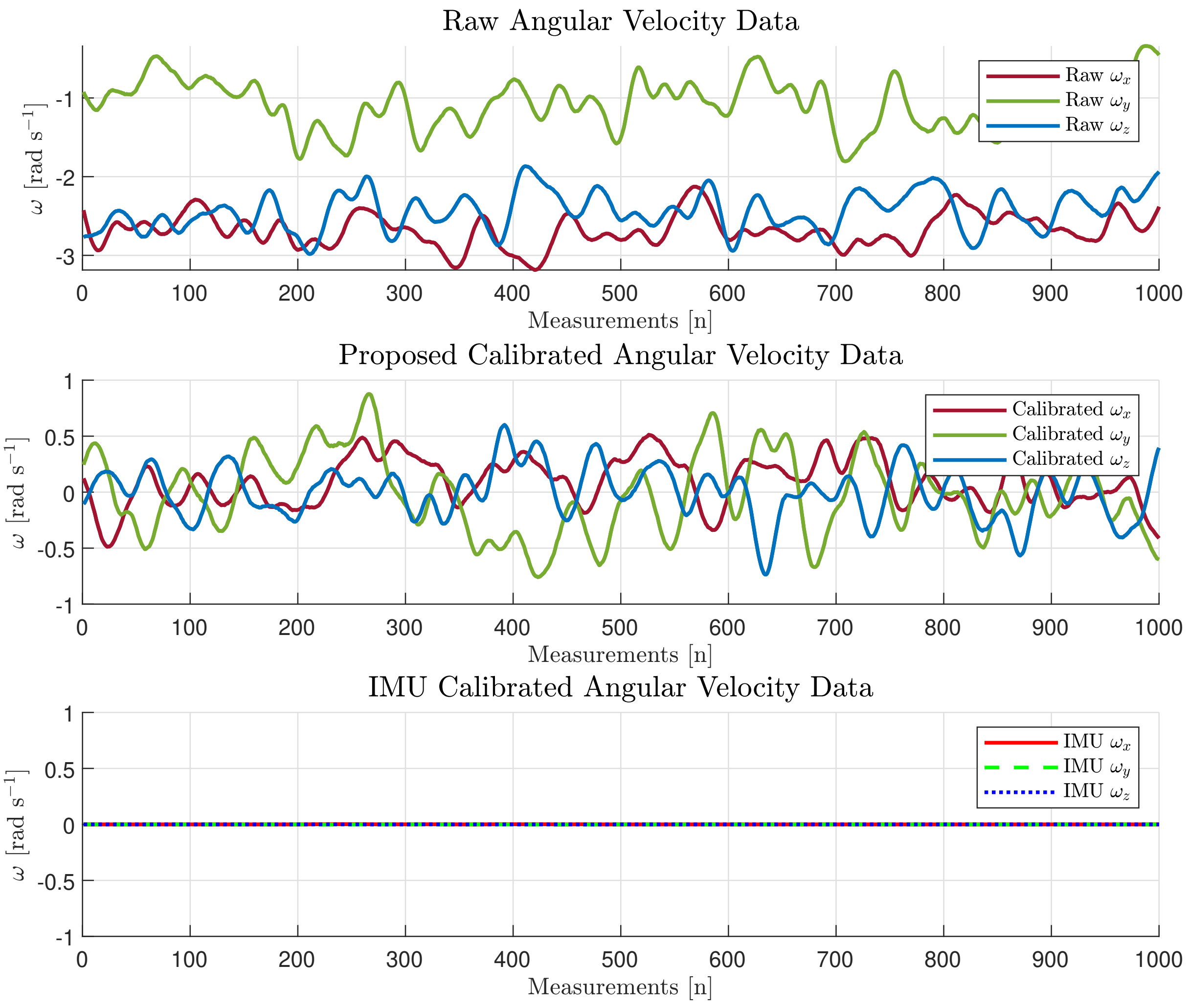


Figure 13: Calibrated vs raw angular velocity measurements taken when on flat surface

Figure 13 shows the angular velocity measurements taken from our sensor for the uncalibrated, proposed, and IMU calibrated methods. The raw data showed a clear offset from 0 for all 3 axes indicating the presence of zero-rate bias offset. Our calibrated results show a large improvement in correcting the raw data offsets for all 3 axes. However we note that the standard deviations for our calibration was not corrected as we had not applied a filtering technique during preprocessing. The IMU data showed excellent calibration results as the DMP dynamically calibrates the gyroscope for any zero-rate offsets when the IMU is relatively stable [36]. There also appears to be a corrective element conducted in the DMP that removes a significant amount of noise in the IMU calibrated readings. It is unclear how the DMP firmware dynamically calibrates out the noise as this information is proprietary. We can statistically compare the different methods as shown in Table 8.

Table 8: Statistical comparison between uncalibrated vs calibrated angular velocity

Measurement	Mean [rad s^{-1}]			Standard deviation [rad s^{-1}]			RMSE [rad s^{-1}]		
	ω_x	ω_y	ω_z	ω_x	ω_y	ω_z	ω_x	ω_y	ω_z
Raw	-2.6780	-1.0740	-2.4390	1.0637	1.5668	1.2206	2.8813	1.8989	2.7271
Proposed	0.0940	-0.0210	0.0010	1.0315	1.5193	1.2417	1.0315	1.5187	1.2410
IMU	0.0002	-0.00004	-0.000006	0.0009	0.0006	0.0006	0.0009	0.0006	0.0006

For our analysis, we have only considered the mean, standard deviation, and RMSE for all 3 axis. As mentioned previously, the RMSE had decreased for our proposed calibration method and the readings for all 3 axes are closer to the intended zero rate. It is clear from the statistical data that the IMU calibration is far superior to the raw and calibration methods as seen by a reduction in orders of magnitude for standard deviation and RMSE values. For future work, we will use the IMU calibrated gyroscope values.

4.2 Quaternion Orientation Fusion

Upon completion of calibration, we can now use the accelerometer and gyroscope data for heading estimation. In order to determine the user's heading, we need to determine the orientation of the mobile receiver device in three-dimensional space. This can be done by fusing the gravity information provided by the accelerometer with the rate of rotation from the gyroscope. This rate of rotation can then be integrated over time yielding angular displacement about an arbitrary starting point. To accurately determine the orientation from these measurements, it is beneficial to use quaternions.

Quaternions are four-dimensional complex numbers that can describe the rotation or orientation of a body or coordinate system in three-dimensional space. They avoid common issues like gimbal lock for Euler angles where the alignment of two of the three rotation axes leads to a loss of rotational degrees of freedom. They are also computationally efficient as a typical Direction Cosine Matrix (DCM) requires 9 elements to describe rotation whereas a quaternion can do so using 4 elements [56]. To greater understand quaternions, we will refer to the paper by Sebastian O.H. Madgwick [57].

$$\boldsymbol{q} = q_w + \boldsymbol{i}q_x + \boldsymbol{j}q_y + \boldsymbol{k}q_z \tag{23}$$

Equation 23 describes the general form of a quaternion where $q_w, q_x, q_y, q_z \in \mathbb{R}$ and $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are mutually orthogonal imaginary unit basis vectors. For $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$, they must satisfy the relation: $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1$. If we denote our quaternion as $\boldsymbol{q} = \langle q_w, q_x, q_y, q_z \rangle$, we can describe the quaternion conjugate as follows.

$$\boldsymbol{q}^* = \langle q_w, -q_x, -q_y, -q_z \rangle \tag{24}$$

We can multiply two quaternions through the Hamilton product, denoted by $\otimes$. If we consider two quaternions $\boldsymbol{a} = \langle a_1, a_2, a_3, a_4 \rangle$ and $\boldsymbol{b} = \langle b_1, b_2, b_3, b_4 \rangle$, the Hamilton product for $\boldsymbol{a} \otimes \boldsymbol{b}$ is as follows.

$$\begin{aligned} \boldsymbol{a} \otimes \boldsymbol{b} = &\ (a_1b_1 - a_2b_2 - a_3b_3 - a_4b_4) \\ &+ (a_1b_2 + a_2b_1 + a_3b_4 - a_4b_3)\,\boldsymbol{i} \\ &+ (a_1b_3 - a_2b_4 + a_3b_1 + a_4b_2)\,\boldsymbol{j} \\ &+ (a_1b_4 + a_2b_3 - a_3b_2 + a_4b_1)\,\boldsymbol{k} \end{aligned} \tag{25}$$

Let us consider a unit quaternion, denoted $\hat{\boldsymbol{q}}$, wherein $q_w^2 + q_x^2 + q_y^2 + q_z^2 = 1$ corresponds to unit length. Importantly, all rotation quaternions must be unit quaternions. We can describe a rotation from frame B to frame A by an angle θ around a rotation axis ${}^{A}\hat{\boldsymbol{r}}$ as follows.

$${}^{A}_{B}\hat{\boldsymbol{q}} = \langle q_w, q_x, q_y, q_z \rangle = \left\langle \cos\frac{\theta}{2}, -r_x \sin\frac{\theta}{2}, -r_y \sin\frac{\theta}{2}, -r_z \sin\frac{\theta}{2} \right\rangle \tag{26}$$

Notably, the quaternion conjugate during rotation is geometrically interpreted as swapping the relative frames which can be described as follows.

$${}^{A}_{B}\hat{\boldsymbol{q}}^* = {}^{B}_{A}\hat{\boldsymbol{q}} = \langle q_w, -q_x, -q_y, -q_z \rangle \tag{27}$$

We can also describe compounded rotations through the Hamilton product. If we consider

${}^{A}_{C}\hat{\boldsymbol{q}}$, this can be interpreted as a compounded rotation of ${}^{B}_{C}\hat{\boldsymbol{q}}$ and ${}^{A}_{B}\hat{\boldsymbol{q}}$.

$$ {}^{A}_{C}\hat{\boldsymbol{q}} = {}^{B}_{C}\hat{\boldsymbol{q}} \otimes {}^{A}_{B}\hat{\boldsymbol{q}} \tag{28} $$

As we can describe three-dimensional rotation using quaternions, we can similarly describe a three-dimensional vector as a pure quaternion. A quaternion vector $\vec{\boldsymbol{v}}$ is a pure quaternion wherein the scalar component (first element) of the quaternion is zero such that $\vec{\boldsymbol{v}} = \langle 0, v_x, v_y, v_z \rangle$. We note that $\vec{\boldsymbol{v}}$ must not necessarily be of unit length. Considering $\vec{\boldsymbol{v}}$ in frame A and B and ${}^{A}\vec{\boldsymbol{v}}$ and ${}^{B}\vec{\boldsymbol{v}}$ respectively, we can describe the rotation of this vector as follows.

$$ {}^{B}\vec{\boldsymbol{v}} = {}^{A}_{B}\hat{\boldsymbol{q}} \otimes {}^{A}\vec{\boldsymbol{v}} \otimes {}^{A}_{B}\hat{\boldsymbol{q}}^{*} \tag{29} $$

Let us consider a practical instance for Equation 23. If we had the gravity vector, described in quaternion space denoted $\mathbb{H}$, we would obtain the pure quaternion: $\vec{\boldsymbol{g}} = \langle 0, 0, 0, g \rangle$. Using a global reference frame of East-North-Up (ENU), this typically implies that $\vec{\boldsymbol{g}}$ would be negative due to downwards direction. However, the IMU measures the acceleration that is associated to forces acting upon it which would in most cases be the reaction force. For a flat table, the reaction force exerted onto the IMU is therefore equal to gravity and would have positive value. Now consider an initial orientation shown in Figure 14.

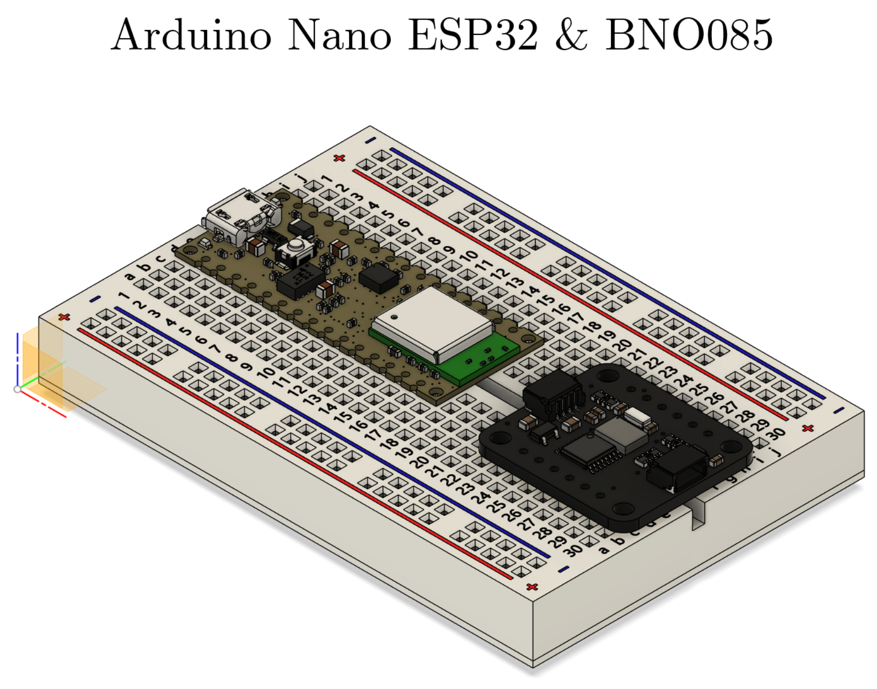


Figure 14: Mobile BLE receiver device initial orientation matching ENU frame

If we define the initial orientation of the device as equal to the ENU frame, we can define the ENU quaternion orientation as the identity quaternion where: ${}^{ENU}\hat{\boldsymbol{q}} = \langle 1, 0, 0, 0 \rangle$. Now for any arbitrary device orientation, we will represent this as the quaternion orientation in the IMU frame where: ${}^{IMU}\hat{\boldsymbol{q}} = \left\langle {}^{IMU}q_w, {}^{IMU}q_x, {}^{IMU}q_y, {}^{IMU}q_z \right\rangle$. As the IMU gives acceleration readings in 3 axes, we can express the acceleration vector readings in quaternion space: $\vec{\boldsymbol{a}} = \langle 0, a_x, a_y, a_z \rangle$ where $\vec{\boldsymbol{a}} \in \mathbb{H}$. For an arbitrary orientation, the gravity component would not necessarily be parallel to the z axis so we can perform the rotation from IMU frame to ENU frame as follows.

$$ {}^{ENU}\vec{\boldsymbol{a}} = {}^{IMU}_{ENU}\hat{\boldsymbol{q}} \otimes {}^{IMU}\vec{\boldsymbol{a}} \otimes {}^{IMU}_{ENU}\hat{\boldsymbol{q}}^{*} \tag{30} $$

We can now correct for the gravity component by subtracting this term from the ENU acceleration vector.

$$ {}^{ENU}\vec{\boldsymbol{\alpha}} = {}^{ENU}\vec{\boldsymbol{a}} - {}^{ENU}\vec{\boldsymbol{g}} \tag{31} $$

Where ${}^{ENU}\vec{\boldsymbol{\alpha}}$ is the corrected acceleration vector for $\vec{\boldsymbol{\alpha}} : \mathbb{R}^3 \to \mathrm{Im}\left\{\mathbb{H}\right\}$. As we have initialised the orientation of the IMU sensor frame to be equal to the ENU global reference frame, the angle about the $\hat{\boldsymbol{z}}$ axis would be identical in both frames. Therefore for any arbitrary orientation, the yaw angle, denoted by ψ, can be obtained from ${}^{IMU}_{ENU}\hat{\boldsymbol{q}}$ as follows.

$$ \psi = \operatorname{atan2}\left(2(q_w q_z + q_x q_y), -1 + 2(q_w^2 + q_x^2)\right) \tag{32} $$

This corresponds to $\psi : \mathbb{H} \to \mathbb{R},\ \forall \hat{\boldsymbol{q}} \in \mathbb{H}$. As the atan2 function is defined in radians, we define its interval as $(-\pi, \pi]$ If we assume that the heading angle is equal to the yaw, this implies that the user heading will be aligned with the forward orientation of the IMU. Therefore, we assume the IMU is pointing in the direction of forward travel for accurate heading estimation.

We have now obtained a heading estimation and a way to obtain gravity-free acceleration in any 3 axis regardless of the orientation of the IMU through quaternions. In order to obtain these quaternions, we must fuse the accelerometer and gyroscope data. Common fusion algorithms for IMUs and MARGs are the Madgwick [57] and Mahony [58] filters based on gradient descent and proportional-integral control respectively. They are specific sensor fusion algorithms tailored to IMU/MARG orientation estimation whilst being computationally more efficient than a standard Kalman filter.

For the BNO085, the fusion procedure can be performed through the DMP. The Adafruit BNO08X library provides examples for 6-DOF fusion for quaternion data by using the `GameRotationVector()` function. The exact fusion algorithm used in the DMP is proprietary but Ceva Inc. reports a drift of 0.5°/min [36]. We can describe this process in Arduino IDE as shown in Algorithm 4. We obtained ψ using quaternion data from the DMP and converted this from radians to degrees. From there we can correct for the gravity component through the Hamilton product for quaternion rotations.

Algorithm 4: Heading Calculation & Gravity Correction

```
Input  : None
Output: Corrected acceleration vector: ${}^{ENU}\vec{\boldsymbol{\alpha}}$ where $\vec{\boldsymbol{\alpha}} \in \mathrm{Im}\{\mathbb{H}\}$
        Heading angle from IMU quaternion: $\psi$
Initialise:
    $g = 9.80665$;                                   // Gravitational acceleration constant
    ${}^{ENU}\hat{\boldsymbol{q}} \leftarrow {}^{IMU}\hat{\boldsymbol{q}}$;        // Define $ENU$ frame using initial orientation
    ${}^{ENU}\vec{\boldsymbol{g}} = \langle 0, 0, 0, g \rangle$;                         // Define gravity vector
Function readSensor():
    if SENSOR ID == ACCELEROMETER ID then
        ${}^{IMU}\vec{\boldsymbol{a}} \leftarrow$ 3-axis acceleration from IMU;       // Obtain acceleration from IMU
        ${}^{ENU}\vec{\boldsymbol{a}} \leftarrow {}^{IMU}_{ENU}\hat{\boldsymbol{q}} \otimes {}^{IMU}\vec{\boldsymbol{a}} \otimes {}^{IMU}_{ENU}\hat{\boldsymbol{q}}^{*}$;                   // $ENU$ acceleration
        return ${}^{ENU}\vec{\boldsymbol{\alpha}} \leftarrow {}^{ENU}\vec{\boldsymbol{a}} - {}^{ENU}\vec{\boldsymbol{g}}$;      // Gravity corrected acceleration
    end if
    if SENSOR ID == GAME ROTATION VECTOR ID then
        ${}^{IMU}_{ENU}\hat{\boldsymbol{q}} \leftarrow$ Current quaternion orientation relative to $ENU$;
        $\psi \leftarrow$ computeHeading$\left({}^{IMU}_{ENU}\hat{\boldsymbol{q}}\right)$;
    end if
end
Function computeHeading($\hat{\boldsymbol{q}}$):
    $q_w \leftarrow \mathbb{R}$ part of $\hat{\boldsymbol{q}}$;
    $q_x \leftarrow \boldsymbol{i}$ part of $\hat{\boldsymbol{q}}$;
    $q_y \leftarrow \boldsymbol{j}$ part of $\hat{\boldsymbol{q}}$;
    $q_z \leftarrow \boldsymbol{k}$ part of $\hat{\boldsymbol{q}}$;
    $\psi = \mathrm{atan2}\left(2(q_z q_w + q_x q_y), -1 + 2(q_w^2 + q_x^2)\right)$;          // $\psi$ in radians
    $\psi \leftarrow \psi \times \frac{180}{\pi}$;                                // $\psi$ in degrees
    if $\psi < 0$ then
        $\psi \leftarrow \psi + 360$;           // Adjust $\psi$ to be within the range [0, 360)
    end if
    return $\psi$;                              // Yaw angle for user heading
end
```

4.3 Butterworth Low-pass Accelerometer Filter

Obtaining a heading estimation allows for the description of directional travel but we need to determine the distance travelled, namely the step length, using the accelerometer data. A common technique involves analysing the vertical acceleration, namely a_z, during motion [59] which can now be obtained through our quaternion fusion. However, acceleration data suffers from high frequency noise therefore any subsequent analysis must be preceded by an appropriate filtering method.

For an embedded system, digital filters like a Finite Impulse Response (FIR) or an Infinite Impulse Response (IIR) can be used to process a signal by passing or attenuating the frequency components of that signal. Both digital filters utilise difference equations for discrete time applications. For the FIR, the difference equation is as follows.

$$y[n] = \sum_{k=0}^{N} b_k \cdot x[n-k] \tag{33}$$

Where $y[n]$ is the output at discrete-time n, N is the filter order, b_k are the filter coefficients, and $x[n-k]$ is the input at discrete-time $n-k$. For the IIR, the difference equations is as follows.

$$y[n] = \sum_{k=0}^{M} b_k \cdot x[n-k] - \sum_{k=1}^{N} a_k \cdot y[n-k] \tag{34}$$

Where b_k are filter coefficients for the input values and a_k are filter coefficients for the output values. FIRs generally have greater stability and a linear phase relative to IIRs [60]. IIRs however require fewer coefficients and are therefore less memory intensive. For this reason, we have chosen an IIR digital filter implementation.

For our application, we require a low-pass filter therefore we need to specify the cutoff frequency, f_c in Hz, and the filter order. For the cutoff frequency, we will use the value specified in the paper by Sung et al. [28] where they studied various walking speeds of 50 participants across the ages of 25 to 35 and selected $f_c = 3$ Hz.

For the filter order, this is largely dependent on the design of the filter. Common analog filters like the Butterworth, Chebyshev Type I & Type II, and Elliptical can be implemented digitally through an IIR filter. To select the most appropriate filter design, we can analyse their frequency response using Figure 15.

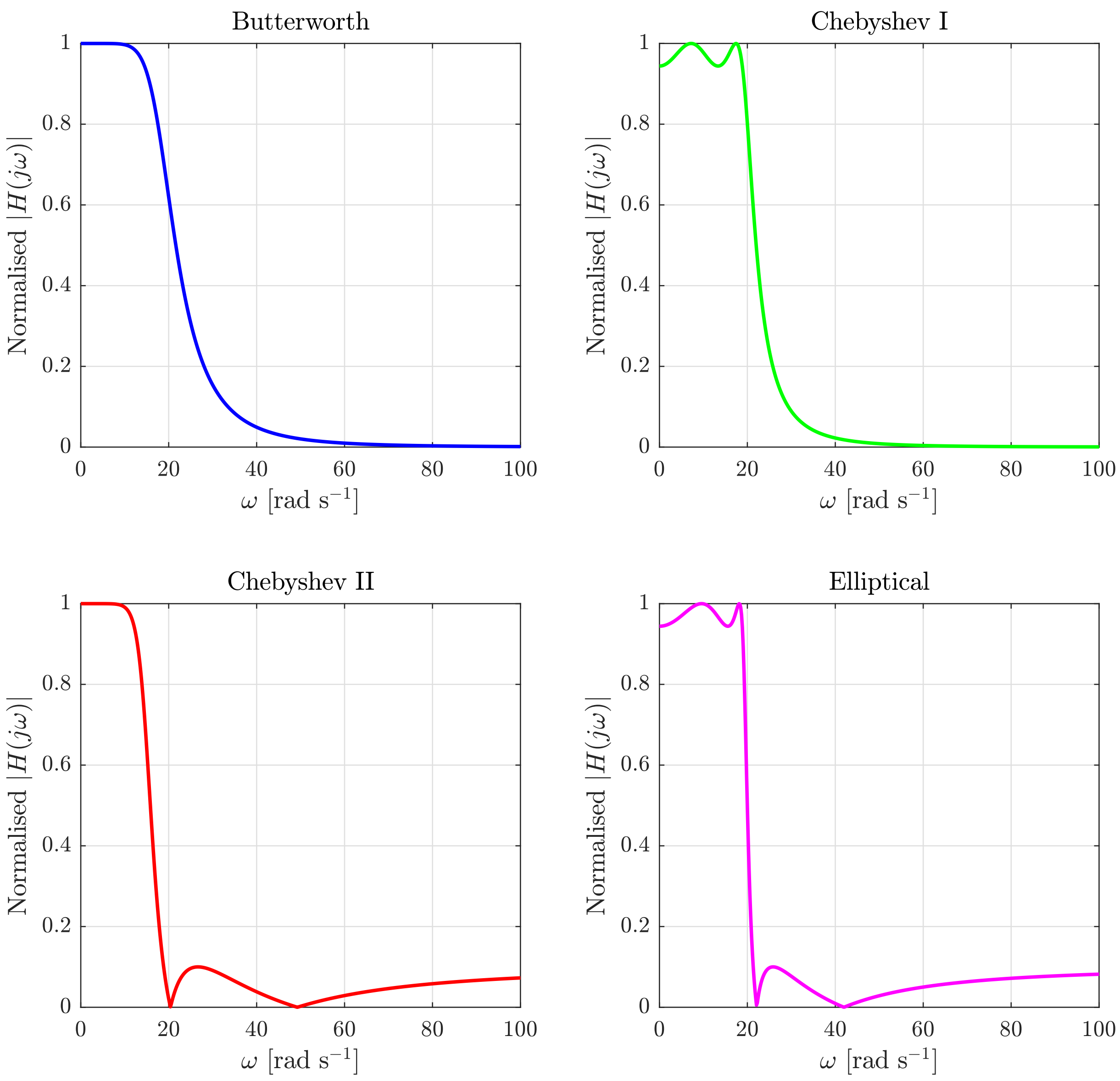


Figure 15: Normalised frequency response bode plots of 4th order low-pass filters

Figure 15 shows the normalised frequency response for 4 common filter designs using a low-pass filter with order of 4 and cutoff frequency of 3 Hz. For the Butterworth filter, it exhibits a maximally flat response in the passband with no ripples in the stopband, whilst having the widest transition band. For Chebyshev I, ripples are present in the passband with a flat stopband response. For Chebyshev II, we observe a flat response in the passband but ripples are observed in the stopband. For the Elliptical filter, ripples are present in the both the passband and stopband but it has the fastest roll off rate.

As we intend to filter acceleration data, the Butterworth filter would be an ideal choice due to its maximally flat passband response. This ensures that no important acceleration characteristics are lost during motion and high frequency noise would not distort the overall signal. When selecting the Butterworth order, we required adequate attenuation at higher frequencies through a higher filter order but this causes phase shift induced time delays and numerical instability. Therefore, a Butterworth filter order of 4 was chosen where a filter derivation & low-pass class header file have been added to the Appendix. We can implement this filter in Arduino IDE to verify this design choice empirically.

For our experiment, we have taken 5000 paired measurements of raw and filtered ${}^{ENU}\alpha_z$ values. By measuring the time taken between each paired measurement using the Python `time.time()` method, we obtain a sample period of $T_s = 0.0078$ s resulting in a sampling frequency of $f_s \approx 128$ Hz which matches the accelerometer's sensor rate. Given $f_c = 3$ Hz, the f_s satisfies the Nyquist–Shannon sampling theorem which requires the sampling frequency to be twice the highest frequency component present in the signal to prevent aliasing. Our results can be visualised through Figure 16.

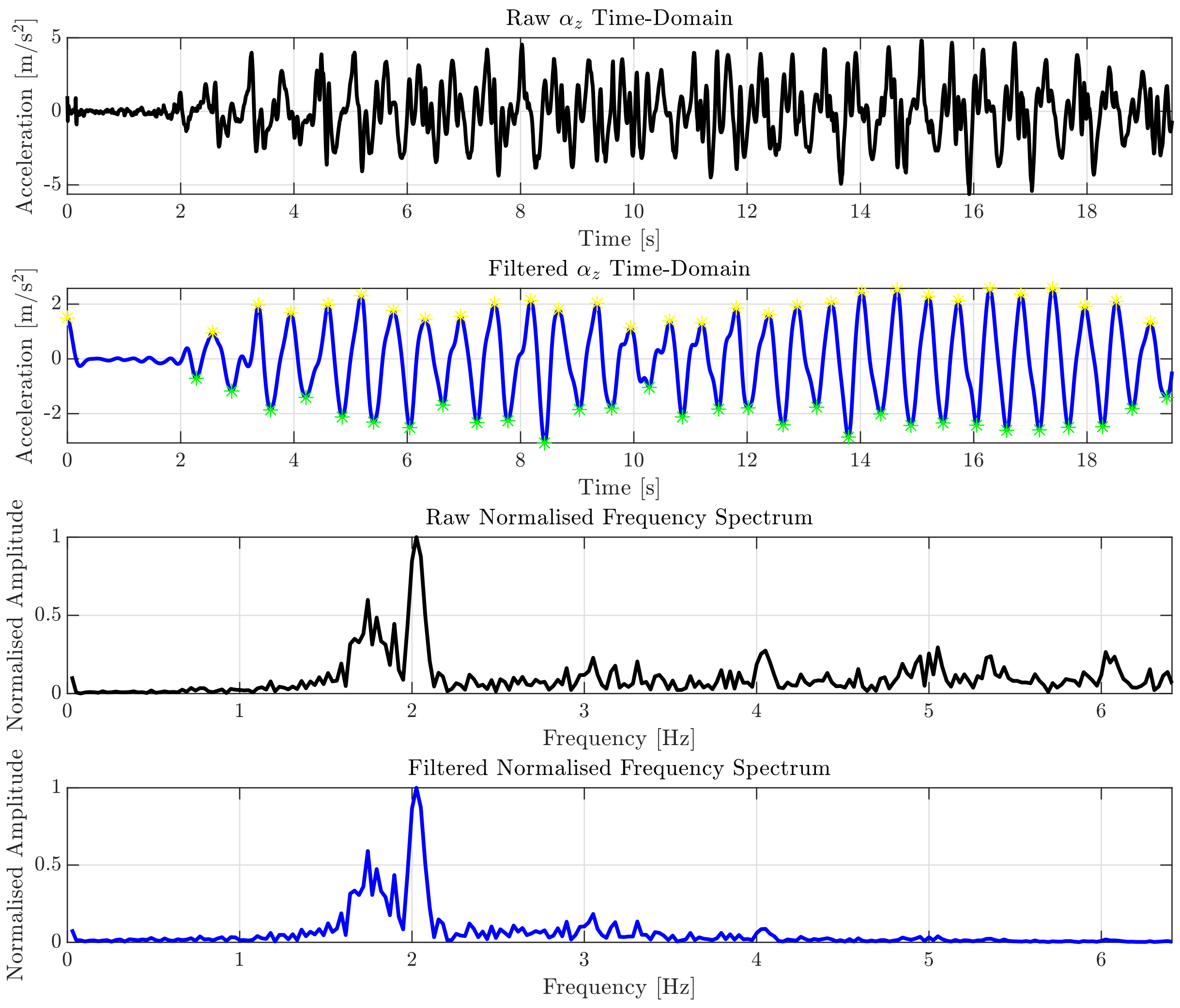


Figure 16: Raw and filtered α_z plots in time & frequency domain

Figure 16 shows the time and frequency domain plots for the raw and filtered data. The Arduino was held parallel to $x - y$ plane & pointed towards forward user travel. To aid our analysis, we have also computed the Fast Fourier Transform to analyse the single-sided frequency spectra for our raw and filtered data. This frequency spectra was normalised by dividing the signal by the maximum amplitude. We observe that the quaternion fusion acts as a high pass filter as the DC gravity component is absent in the frequency spectra. The peak frequency is observed around 2 Hz and frequencies are filtered past 3 Hz. By using a low filter order, we avoided numerical instability. Analysing the time-domain waveforms, it is clear to see that our filtering has successfully removed the high frequency noise. The peaks highlighted yellow and troughs highlighted green were obtained using a non-derivative based peak detection [61]. This was necessary as observing the raw time-domain signal we can see "false peaks" wherein high frequency noise typical zero-derivative extrema detection difficult.

4.4 Step Detection & Estimation

Instead of derivative methods, we can employ a peak detection that leverages thresholding. If we observe the true peaks, they are surrounded by adjacent minima and true troughs are surrounded by adjacent maxima, where these minima and maxima are locally defined. Let us define an acceleration and time threshold respectively: Δa for $a \in \mathbb{R}$, and $\Delta\tau$ for $\tau \in \mathbb{R}$. For the decreasing part of the filtered signal consider: $\alpha_z(t)$ then at $\alpha_z(t + \delta t)$ we expect the function to decrease. This continues until the true minimum where the next point would have a larger value, where: $\alpha_z(t_{min} + \delta t) > \alpha_z(t_{min})$. This search procedure is repeated for the maxima to obtain all peaks and troughs.

We extend this to the raw data by conducting the search but enforcing the criterion that the acceleration difference and time spacing between each maximum and minimum must exceed the given thresholds Δa & $\Delta\tau$ respectively. When the conditions are satisfied, valid peaks and troughs are identified. This procedure is neccessary for our embedded device as we avoid cases of false peak detection.

To understand the importance of peak detection in step length estimation, we can turn to the paper by Harvey Weinberg [62]. As step length varies throughout motion, a dynamic step length estimator is needed. By observing hip movement during motion, an efficient & dynamic equation was proposed by Weinberg as follows.

$$\lambda = \kappa \cdot \sqrt[4]{\alpha_{z_{max}} - \alpha_{z_{min}}} \tag{35}$$

Where $\lambda \in \mathbb{R}$ is the step length in m and $\kappa \in \mathbb{R}^+$ is the Weinberg calibration coefficient. Therefore, λ is proportional to the difference in peak and trough vertical accelerations. Algorithm 5 outlines the Arduino IDE implementation for the step detection & estimation inspired from an 8-bit MCU implementation [63]. We selected $\Delta a = 1$ and $\Delta\tau = 300$ ms and $(+)$ & $(-)$ superscripts are current and previous values respectively.

Algorithm 5: Step Detection & Estimation

```
Input  : Current acceleration value: $\alpha_z(t)$
Output: Step length: $\lambda$
        Step detection flag: STEP_DETECT_FLAG
Initialise:
    $\alpha_z(t-1) \leftarrow \alpha_z(t)$;          // Initially, no previous acceleration
    $\alpha^-_{z_{max}} \leftarrow -\infty$;                // Start with a very low max
    $\alpha^-_{z_{min}} \leftarrow \infty$;                 // Start with a very high min
    $t^-_{max} \leftarrow 0$;                          // Time at last peak
    $t^-_{min} \leftarrow 0$;                          // Time at last trough
    STEP_DETECT_FLAG $\leftarrow$ $False$;       // Start with no steps detected
Function stepDetect($\alpha_z(t)$):
    $t \leftarrow$ millis();                       // Current time
    // Detecting peaks
    if $\alpha_z(t) > \alpha^-_{z_{max}}$ then
        $\alpha^-_{z_{max}} \leftarrow \alpha_z(t)$;
    end if
    else if $(\alpha_z(t) < \alpha_z(t-1)) \wedge ((t - t^-_{max}) > \Delta\tau)$ then
        if $(\alpha^-_{z_{max}} - \alpha_z(t)) > \Delta a$ then
            $t^-_{max} \leftarrow t$;
            $\lambda \leftarrow$ stepEstimate$(\alpha^-_{z_{max}}, \alpha^-_{z_{min}})$;     // Compute the step length
            $step\_detect\_flag \leftarrow True$;      // Flag asserted for valid step
            $\alpha^-_{z_{min}} \leftarrow \alpha_z(t)$;
            $\alpha^-_{z_{max}} \leftarrow -\infty$;
        end if
    end if
    // Detecting troughs
    if $\alpha_z(t) < \alpha^-_{z_{min}}$ then
        $\alpha^-_{z_{min}} \leftarrow \alpha_z(t)$;
    end if
    else if $(\alpha_z(t) > \alpha_z(t-1)) \wedge ((t - t^-_{min}) > \tau_{min})$ then
        if $(\alpha_z(t) - \alpha^-_{z_{min}}) > \delta_a$ then
            $t^-_{min} \leftarrow t$;
            $\alpha^-_{z_{max}} \leftarrow \alpha_z(t)$;
            $\alpha^-_{z_{min}} \leftarrow \infty$;
        end if
    end if
    $\alpha_z(t-1) \leftarrow \alpha_z(t)$;              // Update for next iteration
end
Function stepEstimate($\alpha_{z_{max}}, \alpha_{z_{min}}$):
    // Weinberg formula for step length
    return $\lambda \leftarrow \kappa \cdot \sqrt[4]{\alpha_{z_{max}} - \alpha_{z_{min}}}$;
end
```

We can now calibrate the Weinberg coefficient κ as follows. We took our measurements in TP-B08. We defined slow, normal, and fast walking speeds for 0.6 m, 0.7 m, and 0.8 m step lengths. We used the same 110 BPM cadence employed during trilateration. Each tested step length experiment enforced a minimum of 10 m total distance travelled to ensure proper κ calibration. Therefore, for slow to fast speeds, total distance travelled was 10.2 m, 10.5 m, and 10.4 m respectively. We took 2000 measurements and repeated 10 tests for each speed. As the κ value is different for each step, we compute all the κ values then we used a curve fitting method for the given test. The optimal κ_{opt} value for an experiment is as follows.

$$\kappa_{\text{opt}} = \arg\min_{\kappa} \sum_{i=1}^{n} \left| \lambda_i - \kappa \cdot \sqrt[4]{\alpha_{z_{max}} - \alpha_{z_{min}}} \right| \tag{36}$$

Our κ results are shown as boxplots in Figure 17.

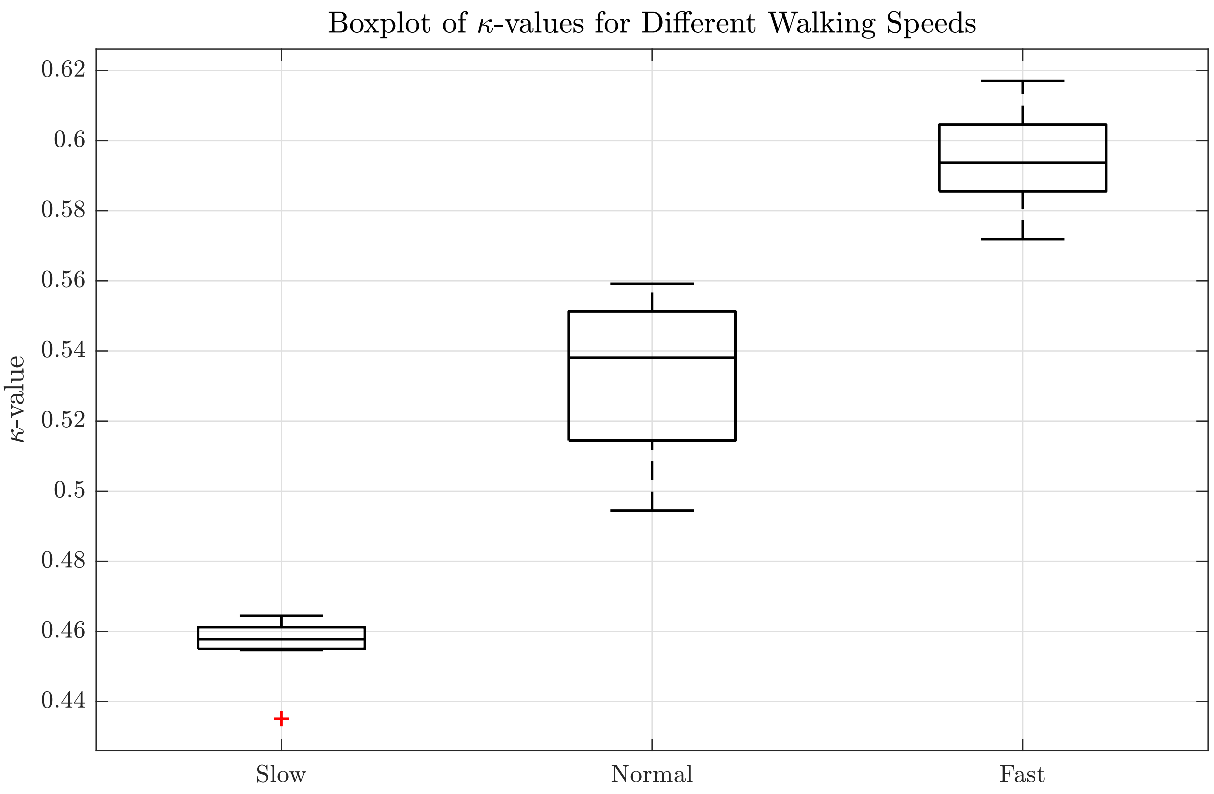


Figure 17: Weinberg coefficient calibration boxplot results

From the boxplot, the κ value would be different depending on the walking speed. We note an outlier κ value in the slow dataset. As the step detection requires a constant value, we can use the mean value $\bar{\kappa}$. After removing the outlier, we computed $\bar{\kappa} = 0.5309$.

4.5 Pedestrian Dead Reckoning

As we can determine step length and heading, we can now perform the dead reckoning. We specify a starting point: $\mathbf{x}_0 = (x_0, y_0)$ and initialise our sensor in the ENU frame as described previously. We implemented a `MobileUser` class to encapsulate the logic for the user's motion, which we have included in the appendix. We took 200 paired measurements in FC-112 taking a path described by a 8.40×6.15 m^2 rectangular grid. Tape markings with 0.7 m spacing and a 110 BPM provided ground truth estimates. Figure 18 is a plot of our results.

FC-112 was a larger testing site compared to FC-132 therefore less turns and more straight paths were taken. We can differentiate between gyroscopic drift and over/under-estimation of step length through their trajectories. If the results appear to rotate over time then it can be attributed to drift. If the results show a general offset, it can be due to over/underestimation for the step length.

For test 1, the dynamics of the user are well captured with minimal systematic error. Test 1 retains the shape of the trajectory but the results are offset. This could be due to the user's irregular walking pattern or the damping of the floor causing a reduction in acceleration difference. For test 2 we notice clear gyroscopic drift over time as the trajectory rotates over time. A suspected cause in faster drift could be due to faster turns taken by the user. If the user increases their angular velocity, where gyroscope noise could be amplified by these turns.

Interestingly, test 3 displays less drift but a systematic error in the initial heading. Our system assumes that the device is properly initialised in the ENU frame but this could be inaccurate at times. Incorrect initial orientations would lead to a bias in the heading angle as evident in the results. Another reason could be that the BNO085 was not reset properly. If the IMU was not reset, the drift could still be accumulated from the prior test phase. Whilst the device was powered on & off before testing, this does not necessarily guarantee a sensor reset.

Possible variations in step length could be the user exerting more force on their steps during motion. Irregular floor surfaces could distort the acceleration measurements. The device was kept planar and in the direction of forward travel but the user could have turned before the device turned leading to an incorrect step propagation direction.

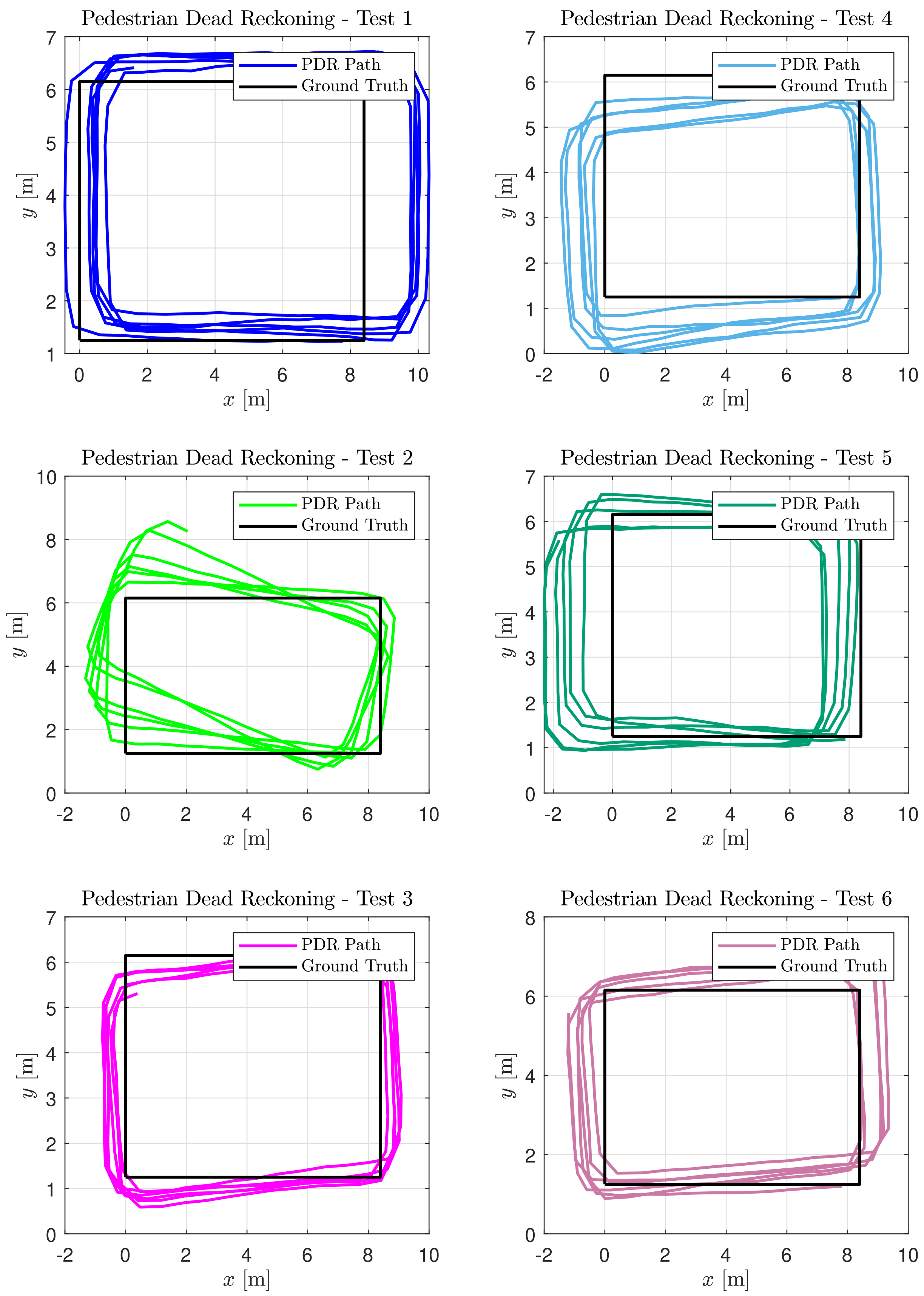


Figure 18: Pedestrian dead reckoning results taken in FC-112

The PDR model can capture the user's dynamics but suffers from accumulated drift and step length errors. These effects are exacerbated when the user deviates from their typical walking patterns particularly when turning faster and varying the velocity. Whilst we have computed the PDR statistics, these results are *not* indicative of "good localisation". The errors are unbounded and simple metrics like averages would diverge. The statistics in Table 9 provide a numerical interpretation of the results. Notably, the $F(\varepsilon)$ plot in Figure 19 would flatten over time as the errors accumulate.

Table 9: Statistical measures of the errors in Pedestrian Dead Reckoning

Test	Central Tendency [m]					Percentiles [m]				
	μ_ε	σ_ε	ε_{min}	ε_{max}	$\varepsilon_{\text{range}}$	25th	50th	75th	90th	IQR
1	5.6951	2.3204	0.7058	10.4646	9.7588	4.0744	5.6152	7.4625	8.8306	3.3881
2	5.3758	2.2626	0.0234	10.2320	10.2085	3.7247	5.4435	6.9089	8.6572	3.1842
3	4.8101	2.1881	0.4910	9.0523	8.5614	3.3407	4.7507	6.4335	8.0256	3.0928
4	4.9095	2.3197	0.3616	9.8863	9.5247	3.1223	5.0570	6.4693	8.0479	3.3469
5	4.5395	2.5020	0.1874	10.8189	10.6315	2.4453	4.6154	5.8766	8.1223	3.4312
6	4.8704	2.1755	0.1205	9.9107	9.7902	3.0023	4.8616	6.3841	7.8817	3.3817

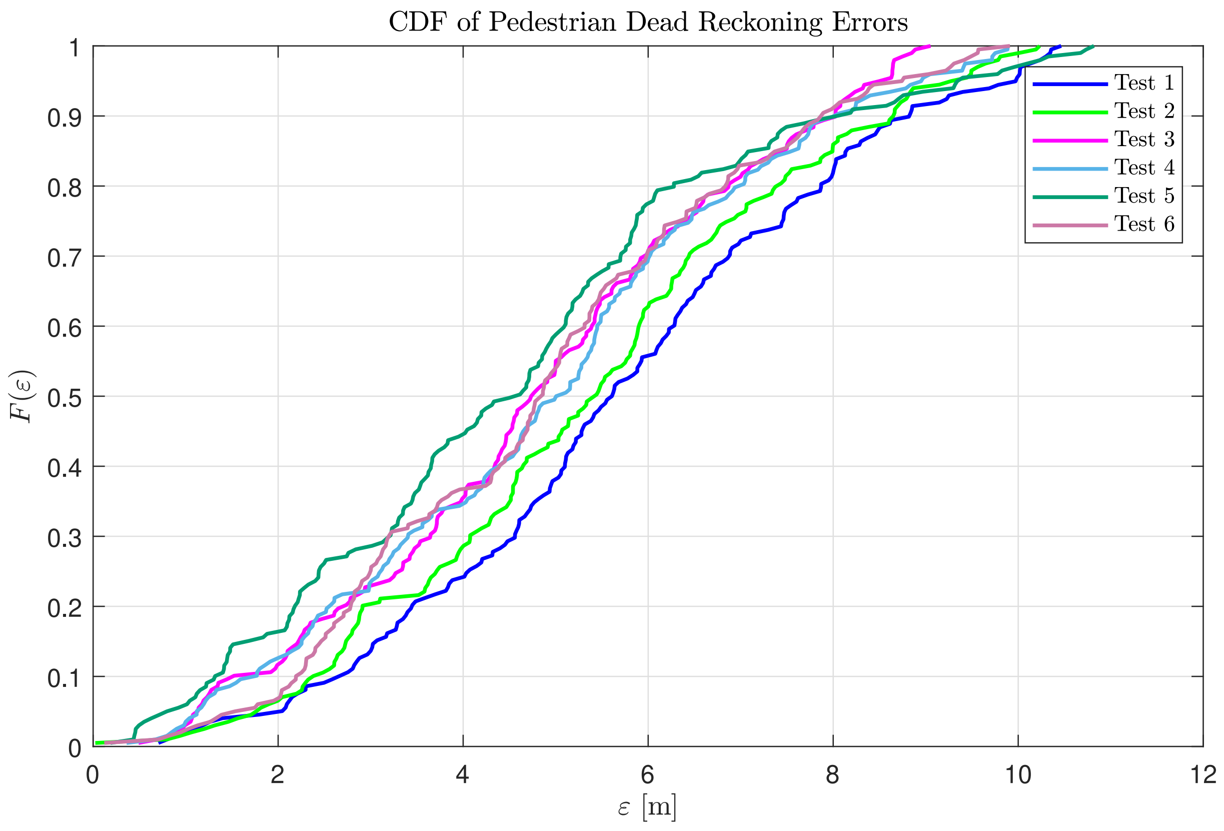


Figure 19: CDF of Pedestrian Dead Reckoning errors

Chapter 5. Sensor Fusion

5.1 Multithreading

When combining the PDR and BLET systems, we must consider concurrent or parallel computation when fusing. If we consider the scanning process for BLE, it is an asynchronous that leverages function callbacks, acting as interrupts. However, each callback deals with a singular advertising device at a time and is therefore inappropriate for localisation. Moreover, when conducting a scan, the device is idle hence we lose track of the user and their subsequent steps. We can instead leverage the Real-Time Operating System (RTOS) environment for multithreading.

As the Arduino contains a Dual-core Xtensa processor, we can conduct the scanning task and the step propagation in parallel with one another. We have achieved this in Arduino IDE through the `FreeRTOS` library. To prevent concurrency issues, a `mutex` object is a type of semaphore that controls thread access to certain sections of the code. In doing so, we prevent race conditions between threads trying to access shared resources. The `xSemaphoreTake()` and `xSemaphoreGive()` functions allows for the exchange of `mutex` control between threads.

5.2 Kalman Filter

The Kalman filter is a special case of Bayesian filtering in which we assume a linear state transition & observation model with Gaussian noise distributions. Under ideal conditions, it is the best linear unbiased estimator as it minimises the mean-square error. We can describe the prediction-update steps as follows.

$$\textbf{Initialisation}\begin{cases} \hat{\mathbf{x}}_{0|0} = \text{Initial state estimate} \\ \mathbf{P}_{0|0} = \text{Initial error covariance estimate} \end{cases} \tag{37}$$

$$\textbf{Prediction}\begin{cases} \hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k & \text{State Prediction} \\ \mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^T + \mathbf{Q}_k & \text{Covariance Prediction} \end{cases} \tag{38}$$

$$\textbf{Update}\begin{cases} \mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} & \text{Kalman Gain} \\ \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}\right) & \text{State Update} \\ \mathbf{P}_{k|k} = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_{k|k-1} & \text{Covariance Update} \end{cases} \tag{39}$$

For the prediction with n number of states & m number of controls: $\hat{\mathbf{x}}_{k|k-1} \in \mathbb{R}^n$ is the predicted state estimate, $\mathbf{P}_{k|k-1} \in \mathbb{R}^{n\times n}$ is the predicted error covariance, $\mathbf{F}_k \in \mathbb{R}^{n\times n}$ is the state transition matrix applied to the prior state estimate $\hat{\mathbf{x}}_{k-1|k-1} \in \mathbb{R}^n$, $\mathbf{B}_k \in \mathbb{R}^{n \times m}$ is the control input matrix for the control input vector $\mathbf{u}_k \in \mathbb{R}^m$, and $\mathbf{Q}_k \in \mathbb{R}^{n \times n}$ is the process noise covariance matrix.

For the update with l number of measurements: $\mathbf{K}_k \in \mathbb{R}^{n \times l}$ is the Kalman gain, $\hat{\mathbf{x}}_{k|k} \in \mathbb{R}^n$ is the updated state estimate after including the measurement $\mathbf{z}_k \in \mathbb{R}^l$, $\mathbf{P}_{k|k} \in \mathbb{R}^{n \times n}$ is the updated estimate of the error covariance after measurement, $\mathbf{H}_k \in \mathbb{R}^{l \times n}$ is the observation matrix that maps the state space to the measurement space, $\mathbf{R}_k \in \mathbb{R}^{l \times l}$ is the measurement noise covariance matrix, and $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix.

A common form for the Kalman filter is adapted into our system [28][31][64].

$$\begin{aligned} \hat{x}_k &= \hat{x}_{k-1} + \lambda_k \cos(\psi_k) + Q_x \\ \hat{y}_k &= \hat{y}_{k-1} + \lambda_k \sin(\psi_k) + Q_y \end{aligned} \tag{40}$$

Where we can express this in matrix form as follows.

$$\begin{bmatrix} \hat{x}_k \\ \hat{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{y}_{k-1} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_k \cos(\psi_k) \\ \lambda_k \sin(\psi_k) \end{bmatrix} + \begin{bmatrix} Q_x \\ Q_y \end{bmatrix} \tag{41}$$

Which simplifies to below.

$$\hat{\mathbf{x}}_k = \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{Q}_k \tag{42}$$

The observation model in matrix form is the result of the BLET solution.

$$\begin{bmatrix} \hat{x}_k \\ \hat{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{y}_{k-1} \end{bmatrix} + \begin{bmatrix} R_x \\ R_y \end{bmatrix} \tag{43}$$

Which simplifies to below.

$$\mathbf{z}_k = \mathbf{H}_k \hat{\mathbf{x}}_k + \mathbf{R}_k \tag{44}$$

Initialisation of the KF is important and since we are not necessarily concerned with filter convergence, we can initialise the filter at the true $\mathbf{x}_0$ with initial noise, $\nu_0 = 0.01$.

$$\begin{aligned} \hat{\mathbf{x}}_0 &= \mathbf{x}_0 \\ \mathbf{P}_0 &= \nu_0 \cdot \mathbf{I} \end{aligned} \tag{45}$$

We can therefore describe the pseudocode logic for the KF using Algorithm 6.

Algorithm 6: Kalman Filter

```
Input   : None
Output: Estimated user coordinates: $\hat{\mathbf{x}}_k$ where $\hat{\mathbf{x}}_k \in \mathbb{R}^n$
Initialise:
    $\hat{\mathbf{x}}_0 = \mathbf{x}_0$;                                   // Initial state estimate
    $\mathbf{P}_0 = \nu_0 \cdot \mathbf{I}$;                               // Initialise with some noise
Function predictKF():
    if step is detected then
        $\hat{\mathbf{x}}_{k|k-1} \leftarrow \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k$;          // Propagate state
        $\mathbf{P}_{k|k-1} \leftarrow \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^T + \mathbf{Q}_k$;      // Predicted error covariance
    end if
end
Function updateKF():
    if trilateration is valid then
        $\mathbf{K}_k \leftarrow \mathbf{P}_{k|k-1}\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1}$;          // Kalman Gain
        $\hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}\right)$;        // Update state
        $\mathbf{P}_{k|k} \leftarrow (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\mathbf{P}_{k|k-1}$;        // Update error covariance
    end if
    return $\hat{\mathbf{x}}_k$;                              // Estimated user position
end
```

Our measurements were taken in FC-132 with similar experimental methods as before. 200 paired measurements were obtained where shorted intervals were saved using the KF prediction and when ready, the updated states were saved. Figure 20 shows our results. We performed the tuning with the assumption that the dynamically model was accurate with noisy RSSI measurements. Firstly, when we decreased the measurement noise, the KF was extremely sensitive to noisy measurements and it tried to correct the estimates which lead to a diverging state. The best performance appears to be when the process noise is set very low with a higher measurement noise. As the assumptions are Gaussian noise, increasing the measurement noise would cause the measurement noise to be almost uniformly distributed due to wider tails. By having a very small process noise, the filter favours the system's dynamics and the wider measurement distribution allows the state distribution to be contained within the measurement distribution without much correction. However, you have collapsed the filter to a singular point and therefore any dynamical changes to the measurements would make the filter much less responsive.

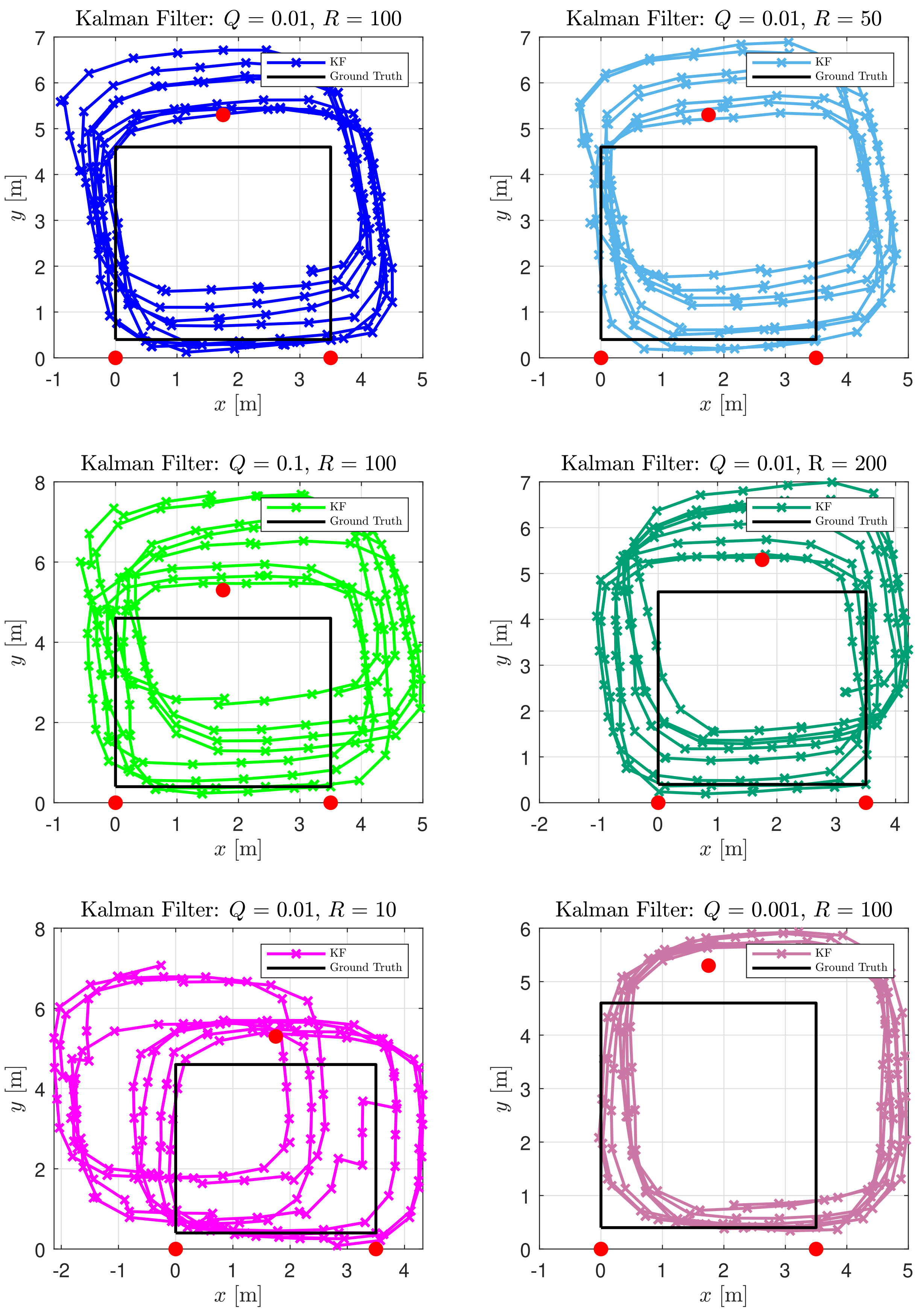


Figure 20: Kalman filter results taken in FC-132

The benefit to having high measurement noise allows the filter to be robust to noise outliers and selecting a low process noise is reflective of our sensor model. However the difference in $\mathbf{Q}_k = 0.01$ and $\mathbf{Q}_k = 0.001$ suggests that the former can better correct itself due to the measurements as it trusts the process model less, whilst the later shows strong consistency in trajectory but it is biased outwards due to the sensor measurements. $\mathbf{Q}_k = 0.001$, $\mathbf{R}_k = 100$ is a prime example of the filter collapsing to a singular point in which subsequent measurements cannot correct the offset. We can analyse the filter's statistics from Table 10. We observe that the best performing filter seems to be $\mathbf{Q}_k = 0.01$, $\mathbf{R}_k = 200$. This is also a likely optimal choice given the conditions as we are not overconfident in the process model but we are also robust to noisy measurements. Empirically, $\mathbf{Q}_k = 0.01$, $\mathbf{R}_k = 200$ displayed the lowest μ_ε with the second lowest σ_ε. These tuned parameters have verified theoretical understanding about the system's noise distributions.

Table 10: Statistical measures of the errors in Kalman Filter

Test	Central Tendency [m]					Percentiles [m]				
	μ_ε	σ_ε	ε_{min}	ε_{max}	$\varepsilon_{\text{range}}$	25th	50th	75th	90th	IQR
1	1.5036	0.9744	0	4.5994	4.5994	0.8230	1.1921	2.1268	3.0458	1.3038
2	1.8265	1.2588	0	4.9396	4.9396	0.8965	1.3778	2.8648	3.8166	1.9683
3	2.1347	1.3230	0	5.3635	5.3635	1.1052	1.7300	2.9294	4.3411	1.8242
4	1.5080	0.9782	0	3.7857	3.7857	0.7622	1.2308	2.3775	3.0089	1.6153
5	1.3602	0.9540	0	3.9155	3.9155	0.5747	1.1993	2.0632	2.8157	1.4885
6	1.7796	0.9182	0	3.9332	3.9332	1.1387	1.7049	2.3663	3.1617	1.2276

We can plot the $F(\varepsilon)$ function as follows. We observe that the 90^{th} percentile for the best case was just less than 3 m. A majority of the filters performed well when observing the medians and the overall distribution of errors. Except for $\mathbf{Q}_k = 0.01$, $\mathbf{R}_k = 10$ and $\mathbf{Q}_k = 0.001$, $\mathbf{R}_k = 100$ displaying median values and distributions much futher away from the other curves. However, for $\mathbf{Q}_k = 0.001$, $\mathbf{R}_k = 100$, we saw improved performance at the upper quartile as it was more robust to errors. We conclude that a high measurement covariance $\mathbf{R}_k$ is needed but care needs to be taken when selecting $\mathbf{Q}_k$ so as to prevent the filter from overestimating the confidence of the process model.

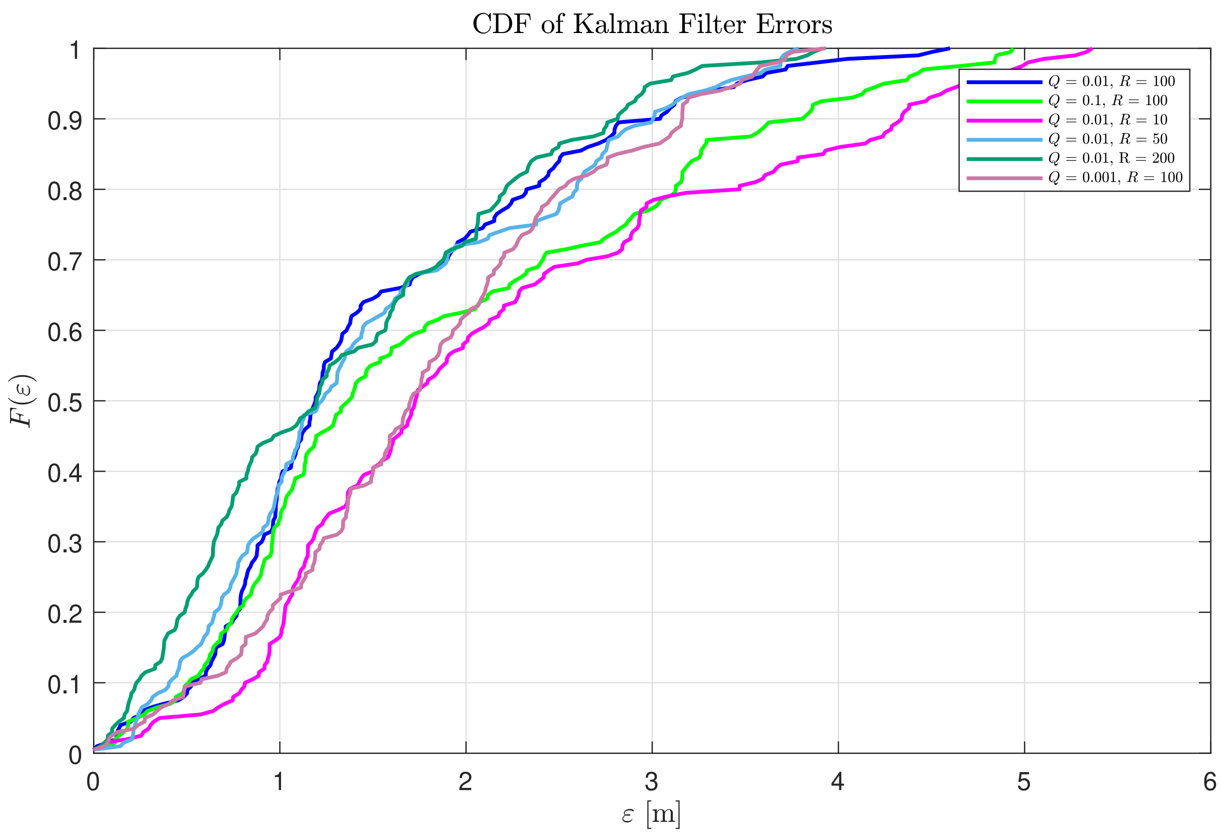


Figure 21: CDF for Kalman Filter errors

5.3 Particle Filter

The PF or Sequential Monte Carlo methods are employed when the system's models are potentially non-linear and/or non-Gaussian. A set of particles are generated randomly in an attempt to approximate a given distribution [65]. The procedure is as follows.

1. **Initialisation:** $X_0^{(i)} \sim p(\mathbf{x}_0), \quad i = 1, \ldots, N$

2. **Prediction:** $X_k^{(i)} \sim p(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)}), \quad i = 1, \ldots, N$

3. **Update Weights:** $w_k^{(i)} = p(\mathbf{z}_k | X_k^{(i)}), \quad i = 1, \ldots, N$

4. **Normalise Weights:** $\tilde{w}_k^{(i)} = \frac{w_k^{(i)}}{\sum_{j=1}^N w_k^{(j)}}, \quad i = 1, \ldots, N$

5. **Resampling:** $\{X_k^{(i)}\}$ **from** $\{X_k^{(i)}, \tilde{w}_k^{(i)}\}$

6. **Estimate:** $\hat{\mathbf{x}}_k = \sum_{i=1}^N \tilde{w}_k^{(i)} X_k^{(i)}$

We can describe the pseudocode as follows.

Algorithm 7: Particle Filter

```
Input  : None
Output: Estimated state: $\hat{\mathbf{x}}_k$
Initialise:
for $i \leftarrow 1$ to $N$ do
    $X_0^{(i)} \sim p(\mathbf{x}_0)$;
end for
Function predictPF():
    if step is detected then
        for $i \leftarrow 1$ to $N$ do
            $X_k^{(i)} \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})$;
        end for
    end if
end
Function updatePF():
    if trilateration is valid then
        for $i \leftarrow 1$ to $N$ do
            $w_k^{(i)} \leftarrow p(\mathbf{z}_k|X_k^{(i)})$;                          // Update
        end for
        for $i \leftarrow 1$ to $N$ do
            $\tilde{w}_k^{(i)} \leftarrow \frac{w_k^{(i)}}{\sum_{j=1}^{N} w_k^{(j)}}$;                 // Normalise
        end for
        $\{X_k^{(i)}\}$ from $\{X_k^{(i)}, \tilde{w}_k^{(i)}\}$;                    // Resample
        $\hat{\mathbf{x}}_k \leftarrow \sum_{i=1}^{N} \tilde{w}_k^{(i)} X_k^{(i)}$;                  // Estimate position
        return $\hat{\mathbf{x}}_k$;                                  // Estimated state
    end if
end
```

We can use systematic resampling as it allows for particle diversity whilst remaining computationally efficient [66]. The resampling method is needed to prevent sample impoverishment where we lack distinct particles to represent the target distribution. Our measurement likelihood function was assumed to be Gaussian with $\sigma = 20$ as we want to be robust to errors whilst allowing for measurement corrections.

$$\Phi(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{46}$$

We have performed the same experimental methods as the KF and we can plot our results as follows.

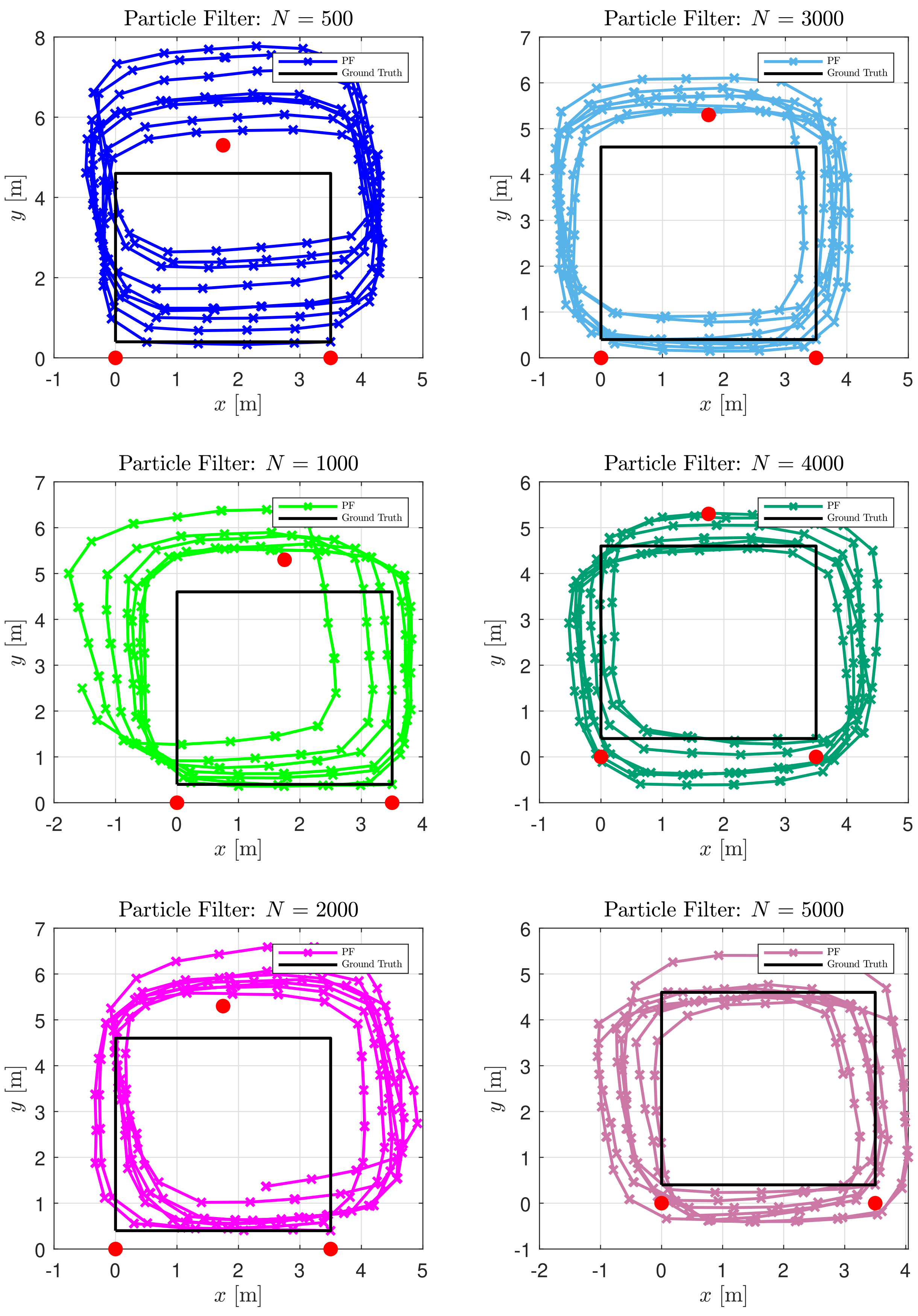


Figure 22: Particle filter results taken in FC-132

The particle filter results show a general trend in improvement in localisation accuracy when increasing the number of particles. As we uniformly distributed the particles radially about the initial point, the particles very far away from the measurements would be eliminated which would lead to sample impoverishment as seen in $N = 500$. The filter would diverge from the true state and the lack of diverse particles would cause the filter to estimate at this diverged state. Therefore, increasing the number of particles would allow for a higher number of particles to remain after updating and resampling. We note that the systematic error in measurement outliers still affects the measurements as seen with $N = 1000$ and $N = 2000$. Perhaps increasing the standard deviation of $\Phi(x)$ would result in a more robust filter but this could also lead to an inability to correct the process model. We can further analyse our system statistically from Table 11.

Table 11: Statistical measures of the errors in Particle Filter

Test	Central Tendency [m]					Percentiles [m]				
	μ_ε	σ_ε	ε_{min}	ε_{max}	$\varepsilon_{\text{range}}$	25th	50th	75th	90th	IQR
1	2.0852	1.3482	0	5.5319	5.5319	0.8084	1.8336	2.9428	4.0249	2.1344
2	1.2234	0.8287	0	3.6707	3.6707	0.4894	1.0434	1.6616	2.4775	1.1723
3	1.7580	0.9101	0	4.1538	4.1538	1.0842	1.6318	2.3771	2.9809	1.2928
4	1.2605	0.7670	0	3.1908	3.1908	0.5929	1.1173	1.8044	2.4437	1.2116
5	1.2432	0.6141	0	2.7911	2.7911	0.8500	1.1930	1.6887	2.0714	0.8387
6	0.9952	0.5575	0	2.8185	2.8185	0.5780	0.8602	1.3311	1.8002	0.7532

We can see that we achieved sub-metre accuracy for $N = 5000$. This was largely due to the large sample size and wider measurement model tails allowing for the correct propagation of the state estimate. The 90^{th} percentile was also below 2 m with a low IQR indicating low variation in the error statistics. By having a large amount of particles, the non-linear process model can also be approximated better as we avoid cases of step length error and early gyro drift which was observed with the KF as it was overconfident in the process model. The low standard deviation also suggests that the system is more reliable than the KF even for $N = 1000$ as we achieved sub 1 m standard deviations. We can now analyse the $F(\varepsilon)$ distributions.

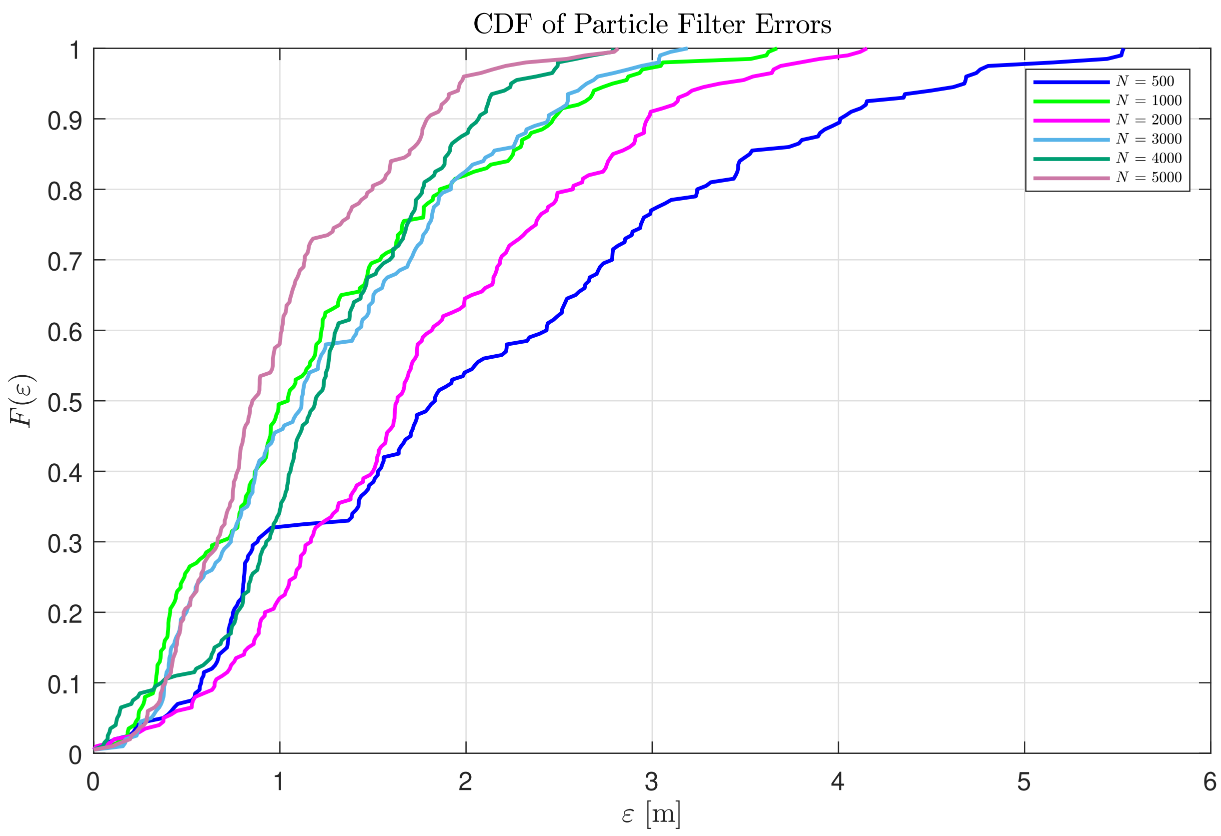


Figure 23: CDF for Particle Filter errors

We can conclude the sensor fusion section by analysing the PF and KF. Typical KF deployment wherein the non-linear terms are placed in $\mathbf{u}_k$ are done to avoid the case of non observable states like λ & ψ appearing in the state space. The motivation behind KF implementation was prior belief that it was computationally efficient when compared to PF. However, the KF is a linear unbiased estimator and in the case of a noisy observation model, trusting a diverging process model would be suboptimal. We also tested in the case where many turns were made and this did cause false propagation of the state estimates for the KF. However, the KF is still mathematically the best linear unbiased estimator under ideal conditions. However, it is difficult to tune the KF when there could be outliers or stochastic RSSI anomalies. Therefore a recommendation to use the PF in the case of highly unpredictable environments would be advisable.

Our proposed PF using 5000 particles achieved the project objectives of $\mu_\varepsilon < 3$ m and $\sigma_\varepsilon < 2$ m. For computation time, this was difficult to determine due to the multithreading so a similar testbench script was used and attached to appendix to verify the computation times. We tested a sample script where propagation of particles and updates with resampling were all computed during run time.

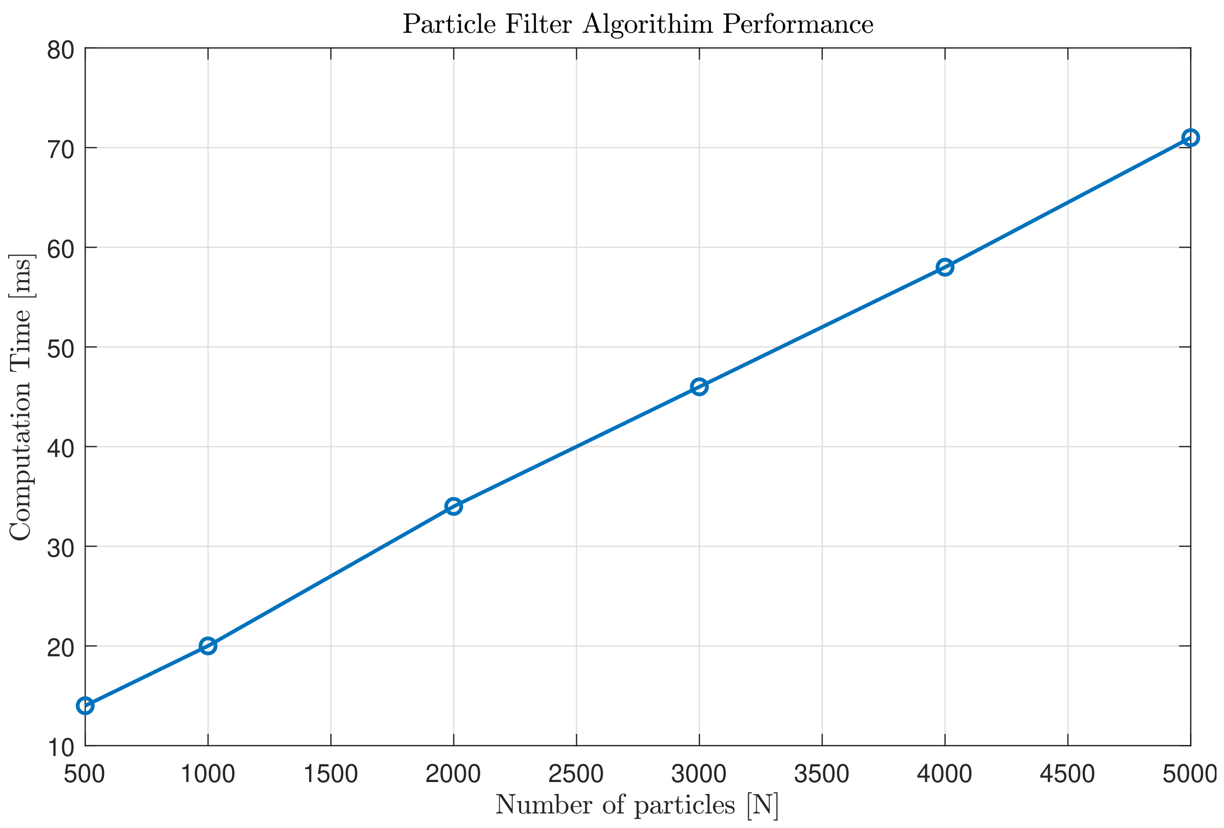


Figure 24: Particle Filter computational time

We note that the computational time for a given test script is well within our goal of $t_{comp} < 2$ s. For coverage area, FC-132 did not satisfy this criteria as we had limited room availability and this room was one of the few that was available during testing. Overall our proposed IPS has been successful in its project goals and objectives.

Chapter 6. Future Work

The next goal would be to reduce gyroscopic drift via magnetometer integration. This would require careful calibration and consideration of hard iron and soft iron distortions. Machine learning methods like RSSI fingerprinting is also a popular option if in the case of an outlier RSSI lateration measurement. Therefore the RSSI lateration method is still a proxy for distance and benefits can be gained by integrating the system with other solutions.

Another popular field of research is the idea of arbitrary phone positions which would require deep understanding of inertial reference frames for IMUs. Specific thresholding methods and conditional cases to handle phones in places like the pocket are emerging fields where ML models are trained to recognise certain patterns and behaviours for a more flexible IPS.

Overall, a final closing statement would be that modern computational power has allowed for previously computationally expensive methods to be implemented on resource constrained devices. Therefore, when approaching a complex non-linear model in which numerous measurements are either noisy or cannot be observed, the particle filter can be a viable solution for embedded devices.

References

[1] W. Elmenreich, "An introduction to sensor fusion," *Vienna University of Technology, Austria*, vol. 502, pp. 1–28, 2002.

[2] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[3] R. Mautz, "Overview of current indoor positioning systems," *Geodezija ir Kartografija*, vol. 35, no. 1, pp. 18–22, 2009. [Online]. Available: https://www.tandfonline.com/doi/abs/10.3846/1392-1541.2009.35.18-22

[4] Q. Liu, Z. Huang, and J. Wang, "Indoor non-line-of-sight and multipath detection using deep learning approach," *GPS Solutions*, vol. 23, no. 3, p. 75, May 2019. [Online]. Available: https://doi.org/10.1007/s10291-019-0869-4

[5] A. Goldsmith, *Path Loss and Shadowing.* Cambridge University Press, 2005, p. 27–63.

[6] C. K. M. Lee, C. M. Ip, T. Park, and S. Chung, "A bluetooth location-based indoor positioning system for asset tracking in warehouse," in *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 2019, pp. 1408–1412.

[7] W. C. Lee, F. H. Hung, K. F. Tsang, C. K. Wu, and H. R. Chi, "Rss-based localization algorithm for indoor patient tracking," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, 2016, pp. 1060–1064.

[8] S. Park and S. Hashimoto, "Indoor localization for autonomous mobile robot based on passive rfid," in *2008 IEEE International Conference on Robotics and Biomimetics*, 2009, pp. 1856–1861.

[9] H.-H. Hsu, J.-K. Chang, W.-J. Peng, T. K. Shih, T.-W. Pai, and K. L. Man, "Indoor localization and navigation using smartphone sensory data," *Annals of Operations Research*, vol. 265, no. 2, pp. 187–204, Jun 2018. [Online]. Available: https://doi.org/10.1007/s10479-017-2398-2

[10] S. G. Leitch, Q. Z. Ahmed, W. B. Abbas, M. Hafeez, P. I. Laziridis, P. Sureephong, and T. Alade, "On indoor localization using wifi, ble, uwb, and imu technologies," *Sensors*, vol. 23, no. 20, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/20/8598

[11] H. Obeidat, W. Shuaieb, O. Obeidat, and R. Abd-Alhameed, "A review of indoor localization techniques and wireless technologies," *Wireless Personal Communications*, vol. 119, pp. 289–327, 2021.

[12] R. Ramirez, C.-Y. Huang, C.-A. Liao, P.-T. Lin, H.-W. Lin, and S.-H. Liang, "A practice of ble rssi measurement for indoor positioning," *Sensors*, vol. 21, no. 15, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/15/5181

[13] M. Collotta, G. Pau, T. Talty, and O. K. Tonguz, "Bluetooth 5: A concrete step forward toward the iot," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 125–131, 2018.

[14] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.

[15] I. Bylemans, M. Weyn, and M. Klepal, "Mobile phone-based displacement estimation for opportunistic localisation systems," in *2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2009, pp. 113–118.

[16] P. Guo, H. Qiu, Y. Yang, and Z. Ren, "The soft iron and hard iron calibration method using extended kalman filter for attitude and heading reference system," in *2008 IEEE/ION Position, Location and Navigation Symposium.* IEEE, 2008, pp. 1167–1174.

[17] P. Neto, N. Mendes, and A. P. Moreira, "Kalman filter-based yaw angle estimation by fusing inertial and magnetic sensing: A case study using low cost sensors," *Sensor Review*, vol. 35, no. 3, pp. 244–250, 2015.

[18] V. Elvira, J. Miguez, and P. M. Djurić, "On the performance of particle filters with adaptive number of particles," *Statistics and Computing*, vol. 31, pp. 1–18, 2021.

[19] P. Wang and Y. Luo, "Research on wifi indoor location algorithm based on rssi ranging," in *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, 2017, pp. 1694–1698.

[20] C. Ma, B. Wu, S. Poslad, and D. R. Selviah, "Wi-fi rtt ranging performance characterization and positioning system design," *IEEE Transactions on Mobile Computing*, vol. 21, no. 2, pp. 740–756, 2022.

[21] J. Wang and J. Park, "An enhanced indoor positioning algorithm based on fingerprint using fine-grained csi and rssi measurements of ieee 802.11 n wlan," *Sensors*, vol. 21, no. 8, p. 2769, 2021.

[22] H. Perakis and V. Gikas, "Evaluation of range error calibration models for indoor uwb positioning applications," in *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2018, pp. 206–212.

[23] A. R. Jiménez Ruiz and F. Seco Granja, "Comparing ubisense, bespoon, and decawave uwb location systems: Indoor performance analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 8, pp. 2106–2117, 2017.

[24] S. Sadowski and P. Spachos, "Optimization of ble beacon density for rssi-based indoor localization," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2019, pp. 1–6.

[25] R. Safwat, E. Shaaban, S. M. Al-Tabbakh, and K. Emara, "Fingerprint-based indoor positioning system using ble: real deployment study," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 1, pp. 240–249, 2023.

[26] K. Huang, K. He, and X. Du, "A hybrid method to improve the ble-based indoor positioning in a dense bluetooth environment," *Sensors*, vol. 19, no. 2, p. 424, 2019.

[27] J. Li, M. Guo, and S. Li, "An indoor localization system by fusing smartphone inertial sensors and bluetooth low energy beacons," in *2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST)*, 2017, pp. 317–321.

[28] K. Sung, D. K. R. Lee, and H. Kim, "Indoor pedestrian localization using ibeacon and improved kalman filter," *Sensors*, vol. 18, no. 6, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/6/1722

[29] Y. You and C. Wu, “Hybrid indoor positioning system for pedestrians with swinging arms based on smartphone imu and rssi of ble,” *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–15, 2021.

[30] F. Ye, R. Chen, G. Guo, X. Peng, Z. Liu, and L. Huang, “A low-cost single-anchor solution for indoor positioning using ble and inertial sensor data,” *IEEE Access*, vol. 7, pp. 162 439–162 453, 2019.

[31] T.-M. T. Dinh, N.-S. Duong, and Q.-T. Nguyen, “Developing a novel real-time indoor positioning system based on ble beacons and smartphone sensors,” *IEEE Sensors Journal*, vol. 21, no. 20, pp. 23 055–23 068, 2021.

[32] G. Guo, R. Chen, F. Ye, Z. Liu, S. Xu, L. Huang, Z. Li, and L. Qian, “A robust integration platform of wi-fi rtt, rss signal, and mems-imu for locating commercial smartphone indoors,” *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16 322–16 331, 2022.

[33] *ENERGIZER E91*, ENERGIZER, 3 2024. [Online]. Available: https://www.batteryspace.com/prod-specs/1142_specification.pdf

[34] *Arduino Nano ESP32*, Arduino, 12 2023. [Online]. Available: https://docs.arduino.cc/resources/datasheets/ABX00083-datasheet.pdf

[35] *Raspberry Pi Pico W*, Raspberry Pi Ltd, 6 2022. [Online]. Available: https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf

[36] *BNO085*, Ceva Inc., 6 2022. [Online]. Available: https://www.ceva-ip.com/wp-content/uploads/2019/10/BNO080_085-Datasheet.pdf

[37] C. Gomez, J. Oller, and J. Paradells, “Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology,” *sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[38] R. Faragher and R. Harle, “Location fingerprinting with bluetooth low energy beacons,” *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2418–2428, 2015.

[39] “Bluetooth assigned numbers,” https://www.bluetooth.com/specifications/assigned-numbers/, Bluetooth Special Interest Group, accessed: [15/12/2023].

[40] M. Nikodem and M. Bawiec, “Experimental evaluation of advertisement-based bluetooth low energy communication,” *Sensors*, vol. 20, no. 1, p. 107, 2019.

[41] Infineon Technologies AG, *CYW43439 Single-Chip IEEE 802.11 b/g/n MAC/Baseband/Radio with Bluetooth 5.2 + EDR and HS*, Mar. 2024, data Sheet, Version 05.00. [Online]. Available: https://www.infineon.com/dgdl/Infineon-CYW43439-DataSheet-v05_00-EN.pdf?fileId=8ac78c8c8929aa4d01893ee30e391f7a

[42] B. Everard, “Getting to grips with bluetooth on pico w,” 2022, accessed: [15/12/2023]. [Online]. Available: https://www.raspberrypi.com/news/getting-to-grips-with-bluetooth-on-pico-w/

[43] K. Peppas, H. Nistazakis, and G. Tombras, “An overview of the physical insight and the various performance metrics of fading channels in wireless communication systems,” *Advanced trends in wireless communications*, pp. 1–22, 2011.

[44] T.-M. T. Dinh, N.-S. Duong, and Q.-T. Nguyen, “Developing a novel real-time indoor positioning system based on ble beacons and smartphone sensors,” *IEEE Sensors Journal*, vol. 21, no. 20, pp. 23 055–23 068, 2021.

[45] G. Guo, R. Chen, F. Ye, Z. Liu, S. Xu, L. Huang, Z. Li, and L. Qian, “A robust integration platform of wi-fi rtt, rss signal, and mems-imu for locating commercial smartphone indoors,” *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16 322–16 331, 2022.

[46] M. Ibrahim and O. Moselhi, “Enhanced localization for indoor construction,” *Procedia Engineering*, vol. 123, pp. 241–249, 2015.

[47] M. Wrona, “Accelerometer calibration,” 2023, accessed on: [16/12/2023]. [Online]. Available: https://github.com/michaelwro/accelerometer-calibration

[48] K. Mekki, E. Bajic, and F. Meyer, “Indoor positioning system for iot device based on ble technology and mqtt protocol,” in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 04 2019, p. Specific page numbers if available.

[49] J. Yang and Y. Chen, “Indoor localization using improved rss-based lateration methods,” in *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, 2009, pp. 1–6.

[50] Å. Björck, “Least squares methods,” *Handbook of numerical analysis*, vol. 1, pp. 465–652, 1990.

[51] V. Cantón Paterna, A. Calveras Auge, J. Paradells Aspas, and M. A. Perez Bullones, “A bluetooth low energy indoor positioning system with channel diversity, weighted trilateration and kalman filtering,” *Sensors*, vol. 17, no. 12, p. 2927, 2017.

[52] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, “An estimate for the condition number of a matrix,” *SIAM Journal on Numerical Analysis*, vol. 16, no. 2, pp. 368–375, 1979.

[53] J. Lv, A. A. Ravankar, Y. Kobayashi, and T. Emaru, “A method of low-cost imu calibration and alignment,” in *2016 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2016, pp. 373–378.

[54] MathWorks, “Calibrate accelerometer,” accessed: December 18, 2023. [Online]. Available: https://uk.mathworks.com/help/supportpkg/beagleboneblue/ref/calibrateaccel.html

[55] S. Instruments, “Download magneto v1.2,” accessed: December 18, 2023. [Online]. Available: https://sites.google.com/view/sailboatinstruments1

[56] A. Alaimo, V. Artale, C. Milazzo, and A. Ricciardello, “Comparison between euler and quaternion parametrization in uav dynamics,” in *AIP Conference Proceedings*, vol. 1558, no. 1. American Institute of Physics, 2013, pp. 1228–1231.

[57] S. Madgwick *et al.*, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays,” *Report x-io and University of Bristol (UK)*, vol. 25, pp. 113–118, 2010.

[58] R. Mahony, T. Hamel, and J.-M. Pflimlin, “Nonlinear complementary filters on the special orthogonal group,” *IEEE Transactions on automatic control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[59] Y. Zhang, Y. Li, C. Peng, D. Mou, M. Li, and W. Wang, “The height-adaptive parameterized step length measurement method and experiment based on motion parameters,” *Sensors*, vol. 18, p. 1039, 03 2018.

[60] L. Litwin, “Fir and iir digital filters,” *IEEE Potentials*, vol. 19, no. 4, pp. 28–31, 2000.

[61] E. Billauer, “peakdet: Peak detection using MATLAB (non-derivative local extremum, maximum, minimum),” Blog Post, January 2009, accessed: 2023-04-14. [Online]. Available: https://billauer.co.il/blog/2009/01/peakdet-matlab-octave/

[62] H. Weinberg, “Using the adxl202 in pedometer and personal navigation applications,” *Analog Devices AN-602 application note*, vol. 2, no. 2, pp. 1–6, 2002.

[63] L. Russell, A. L. Steele, and R. Goubran, “Stride time estimation: Realtime peak detection implemented on an 8-bit, portable microcontroller,” in *2012 IEEE International Symposium on Medical Measurements and Applications Proceedings*, 2012, pp. 1–5.

[64] J. Röbesaat, P. Zhang, M. Abdelaal, and O. Theel, “An improved ble indoor localization with kalman-based fusion: An experimental study,” *Sensors*, vol. 17, no. 5, p. 951, 2017.

[65] J. Liang, “Bayesian approaches to tracking, sensor fusion and intent prediction,” Ph.D. dissertation, University of Cambridge, 2020.

[66] R. Douc and O. Cappé, “Comparison of resampling schemes for particle filtering,” in *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.* Ieee, 2005, pp. 64–69.

[67] curiores, “High pass filter design - basic filters,” https://github.com/curiores/ArduinoTutorials/tree/main/BasicFilters/Design/HighPass, 2024, accessed: 2024-04-15.

Appendices

Appendix A. 4-th Order Low-pass Butterworth Filter Coefficients Derivation

This section aims to derive the 4th Order Low-pass Butterworth filter in the form of a difference equation. This derivation will provide the IIR filter coefficients needed for the embedded device implementation. This derivation is an extension to the 2nd order derivation and will therefore use the same procedure [67]. We can express the continuous-time transfer function $H(s)$ for a Butterworth filter with cutoff frequency $\omega_c = 1$ as follows.

$$H(s) = \frac{1}{\sum_{k=0}^{n} \beta_k s^k}$$

Where $s = \sigma + \boldsymbol{j}\omega$ for the complex Laplace variable s, n is the order of the filter, and β_k are the Butterworth coefficients. These β_k coefficients are given by the following recursion formula.

$$\beta_{k+1} = \frac{\cos\left(\frac{k\pi}{2n}\right)}{\sin\left(\frac{(k+1)\pi}{2n}\right)}\beta_k$$

Using $\beta_0 = 1$ and the recursion formula, we can obtain the 4th Order Butterworth coefficients as follows.

$$\{\beta_k\}_{k=0}^{4} = \left\{1,\ \frac{1}{\sin\left(\frac{\pi}{8}\right)},\ \frac{\cos\left(\frac{\pi}{8}\right)}{\sin\left(\frac{\pi}{4}\right)\sin\left(\frac{\pi}{8}\right)},\ \frac{\cos\left(\frac{\pi}{8}\right)}{\sin\left(\frac{3\pi}{8}\right)\sin\left(\frac{\pi}{8}\right)},\ 1\right\}$$

We can express the n^{th} order Butterworth polynomial in the Laplace domain with cutoff frequency ω_c as follows.

$$B_n(s) = \sum_{k=0}^{n} \frac{\beta_k}{\omega_c^k} s^k$$

We can express the 4th Order transfer function in Laplace domain as follows.

$$H(s) = \frac{1}{\beta_0 + \frac{\beta_1}{\omega_c}s + \frac{\beta_2}{\omega_c^2}s^2 + \frac{\beta_3}{\omega_c^3}s^3 + \frac{\beta_4}{\omega_c^4}s^4}$$

We can express the discrete transfer function using Tustin's method (bilinear transform) wherein $s = \frac{2}{\Delta t}\frac{(1-z^{-1})}{(1+z^{-1})}$.

$$H(z) = \frac{1}{\beta_0 + \frac{\beta_1}{\omega_c}\left[\frac{2}{\Delta t}\frac{(1-z^{-1})}{(1+z^{-1})}\right] + \frac{\beta_2}{\omega_c^2}\left[\frac{2}{\Delta t}\frac{(1-z^{-1})}{(1+z^{-1})}\right]^2 + \frac{\beta_3}{\omega_c^3}\left[\frac{2}{\Delta t}\frac{(1-z^{-1})}{(1+z^{-1})}\right]^3 + \frac{\beta_4}{\omega_c^4}\left[\frac{2}{\Delta t}\frac{(1-z^{-1})}{(1+z^{-1})}\right]^4}$$

Which then simplifies to the following.

$$H(z) = \frac{1}{\beta_0 + \frac{2\beta_1}{\omega_c \Delta t}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) + \frac{4\beta_2}{\omega_c^2 \Delta t^2}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)^2 + \frac{8\beta_3}{\omega_c^3 \Delta t^3}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)^3 + \frac{16\beta_4}{\omega_c^4 \Delta t^4}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)^4}$$

By substituting $\alpha = \omega_c \Delta t$ we have the following.

$$H(z) = \frac{1}{\beta_0 + \frac{2\beta_1}{\alpha}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) + \frac{4\beta_2}{\alpha^2}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)^2 + \frac{8\beta_3}{\alpha^3}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)^3 + \frac{16\beta_4}{\alpha^4}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)^4}$$

Which then simplifies to the following.

$$H(z) = \frac{\alpha^4 \left(1+z^{-1}\right)^4}{\sum_{k=0}^{4} 2^k \alpha^{4-k} \beta_k \left(1-z^{-1}\right)^k \left(1+z^{-1}\right)^{4-k}}$$

We can use the Binomial theorem to expand the $\left(1-z^{-1}\right)^k \left(1+z^{-1}\right)^{4-k}$ terms. After expansion, we make the following substitution: $D = \alpha^4\beta_0 + 2\alpha^3\beta_1 + 4\alpha^2\beta_2 + 8\alpha\beta_3 + 16\beta_4$. This would lead to a simplified form for $H(z)$.

$$H(z) = \frac{\frac{\alpha^4}{D}z^{-4} + \frac{4\alpha^4}{D}z^{-3} + \frac{6\alpha^4}{D}z^{-2} + \frac{4\alpha^4}{D}z^{-1} + \frac{\alpha^4}{D}}{1 + \left(\frac{4\alpha^4\beta_0 + 4\alpha^3\beta_1 - 16\alpha\beta_3 - 64\beta_4}{D}\right)z^{-1} + \left(\frac{6\alpha^4\beta_0 - 8\alpha^2\beta_2 + 96\beta_4}{D}\right)z^{-2} + \cdots}$$

Let us consider the difference equation for the IIR filter in the following form.

$$y[n] = a_1 y[n-1] + a_2 y[n-2] + \cdots + b_0 x[n] + b_1 x[n-1] + \cdots$$

This IIR filter can be expressed as a discrete transfer function in the following form.

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

We can compare the IIR discrete transfer function to the discrete transfer function of our Butterworth filter. Upon inspection, we can obtain the Butterworth low-pass filter coefficients.

$$
\begin{aligned}
b_0 &= \frac{\alpha^4}{D} \\
b_1 &= \frac{4\alpha^4}{D} \\
b_2 &= \frac{6\alpha^4}{D} \\
b_3 &= \frac{4\alpha^4}{D} \\
b_4 &= \frac{\alpha^4}{D} \\
a_0 &= \frac{-4\alpha^4\beta_0 - 4\alpha^3\beta_1 + 16\alpha\beta_3 + 64\beta_4}{D} \\
a_1 &= \frac{-6\alpha^4\beta_0 + 8\alpha^2\beta_2 - 96\beta_4}{D} \\
a_2 &= \frac{-4\alpha^4\beta_0 + 4\alpha^3\beta_1 - 16\alpha\beta_3 + 64\beta_4}{D} \\
a_3 &= \frac{-\alpha^4\beta_0 + 2\alpha^3\beta_1 - 4\alpha^2\beta_2 + 8\alpha\beta_3 - 16\beta_4}{D}
\end{aligned}
$$

This concludes the complete derivation for the Butterworth filter coefficients for an IIR, 4th order low-pass filter designed using Tustin's method.