Design of orientation and positioning algorithms for inertial measurement units

Diseño de algoritmos de orientación y posicionamiento para unidades de medición inercial

TRABAJO DE FIN DE GRADO

Lucía Alonso Mozo

Director:
Daniel Báscones García

Facultad de Informática
Universidad Complutense de Madrid
10 (SB)

Mayo 2025

A mi familia.

To my family.

Agradecimientos

A mis padres, por apoyarme siempre, enseñarme la importancia del esfuerzo, la constancia y a no rendirme.

A mi hermana, por ser desde siempre un ejemplo a seguir y por motivarme a dar lo mejor de mí.

A los amigos que he conocido en esta etapa, en especial a los *escape roomers*, por todas las horas compartidas en la biblioteca y en los exámenes.

A Javi, por confiar en mí incondicionalmente y estar siempre a mi lado.

A Daniel Báscones, por acompañarme durante el desarrollo de este trabajo y por dedicarme su tiempo, conocimientos y entusiasmo. Sin su ayuda, este proyecto no habría sido lo mismo.

Abstract

The ability to determine one's orientation and position has always been fundamental for human beings. While ancient navigation relied on natural cues such as the stars or magnetic compasses, modern systems use advanced sensors that provide accurate, real-time orientation estimation.

This project focuses on the detailed analysis of an Inertial Measurement Unit (IMU), particularly the accelerometer, gyroscope, and magnetometer, which measure specific force, angular velocity, and the Earth's magnetic field, respectively. Several techniques were evaluated to compute spatial orientation from these sensors, including sensor fusion algorithms based on Kalman filters and Attitude and Heading Reference Systems (AHRS).

The initial motivation behind the project was to explore alternatives to GPS in scenarios where satellite connectivity is unavailable or unreliable, such as indoors, tunnels, or remote areas. In this context, gait tracking was also investigated as a viable solution for estimating a person's walking path using only inertial sensors.

Additionally, a comparative evaluation was conducted on various data acquisition methods, including polling, interrupt-driven acquisition, and the use of the Digital Motion Processor (DMP), with respect to their power consumption. A series of experiments were carried out using a development board with the ICM-20948 sensor and a precision power profiler.

The outcome offers a comprehensive perspective on both the accuracy and energy impact of orientation techniques, which is crucial for the design of low-power embedded systems such as wearable devices or autonomous navigation platforms.

Keywords: IMU, orientation, accelerometer, gyroscope, magnetometer, power consumption, gait tracking.

Resumen

La capacidad de orientarse y conocer la propia posición ha sido siempre esencial para los seres humanos. Mientras que en la antigüedad se empleaban métodos naturales como la observación de los astros o el uso de brújulas magnéticas, en la actualidad disponemos de sensores avanzados que permiten una estimación precisa de la orientación en tiempo real.

Este trabajo se centra en el estudio detallado de una Unidad de Medida Inercial (IMU), en particular los sensores de aceleración, giroscopio y magnetómetro, que permiten medir fuerzas específicas, velocidad angular y el campo magnético terrestre, respectivamente. Se han analizado diversas formas de obtener la orientación espacial a partir de estos sensores, incluyendo el uso de algoritmos como la fusión de sensores mediante filtros de tipo Kalman o sistemas de referencia de actitud y rumbo (AHRS).

El planteamiento inicial del trabajo surgió de la necesidad de disponer de una solución para contextos donde el uso del GPS es inviable o está limitado, como por ejemplo en interiores, túneles, o zonas sin cobertura de satélite. En este sentido, se estudió también el concepto de análisis de la marcha como alternativa viable para estimar el recorrido de una persona caminando, utilizando únicamente sensores inerciales.

Además, se ha llevado a cabo una evaluación comparativa de diferentes métodos de adquisición de datos, incluyendo lectura por sondeo, interrupciones y el uso del DMP (Digital Motion Processor), en función de su consumo energético. Para ello, se realizaron múltiples experimentos utilizando una placa de desarrollo equipada con un sensor ICM-20948, junto con un medidor de consumo de precisión.

Como resultado, se obtuvo una visión integral no solo de la precisión de las técnicas de orientación, sino también del impacto energético de cada una, lo cual es esencial para el diseño de sistemas embebidos de bajo consumo, como dispositivos portátiles o sistemas de navegación autónoma.

Palabras clave: IMU, orientación, acelerómetro, giroscopio, magnetómetro, consumo energético, análisis de la marcha.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 History of orientation and navigation systems

Navigation encompasses a set of knowledge and methods used to determine the spatial position of objects relative to specific landmarks. Yet, the fundamental question of "Where am I?" remains a timeless challenge. From taking a wrong turn or missing a highway exit to finding oneself lost in unfamiliar places, the struggle to determine one's position has persisted throughout human history.

The origins of orientation and navigation trace back to early human migration out of the African savannah. Our ancestors relied on permanent natural landmarks such as rivers and trails to guide their movements in search of food and water. When crossing featureless landscapes, they developed directional reference systems to ensure they could return to their starting point [8].

Before the advent of air travel, intercontinental transport was limited to maritime routes. As maritime navigation advanced, calculating one's route required precise time and distance measurements. Early navigators oriented themselves by observing celestial bodies, such as the sun, moon, and stars [48]. For example, they understood that the altitude of the North Star indicated their latitude, and that at any given point on Earth, the sun reaches its highest altitude at noon [8].

The understanding of the heliocentric model and Eratosthenes of Alexandria's measurement of Earth's radius significantly advanced navigation. These developments laid the groundwork for the creation of geographical maps based on latitude and longitude grids [8].

Navigation tools evolved with the introduction of the astrolabe. This instrument allowed users to measure the angle of the sun, moon, planets, or stars. It also helped determine the sun's position and that of major stars relative to the meridian and horizon, enabling users to calculate their geographic latitude and locate true north, even during the day when stars were not visible [41].

The astrolabe eventually evolved into the sextant [18], a more advanced instrument that enabled navigators to determine both latitude and longitude by measuring angles between celestial bodies.

The 20^{th} century brought significant advancements in navigation through the advent of radio technology. In the early 1900s, direction-finding radio stations allowed ships and aircraft in distress to receive navigational assistance. This innovation advanced further in 1926 when the U.S. National Bureau of Standards established a nationwide system of directional radio beams, greatly enhancing commercial aviation navigation [27].

Figure 1.1: The earliest astrolabe ever found [37].

These developments culminated in the creation of the Global Positioning System (GPS), a highly sophisticated navigation system comprising a constellation of satellites orbiting at an altitude of 20,200 kilometers. GPS provides highly accurate spatial and temporal data to users on land, sea, air, and even in space, including the International Space Station. Developed in the late 1960s, GPS has since evolved into a globally accessible system offering free, unrestricted service to users worldwide [13].

Modern positioning and navigation solutions are typically divided into two main categories: relative positioning, which includes methods such as odometry and inertial navigation, and absolute positioning, which includes techniques like magnetic compass readings, beacon-based triangulation, GPS, landmark recognition, and model matching [12]. Due to the limitations inherent in any single method, hybrid systems that combine relative and absolute approaches are commonly employed to improve navigational accuracy.

1.2 Living being orientation

As mentioned in [1], orientation with respect to the environment is one of the most important capabilities a living being must develop. All living organisms orient themselves to satisfy their primary needs, those related to movement, survival, growth, and reproduction.

Plants orient their leaves and flowers toward light in search of energy, while their roots grow toward the ground in search of water.

Animals, in contrast, seek stimuli from the external environment to develop their senses and locate favorable conditions for survival. Over time, animals have developed specialized receptors, such as vision, hearing, and smell, to perceive their surroundings. These stimuli are then processed using a phylogenetic code that allows animals to orient themselves and modify their behavior accordingly. Notable examples include bats, migratory birds, and fish such as salmon.

Humans also require orientation, not necessarily to acquire vital resources, but to better understand and enjoy the world around them. For example, in earlier cultures, smells were often closely related to identity and environment. With societal evolution, many orientation skills once essential, like following animal tracks, navigating using sounds, smells, or stars, have become less critical. As a result, some human sensory functions have become less active. Also, the development of orientation instruments has contributed to a reduced reliance on innate human orientation capabilities.

The natural mechanism of human orientation consists of several stages. First, humans receive information through their sensory receptors. This information is transmitted to the brain through afferent pathways, where it is processed. The brain then sends commands via efferent pathways to the effector organs, which execute the actions. Once the action is completed, the brain's relationship to its surroundings changes, forming a feedback loop.

1.2.1 Auditory Orientation

It is well established that one of the primary techniques that living beings use for orientation, balance, and gait involves auditory input.

Hearing is crucial for maintaining balance and proper locomotion. For example, a study comparing deaf and hearing students [36] concluded that deaf students exhibited changes in gait and were at greater risk of falling than their hearing counterparts.

This phenomenon is not exclusive to humans. Insects, despite having small auditory organs, demonstrate remarkably advanced auditory processing. Recent research [46] highlights how flies acoustically locate targets in space, how mosquitoes have evolved highly sensitive ears, and how crickets avoid deafening themselves with their own songs. These findings reveal the extraordinary analytical abilities of highly specialized microscale auditory systems.

1.2.2 Olfactory Orientation

Although theory has long suggested it, there was no direct evidence that animals can define an arbitrary location in space based on an odor grid until recently.

A study [25] demonstrated that humans may, in fact, possess this ability. In the experiment, participants were guided to a random location in a room filled with two diffused odors. After briefly sampling the environment and undergoing spatial disorientation, they were asked to return to the same location. The experiment was carried out under three conditions: olfactory-only, visual-only, and a control condition without olfactory, visual, or auditory cues. The results showed that participants were significantly more accurate in the olfactory-only condition compared to the control, and also exceeded chance-level accuracy.

1.3 Inertial Measurement Unit

In the context of inertial navigation, the *Inertial Measurement Unit* (IMU) is a fundamental component. An IMU is an electronic device used to measure and report a body's specific force, angular velocity, and, in some configurations, its orientation [65]. These measurements are obtained through a combination of accelerometers and gyroscopes, and optionally, magnetometers. When magnetometers are included, the system is referred to as an *Inertial Magnetic Measurement Unit* (IMMU).

IMUs play a critical role in the navigation and control of a wide array of modern vehicles, including motorcycles, missiles, aircraft (where they underpin Attitude and Heading Reference Systems), unmanned aerial vehicles (UAVs), and spacecraft such as satellites and planetary landers [65].

Recent technological advances have enabled the integration of IMUs with Global Positioning System (GPS) receivers. This fusion allows for continued navigation in GPS-degraded or denied environments such as tunnels, indoor facilities, or regions affected by electronic interference.

A historically significant example of an IMU is shown in Figure 1.2, which depicts the Apollo IMU [56]. This unit was a vital component of the Apollo Primary Guidance, Navigation, and Control System (PGNCS), a self-contained inertial navigation system. It allowed the Apollo spacecraft to autonomously navigate and perform mission-critical operations, even during communication blackouts, such as when orbiting the far side of the Moon or encountering unexpected radio failures.

The Apollo IMU was mounted on a three-axis gimbal system and contained three gyroscopes and three accelerometers. Despite its precision, the system exhibited a drift of approximately

one milliradian per hour. To mitigate this, the inertial platform was periodically realigned using star sightings, ensuring long-term accuracy throughout the mission.

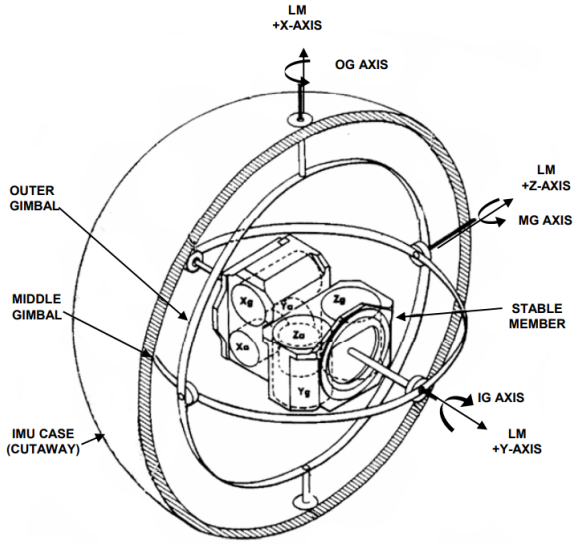

Figure 1.2: Apollo IMU, where gyros sense attitude changes, and accelerometers sense velocity changes [65].

1.3.1 Accelerometer, gyroscope and magnetometer

As previously mentioned, IMUs are capable of measuring acceleration, angular velocity, and magnetic field strength along the three spatial axes (x, y, and z). Each of them is measured in a different unit. While integrating measured rates provides position, inertial sensor data suffers from drift. This is because constant errors accumulate during the integration process, leading to unbounded position errors over time. Consequently, inertial sensors are not reliable for long-term precise positioning. These sensors have some shortcomings, which will be addressed in Section 2.5.

Accelerometer

An accelerometer, as illustrated in Figure 1.3, is a transducer used to measure the physical or measurable acceleration experienced by an object. It is an electromechanical device that determines the specific force acting on the object. This force results from the weight phenomenon when the object remains within the frame of reference of the accelerometer [26].

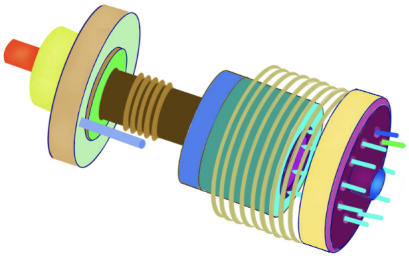

Figure 1.3: Diagram of a typical accelerometer [26].

They can operate based on various principles, with the majority being piezoelectric types. To understand their underlying mechanism, it is important to introduce the concept of the

piezoelectric effect [9]. A piezoelectric accelerometer typically incorporates a quartz crystal with piezoelectric properties. When subjected to an external acceleration, the resulting force acts on the crystal, which generates a voltage proportional to that force due to its self-generating nature. The operating principle of the sensor is illustrated in Figure 1.4.

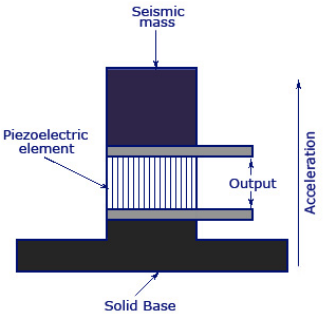

Figure 1.4: Schematic of a piezoelectric accelerometer [26].

Although m/s^2 is the international SI unit for acceleration, g is also used as a unit for acceleration, relative to standard gravity ($1g \approx 9.81 m/s^2$). An accelerometer at rest relative to the Earth's surface will indicate approximately $1g$ upward because the Earth's surface exerts a normal force upward relative to the local inertial frame.

Accelerometers suffer from extensive drift and are sensitive to uneven ground because any disturbance of a perfectly horizontal position will cause the sensor to detect a component of gravitation acceleration g [12].

Gyroscope

Gyroscopes, or gyros, are devices used to measure or maintain rotational motion [63]. MEMS (Micro-Electro-Mechanical Systems) gyroscopes are small, cost-effective sensors that measure angular velocity. This velocity is typically expressed in degrees per second (°/s) or revolutions per second (RPS).

When an object rotates around an axis, it possesses angular velocity. A triple-axis MEMS gyroscope is capable of measuring rotation around three perpendicular axes: x, y, and z. Although there are single- and dual-axis variants, the triple-axis version integrated into a single chip has become increasingly popular due to its decreasing size and cost. These sensors are extremely compact, ranging from 1 to 100 micrometers in size [67].

As illustrated in Figure 1.5, MEMS gyroscopes detect angular velocity by observing the displacement of a small resonating mass (m) attached to an internal frame via a spring-damper system as the device rotates. An actuator, controlled by a positive feedback loop (not depicted), drives the mass into harmonic oscillation along the x-axis. This feedback loop ensures that the oscillation remains regulated and continuous. The internal frame is further connected to the sensor housing through a second spring-damper structure oriented orthogonally to the direction of oscillation. This setup enables movement along the y-axis in response to external forces [5].

This configuration allows the sensor to detect angular velocity when it is rotated about an axis perpendicular to the XY plane shown in the diagram.

In Figure 1.6, the gyroscope is depicted at two different time instants. While the distance of the sensor package from the center of the rotating platform remains constant, the oscillating proof mass shifts its position in the x-direction from r_1 to r_2, with $r_2 > r_1$. This variation results in a tangential acceleration along the positive y-axis.

The proof mass is rigidly connected to the internal frame along the y-axis. As a result, this

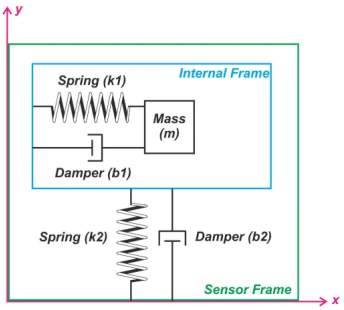

Figure 1.5: Basic structure of a vibratory gyroscope [5].

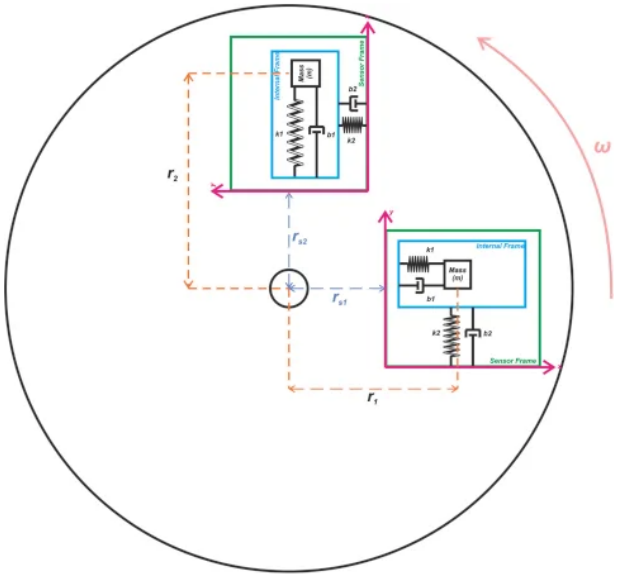

Figure 1.6: A gyroscope on a fixed platform rotating counterclockwise [5].

acceleration is transmitted to the internal frame, which initially tends to stay at rest due to inertia. This creates a relative displacement between the internal frame and the sensor housing.

This relative movement is proportional to the input angular velocity and can be measured through various methods, including capacitive sensing techniques.

Magnetometer

A magnetometer measures magnetic fields or magnetic dipole moments, determining their direction, strength, or changes at a specific point [66]. They can be broadly categorized as scalar or vector types. The ones used in this project are vector magnetometers, which are capable of measuring the magnetic field's components along specific directions relative to the device's orientation.

At any point on Earth, the magnetic field can be approximated locally by a constant three-dimensional vector, denoted as $\mathbf{h}_0$. This vector is defined by three main characteristics: its magnitude ($\mathcal{F}$), inclination ($\mathcal{I}$), and declination ($\mathcal{D}$), which are illustrated in Figure 1.7.

Using the reference frame shown in the figure, the magnetic field vector $\mathbf{h}_0$ can be written as shown in Equation (1.1):

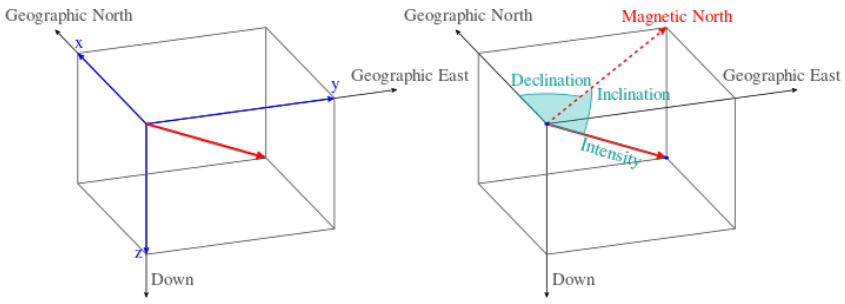

Figure 1.7: Declination, magnitude, and inclination [22].

$$\mathbf{h}_0 = \mathcal{F} \begin{bmatrix} \cos(\mathcal{I}) \cos(\mathcal{D}) \\ \cos(\mathcal{I}) \sin(\mathcal{D}) \\ \sin(\mathcal{I}) \end{bmatrix} \tag{1.1}$$

While a magnetic compass provides only a horizontal bearing, a vector magnetometer measures the complete magnetic field vector, including both magnitude and direction. To capture the field's components in all three spatial dimensions, a vector magnetometer requires three orthogonal sensors. The strength of a magnetic field is measured in units of tesla in the SI units, and in gauss in the cgs system of units (10,000 gauss are equal to one tesla).

One of the operating principles used by magnetometers is based on the Hall effect [64]. This effect refers to the generation of a voltage difference, known as the Hall voltage, across an electrical conductor when it is subjected to a magnetic field that is perpendicular to both the conductor and the direction of the electric current flowing through it.

In practical terms, magnetometers that utilize the Hall effect, as seen in Figure 1.8, employ semiconducting materials through which an electric current is passed. When a magnetic field is present in the vicinity, it causes a distortion or deflection in the path of the current, resulting in the appearance of the Hall voltage. This voltage is directly proportional to the strength of the magnetic field, allowing the magnetometer to measure its magnitude by analyzing this induced voltage [53].

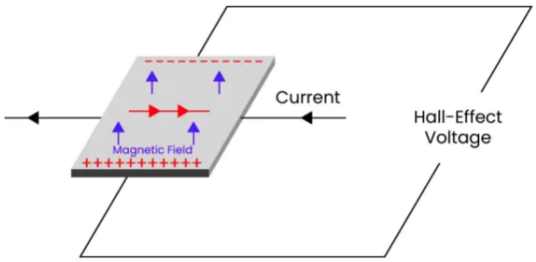

Figure 1.8: Schematic of a magnetometer using the Hall effect [53].

1.4 Potential uses

IMUs let people estimate their movements and orientation effectively. This has various potential applications, particularly in positioning objects in space and gait tracking.

1.4.1 Navigation and localization

Indoor navigation

GPS is highly effective for navigation in vehicles, public transport, or even walking, but it struggles indoors. Although it can indicate whether someone is in their bedroom, for example, it cannot determine whether they are lying on the bed, sitting at a desk, or standing near a wardrobe. Moreover, if the GPS signal is unavailable, such as when there is no network connectivity, it may not even be able to tell whether the person is in the bedroom or a different room. In contrast, IMU-based systems enable real-time positioning, making them useful for applications such as warehouse automation and airport navigation.

Simultaneous localization and mapping (SLAM)

One notable application is Simultaneous Localization and Mapping (SLAM), a technique that combines IMUs with distance sensors to determine an object's position while mapping an environment. The solution of the SLAM problem has seen rapid improvement in the last few decades, either using active sensors like Radio Detection and Ranging (Radar) and Light Detection and Ranging (LiDAR) or passive sensors like cameras [3].

SLAM is widely used in robotics, autonomous vehicles, and augmented reality. However, the main drawback of SLAM systems is their high cost. In contrast, an accelerometer costs as little as 5€, making IMU-based solutions significantly more affordable.

Subterranean navigation

Where GPS lacks signal, for example, in mines, tunnels, or underwater environments, the use of IMUs can be essential for exploration and safety.

1.4.2 Motion tracking and human movement analysis

Gait analysis [61] is the systematic study of animal locomotion, more specifically the study of human motion.

Human motion tracking systems

Human motion tracking systems have been widely utilized in medicine [73], biomechanics [14], filmmaking, games, or humanoid robotics [30]. A tracking system typically consists of multiple IMUs that are attached to body segments, which is easy to operate and portable, as seen in Figure 1.9.

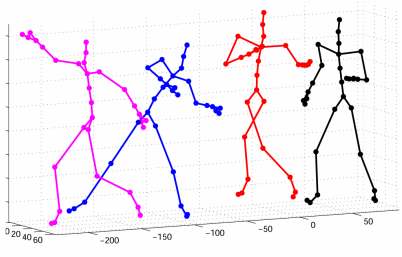

Figure 1.9: Simulation of human movements [11].

Wearable technology

Smart devices, such as smart watches and fitness trackers, are equipped with inertial measurement units, enabling a wide range of motion-based applications. These include hand gesture recognition [4], which allows intuitive user interaction and control of digital systems, as well as arm pose estimation [19], which is particularly useful in applications such as virtual reality, rehabilitation, and human-computer interaction. In addition, IMUs can be used for real-time detection of at-risk movements [6], helping to prevent falls and assess injury risk, particularly for elderly people or workers in physically demanding environments.

Sports performance monitoring

IMUs can also play a crucial role in sports science by helping athletes analyze and refine their techniques [55]. These sensors enable objective performance assessment, injury prevention strategies, and improved training efficiency. For example, IMUs can be attached to sports equipment, such as rackets, to provide real-time feedback to both athletes and coaches, enhancing decision making during training sessions or competitions [70]. In addition, IMUs are valuable in swimming, where they can accurately measure body angles and movement patterns, contributing to the optimization of the technique [23]. Beyond specific sports, these sensors facilitate biomechanical analysis, allowing for precise estimation of hip and knee joint kinematics [7], which is particularly useful for monitoring and improving athletic performance in activities such as running, jumping, and other dynamic movements.

1.4.3 Robotics and Industrial Automation

Drone stabilization and navigation

A fundamental challenge in autonomous navigation is to ensure safe operation within an unknown environment. Most existing approaches rely on GPS to determine the absolute position of a drone. However, GPS accuracy may not be sufficient for certain applications, particularly in environments with weak or no satellite coverage. To address this limitation, positioning systems that integrate monocular SLAM with IMUs have emerged as effective alternatives [38]. These systems not only improve localization accuracy, but can also be beneficial for camera calibration.

Factory automation

Industrial robotic arms rely on motion tracking to perform high-precision tasks such as assembly, welding, and quality inspection. Research has demonstrated that integrating stereoscopic vision with IMU sensors can enhance the real-time performance of industrial robots [43]. Additionally, the fusion of camera systems with IMU sensors has been shown to play a key role in the development of smart factories, enabling more efficient and autonomous manufacturing processes [54].

1.4.4 Military and aerospace applications

Missile and aircraft navigation

Miniature gun-hardened MEMS IMUs have been developed for applications in guided projectiles, rockets, and missiles [47]. These sensors are specifically designed to withstand the extreme shock and vibration conditions experienced during a guided projectile gun launch event while maintaining their functionality and performance.

Soldier tracking

As discussed previously, IMUs can be utilized to track human position. In a military context, this capability becomes particularly valuable, as it enables precise tracking of soldiers' locations in real time. This information can improve strategic planning, improve coordination in the field, and increase overall safety during operations.

1.4.5 Augmented reality and virtual reality

The tracking of the head, arm, or hand in AR/VR headsets plays a crucial role in enhancing immersion by accurately capturing orientation and movement. This allows for a more natural and responsive virtual experience. Furthermore, as previously mentioned in the context of sports, gesture recognition enables seamless interaction with virtual objects and gaming environments, improving both user engagement and the overall realism of the simulation.

1.4.6 Smart vehicles and driver assistance systems

Driver monitoring

IMUs have also found novel applications in vehicular technologies, particularly in enhancing driver safety and road awareness. For example, they can be used for detecting drowsiness driving by analyzing steering wheel movements [32], as well as to identify a driver based on their unique steering patterns during vehicle turns [15]. In addition, IMUs contribute significantly to Advanced Driver Assistance Systems (ADAS), improving vehicle automation and safety features [29]. Beyond individual vehicle applications, these sensors also play a role in monitoring traffic and road conditions, providing valuable data for intelligent transportation systems [33].

Autonomous vehicles

To ensure smooth and reliable navigation while overcoming the individual limitations of both GPS and IMU systems, sensor fusion techniques have been developed to integrate data from both sources. This fusion method combines GPS measurements with IMU data to improve positioning accuracy, compensate for signal loss in challenging environments, and provide continuous navigation capabilities [2]. This helps self-driving cars to maintain control and track movement when GPS is unreliable.

1.5 Objectives and work plan

The main objectives of this final degree project were defined in August 2024, providing a clear schema to follow throughout its development. These objectives were as follows.

- Acquire proficiency in the C programming language and establish communication between the *ESP32-C3* microcontroller and the computer.
- Program the ICM20948 Inertial Measurement Unit (IMU) using the SparkFun Arduino library.
- Understand the theoretical foundations of orientation, particularly the use of quaternions and Tait-Bryan angles.

- Research and implement various algorithms to convert raw sensor data into interpretable orientation values. This includes techniques such as the Complementary Filter, the Kalman Filter, and Attitude and Heading Reference System (AHRS) algorithms.
- Study the calibration processes for gyroscopes, accelerometers, and magnetometers, and determine appropriate calibration parameters for the sensor.
- Implement a gait tracking algorithm to estimate position, emulating a GPS-like system without requiring network connectivity.
- Obtain detailed power consumption measurements for the magnetometer, accelerometer, and gyroscope, as well as for the different data acquisition methods evaluated. Draw conclusions to determine the most efficient method for extracting data from the IMU in terms of energy consumption.

The development of the project began in September 2024, driven by the motivation and commitment to dedicate the necessary time and effort to a project of this scope and complexity.

Throughout the months, the process has been filled with challenges, discoveries, and achievements, ultimately leading to the successful implementation and analysis of several algorithms. The timeline of the project development is outlined below:

1. **September 2024 to mid-October 2024:** The initial phase focused on configuring the sensor, installing the necessary software, evaluating different tools based on their ease of use, and establishing a basic connection between the sensor and the computer.
2. **Mid-October 2024 to November 2024:** Work was centered on programming the sensor in C to obtain and transmit raw data accurately.
3. **November 2024 to January 2025:** The project transitioned into a theoretical phase, aiming to understand the mathematical foundations of orientation and its representation.
4. **February 2025 to March 2025:** A deep exploration of various orientation algorithms was undertaken, focusing on transforming the raw sensor data into meaningful and interpretable information. Simultaneously with the ongoing research and development, the writing of this report began and continued progressively until the completion of the project.
5. **Mid-March 2025:** The study was extended to include calibration algorithms and gait tracking methodologies.
6. **April 2025:** Multiple data sets were recorded and analyzed to evaluate the strengths and limitations of each implemented algorithm. These experiments formed the basis for the conclusions presented in this report.
7. **May 2025:** During this month, experiments focused on measuring the power consumption of the IMU components were initiated. Eight different configurations were tested, including various operational modes for the gyroscope, accelerometer, and magnetometer, along with multiple data acquisition strategies. Based on these results, a conclusion was reached regarding the most energy-efficient approach for data extraction, aiming to maximize battery life.

Chapter 2

Theoretical basis

After establishing the importance of determining position and orientation, along with highlighting several relevant applications, this chapter presents the theoretical foundations for interpreting orientation in a three-dimensional space. It introduces common mathematical representations of orientation, including Tait-Bryan angles and quaternions. These models are examined from a theoretical perspective, outlining their characteristics, advantages, and limitations in preparation for subsequent practical applications.

2.1 Tait-Bryan angles

Named after Peter Guthrie Tait and George Bryan [60], the Tait-Bryan angles are three elemental rotations about each one of the principal axis of a body, x, y and z, respectively called roll (ϕ), pitch (θ) and yaw (ψ), as seen in Figure 2.1. The range for the angles ψ and ϕ is 2π radians, while for θ the range is π radians.

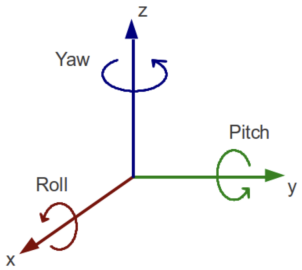

Figure 2.1: Representation of pitch, roll and yaw along the three axes [20].

A 3D rotation can be numerically represented using matrices. One common way to describe a rotation using Tait-Bryan angles is through the composition:

$$R = R_x(\phi) \cdot R_y(\theta) \cdot R_z(\psi) \tag{2.1}$$

where the matrices are:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \quad R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

They are often called Euler angles because both are two methods for representing 3D orientation using three successive rotations. But this conflicts with existing usage elsewhere, because Tait-Bryan rotations have differences with Euler angles. Euler angles involve two rotations about the same axis and one about a different axis (e.g., z-x-z), whereas Tait-Bryan angles use three distinct axes (e.g., x-y-z). Both suffer from singularities such as gimbal lock, but Tait-Bryan angles provide a more intuitive representation for many applications.

2.1.1 Gimbal lock

Gimbal lock is a mathematical problem that arises only when Euler or Tait-Bryan angles are used to represent 3D orientation.

More specifically, it is a phenomenon where a multi-axis mechanism loses one of its degrees of freedom due to specific axis alignments [62]. In a system with three gimbals operating in three dimensions, this occurs when two of the gimbal axes align, reducing the system's ability to rotate freely and restricting motion to a two-dimensional plane.

For example, it can occur when the x-axis points straight up or straight down, when the pitch is equal to $\pm 90°$, as shown in Figure 2.2. In this orientation, the roll and yaw angles will become mathematically unstable, since they are "locked" together in this orientation; both are affected by a rotation around the x-axis of the rotating body.

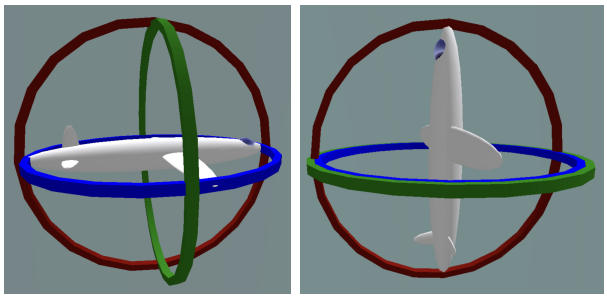

Figure 2.2: Gimbal lock [62].

Mathematically, this example can be seen taking $\theta = \frac{\pi}{2}$ in Equation (2.1). Using the identities $\cos\left(\frac{\pi}{2}\right) = 0$ and $\sin\left(\frac{\pi}{2}\right) = 1$, the rotation becomes:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying the matrices:

$$R = \begin{bmatrix} 0 & 0 & 1 \\ \sin\phi & \cos\phi & 0 \\ -\cos\phi & \sin\phi & 0 \end{bmatrix} \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ \sin\phi\cos\psi + \cos\phi\sin\psi & -\sin\phi\sin\psi + \cos\phi\cos\psi & 0 \\ -\cos\phi\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\psi + \sin\phi\cos\psi & 0 \end{bmatrix}$$

Using trigonometric identities, it can be simplified further:

$$R = \begin{bmatrix} 0 & 0 & 1 \\ \sin(\phi+\psi) & \cos(\phi+\psi) & 0 \\ -\cos(\phi+\psi) & \sin(\phi+\psi) & 0 \end{bmatrix}$$

This result shows that the net effect of this specific configuration is a rotation by $\phi+\psi$, while the rotation axis remains fixed along the Z-axis. Therefore, ϕ and ψ are indistinguishable in terms of their contribution to orientation, and changing one without the other cannot reverse this effect unless θ changes.

2.2 Quaternions

Quaternions don't have gimbal lock, so they represent a convenient alternative to the Euler or Tait-Bryan angles. They use a 4-dimensional mathematical object to represent the 3D orientation. The additional component ensures that each possible orientation is uniquely represented, and avoids the gimbal lock problem. Quaternions can be used to concatenate a series of rotations into a single representation.

In mathematics, the quaternion number system extends the complex numbers and is often denoted by $\mathbb{H}$. Quaternions provide a definition of the quotient of two vectors in a three-dimensional space [69]. They are generally represented in the following way:

$$q = (q_w, q_x, q_y, q_z) \in \mathbb{H}, \text{where } \mathbb{H} = \{a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} \mid a, b, c, d \in \mathbb{R}, \mathbf{i}, \mathbf{j}, \mathbf{k} \in \mathbb{C}\} \subset \mathbb{C}^2 \tag{2.3}$$

On the one hand, q_w represents the real part of the quaternion and is related to the angle of rotation. On the other hand, q_x, q_y, q_z represents the imaginary components, often written as $q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}$.

The norm of a quaternion $q = (q_w, q_x, q_y, q_z)$ is given by:

$$\|q\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}$$

The conjugate of a quaternion is defined as:

$$q^* = (q_w, -q_x, -q_y, -q_z)$$

A unit quaternion q, satisfying $\|q\| = 1$ and $q^{-1} = q^*$, can represent the rotation between two coordinate frames. The unit quaternion that represents a rotation by angle θ around a unit axis $\mathbf{v} = (v_x, v_y, v_z)$ is given by:

$$q = (\cos(\theta/2), v_x \sin(\theta/2), v_y \sin(\theta/2), v_z \sin(\theta/2)) \tag{2.4}$$

Continuing with the same example from the previous section, consider a rotation of $\pm 90°$ around the x-axis. In this case, the rotation axis is $\mathbf{v} = (1, 0, 0)$, and by applying the formula from Equation (2.4), the corresponding quaternion is given by $q = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0, 0)$.

The rotation matrix [10] corresponding to a unit quaternion is given by:

$$R = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & -2q_wq_z + 2q_xq_y & 2q_wq_y + 2q_xq_z \\ 2q_wq_z + 2q_xq_y & q_w^2 - q_x^2 + q_y^2 - q_z^2 & -2q_wq_x + 2q_yq_z \\ -2q_wq_y + 2q_xq_z & 2q_wq_x + 2q_yq_z & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{2.5}$$

2.2.1 Quaternions to Tait-Bryan angles

The quaternions can be converted to Tait-Bryan angles using the expression given in Equation (2.6) [10, 59]. However, as previously discussed in Section 2.1.1, Tait-Bryan angles are susceptible to the gimbal lock problem. Consequently, when quaternions are transformed into Tait-Bryan angles, this issue can reemerge. As a result, certain orientations may not be accurately represented after conversion, potentially leading to ambiguities or loss of information in the angular representation.

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}\left(2(q_w q_x + q_y q_z), 1 - 2(q_x^2 + q_y^2)\right) \\ -\frac{\pi}{2} + 2\text{atan2}\left(\sqrt{1 + 2(q_w q_y - q_x q_z)}, \sqrt{1 - 2(q_w q_y - q_x q_z)}\right) \\ \text{atan2}\left(2(q_w q_z + q_x q_y), 1 - 2(q_y^2 + q_z^2)\right) \end{bmatrix} \tag{2.6}$$

2.3 Axes convention

Axes conventions are structured systems used to define the position and orientation of coordinate axes, serving as a consistent frame of reference for spatial measurements and transformations [58].

There are two commonly used world reference frames:

- **East-North-Up (ENU)**: This convention is primarily used in geodetic and geographic applications. In this system, the x-axis points East, the y-axis points North, and the z-axis points upward.
- **North-East-Down (NED)**: This convention is widely used in aerospace and marine navigation. Here, the x-axis points North, the y-axis points East, and the z-axis points downward.

To define a consistent convention for describing the orientation (attitude) of a rigid body, it is essential to specify both the reference coordinate system and the body-fixed axes of the vehicle.

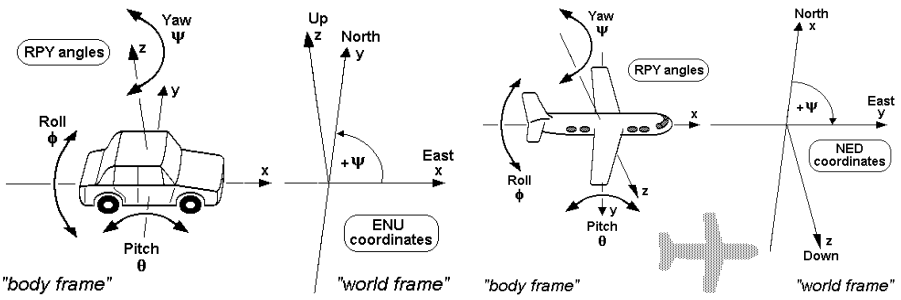

Figure 2.3: ENU convention for a vehicle [45]. NED convention for a plane [44].

In all cases, the positive x-axis of a vehicle is aligned with its forward direction of motion. However, the orientation of the y- and z-axes can vary depending on the domain of application [58]:

- For land-based vehicles, the ENU convention is typically used, with the y-axis pointing left and the z-axis pointing upward. It can be seen on the left in Figure 2.3.

- For aerial and maritime vehicles, the NED convention is generally adopted, as seen on the right in Figure 2.3, where the y-axis points right and the z-axis points downward.

2.4 Calibration

In scientific terms, calibration refers to the process of adjusting an instrument so that its output for a given sample falls within an acceptable range. This step is essential to ensure the precision and dependability of sensor measurements [31]. Sensor raw data often contains errors due to manufacturing tolerances, environmental factors, and surrounding interference. These errors can cause deviations in readings, affecting the sensor's performance.

In the case of IMU, proper calibration is essential to minimize bias, scale factor errors, and sensor drift. Each of these sensors has unique calibration challenges:

- Accelerometer calibration corrects for offsets and scale factors to ensure proper gravity-based orientation measurement.
- Gyroscope calibration compensates for drift and biases that accumulate over time.
- Magnetometer calibration addresses distortions caused by nearby ferromagnetic materials and electronic components, such as hard iron and soft iron effects.

The most common calibration techniques aim to correct the instrumentation errors such as offset, sensitivity, and misalignment. These errors are constant and specific to each sensor unit.

Offset

Offset, also known as bias or zero error, refers to the constant bias present in the sensor output when it should ideally be zero or another value, depending on the sensor. This error can result from manufacturing imperfections, temperature variations, etc.

For a gyroscope, the offset corresponds to the output value when the sensor is stationary, as no angular velocity should be detected. In the case of an accelerometer or magnetometer, the offset represents the difference between the actual measured center and the expected reference value.

Offset calibration involves determining this constant bias, which can be modeled as Equation (2.7), and subtracting it from all future measurements to center the output around the true zero value as seen in Figure 2.4. This constant actually consists of three separate components, one offset error per axis, since each axis operates independently.

$$\mathbf{b}_{so} = \begin{bmatrix} b_{so_x} \\ b_{so_y} \\ b_{so_z} \end{bmatrix} \tag{2.7}$$

Sensitivity

Sensitivity calibration corrects errors in the sensor scale factor. Ideally, a sensor should provide a linear response to a physical input with a known gain. However, due to imperfections in the sensing elements, the actual output may differ from the expected value. Sensitivity errors can cause the sensor to overestimate or underestimate the magnitude of the measured quantity as shown in Figure 2.5.

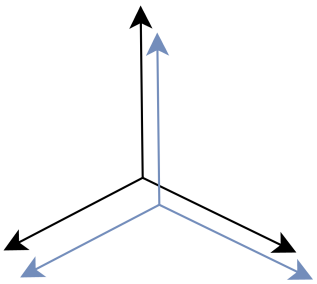

Figure 2.4: Ideal sensor axes (black) compared to actual measured axes with offset error (blue).

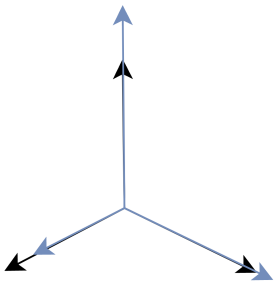

Figure 2.5: Ideal sensor axes (black) compared to actual measured axes with scale error (blue).

Calibration in this context involves determining the actual scale factors for each axis, as seen in the diagonal matrix **S** in Equation (2.8), and applying correction multipliers to bring the output in line with the true physical values.

$$\mathbf{S} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \tag{2.8}$$

Misalignment

Misalignment calibration addresses the non-orthogonality of the sensor axes. In a perfectly aligned sensor, the measurement axes should be exactly orthogonal to each other. However, due to mechanical imperfections or assembly errors, there may be slight angular deviations between the axes as seen in Figure 2.6. This results in cross-axis sensitivity, where movement along one axis influences readings on another.

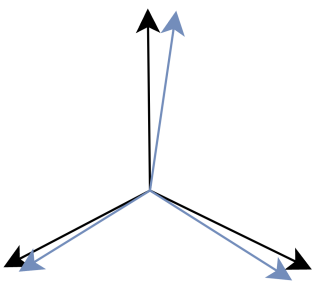

Figure 2.6: Ideal sensor axes (black) compared to actual measured axes with misalignment error (blue).

Misalignment calibration uses a transformation matrix, represented in Equation (2.9), to correct these deviations, aligning the sensor's frame with the reference coordinate frame and

ensuring accurate multi-axis measurements.

$$\mathbf{N} = \begin{bmatrix} n_{1x} & n_{1y} & n_{1z} \\ n_{2x} & n_{2y} & n_{2z} \\ n_{3x} & n_{3y} & n_{3z} \end{bmatrix} \tag{2.9}$$

2.4.1 Quadrics

In a three-dimensional Euclidean space, quadrics are surfaces described by second-degree polynomials in x, y, and z; examples include spheres, ellipsoids, and paraboloids [42]. The general implicit form of a quadric surface S is given by Equation (2.10), where

$$a, b, c, d, f, g, h, p, q, r \in \mathbb{R},$$

and at least one of a, b, c is nonzero [68].

$$S : ax^2 + by^2 + cz^2 + 2fyz + 2gxz + 2hxy + px + qy + rz + d = 0 \tag{2.10}$$

This equation can also be expressed in matrix form:

$$S : \mathbf{x}^t\mathbf{M}\mathbf{x} + \mathbf{x}^t\mathbf{n} + d = 0 \tag{2.11}$$

where

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} a & h & g \\ h & b & f \\ g & f & c \end{bmatrix} \qquad \mathbf{n} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.12}$$

The combined effects of sensitivity and misalignment errors, mentioned in the previous section, can be represented geometrically as measurements that form an ellipsoid rather than a perfect sphere. Calibration involves transforming this ellipsoid into a sphere. By adjusting the radius, scale factor errors (sensitivity) are corrected, while aligning the ellipsoid with the sphere also compensates for axis misalignment. Offset calibration is applied when the center of the sphere is adjusted to match its expected position. This transformation process is illustrated in Figure 2.7.

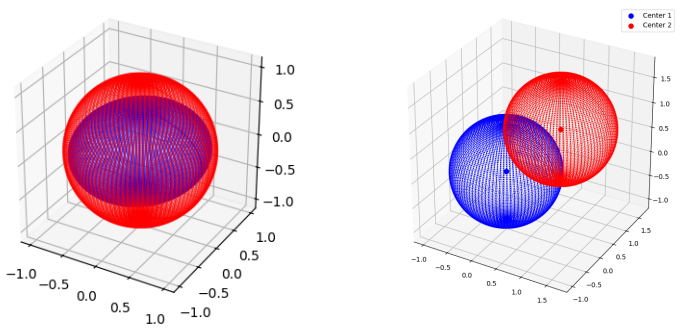

Figure 2.7: Transformation used in calibration to correct errors. Left, misalignment and sensitivity. Right, offset.

2.4.2 Magnetic calibration

Assuming the sensor is in an ideal, magnetically undisturbed environment and has a perfect 3-axis magnetometer, a magnetic reading $\mathbf{h}$ taken at any arbitrary orientation can be described as:

$$\mathbf{h} = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\varphi)\mathbf{h}_0 \tag{2.13}$$

Here, $\mathbf{h}_0$ is the local magnetic field vector defined in Equation (1.1), and $\mathbf{R}_x(\phi)$, $\mathbf{R}_y(\theta)$, and $\mathbf{R}_z(\varphi)$ are the rotation matrices around the x, y, and z axes, defined in Equation (2.2). Since these matrices are orthogonal (i.e., their transposes are equal to their inverses), it follows that the locus of all possible readings $\mathbf{h}$ forms a sphere of radius $\mathcal{F}$:

$$\begin{aligned}
\mathbf{h}^t\mathbf{h} &= (\mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\varphi)\mathbf{h}_0)^t(\mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\varphi)\mathbf{h}_0) \\
&= \mathbf{h}_0^t\mathbf{R}_z(\varphi)^t\mathbf{R}_y(\theta)^t\mathbf{R}_x(\phi)^t\mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\varphi)\mathbf{h}_0 \\
&= \mathbf{h}_0^t\mathbf{R}_z(\varphi)^t\mathbf{R}_y(\theta)^t\mathbf{R}_y(\theta)\mathbf{R}_z(\varphi)\mathbf{h}_0 \\
&= \mathbf{h}_0^t\mathbf{R}_z(\varphi)^t\mathbf{R}_z(\varphi)\mathbf{h}_0 \\
&= \mathbf{h}_0^t\mathbf{h}_0 \\
&= |\mathbf{h}_0^t||\mathbf{h}_0| \cdot \cos(0) \\
&= \mathcal{F}^2
\end{aligned} \tag{2.14}$$

In real-world conditions, both the sensor and environment introduce imperfections. Apart from the instrumentation errors mentioned before, magnetometers also introduce errors due to magnetometer interferences [51], which are caused by surrounding materials, especially ferromagnetic ones. These are divided into hard iron and soft iron and are based on the ellipsoid fitting technique.

Hard iron

Hard iron is caused by permanent magnets or magnetized materials near the sensor, modeled as a bias as Equation (2.15). When the magnetic material is rigidly fixed to the same reference frame as the sensor, this type of hard iron distortion introduces a constant bias in the sensor's measurements [52].

$$\mathbf{b}_{hi} = \begin{bmatrix} b_{hi_x} \\ b_{hi_y} \\ b_{hi_z} \end{bmatrix} \tag{2.15}$$

Soft iron

Soft iron is induced magnetism due to nearby metallic materials that alter the magnetic field, represented as Equation (2.16).

$$\mathbf{A}_{si} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{2.16}$$

Measurement model

In this section, the ellipsoid fitting technique will be explained. Combining all the above distortions, the instrumentation errors and the magnetic interferences, the measured magnetic field $\mathbf{h}_m$ can be modeled as:

$$\mathbf{h}_m = \mathbf{SN}(\mathbf{A}_{si}\mathbf{h} + \mathbf{b}_{hi}) + \mathbf{b}_{so} \tag{2.17}$$

By defining a single matrix for all scale-related transformations and a single bias vector, it can be expressed as follows:

$$\begin{aligned} \mathbf{A} &= \mathbf{SNA}_{si} \\ \mathbf{b} &= \mathbf{SNb}_{hi} + \mathbf{b}_{so} \\ \mathbf{h}_m &= \mathbf{Ah} + \mathbf{b} \end{aligned} \tag{2.18}$$

This transformation causes the measurements $\mathbf{h}_m$ to lie on an ellipsoid. Solving for the ideal field vector $\mathbf{h}$:

$$\mathbf{h} = \mathbf{A}^{-1}(\mathbf{h}_m + \mathbf{b}) \tag{2.19}$$

Substituting into the quadric form gives:

$$\mathbf{h}_m^t\mathbf{M}\mathbf{h}_m + \mathbf{h}_m^t\mathbf{n} + d = 0 \tag{2.20}$$

with:

$$\begin{aligned} \mathbf{M} &= \mathbf{A}^{-t}\mathbf{A}^{-1} \\ \mathbf{n} &= -2\mathbf{A}^{-t}\mathbf{A}^{-1}\mathbf{b} \\ d &= \mathbf{b}^t\mathbf{A}^{-t}\mathbf{A}^{-1}\mathbf{b} - \mathcal{F}^2 \end{aligned} \tag{2.21}$$

Given estimated values $\hat{\mathbf{M}}, \hat{\mathbf{n}}, \hat{d}$, the calibration matrix $\hat{\mathbf{A}}$ and bias vector $\hat{\mathbf{b}}$ can be recovered as:

$$\begin{aligned} \hat{\mathbf{b}} &= -\hat{\mathbf{M}}^{-1}\hat{\mathbf{n}} \\ \hat{\mathbf{A}}^{-1} &= \frac{\mathcal{F}}{\sqrt{\hat{\mathbf{n}}^t\hat{\mathbf{M}}^{-1}\hat{\mathbf{n}} - \hat{d}}}\hat{\mathbf{M}}^{1/2} \end{aligned} \tag{2.22}$$

2.4.3 Accelerometer calibration

There are mainly three elements that are important in accelerometer calibration: accelerometer offset, accelerometer misalignment, and accelerometer sensitivity.

Accelerometer offset and accelerometer misalignment can be calculated in the same way as hard iron and soft iron, but using a radius ellipsoid $\mathcal{F} = 9.81$, which will calibrate the measurements to be in m/s^2.

The accelerometer sensitivity is typically calculated using automated machines to ensure precision and repeatability [49]. These machines follow standardized procedures such as controlled rotation in a precision gimbal where the accelerometer is mounted on a high-precision calibration table that rotates at known angles; vibration testing on a shaker table where the

accelerometer is subjected to sinusoidal oscillations at different frequencies and amplitudes; and temperature compensation where the sensor is exposed to various temperature conditions in an environmental chamber.

2.4.4 Gyroscope calibration

Gyroscope calibration involves the same three components as the accelerometer, which are: gyroscope offset, gyroscope misalignment, and gyroscope sensitivity.

The gyroscope offset is determined differently from the accelerometer offset. Instead of using an ellipsoid fit algorithm, it is calculated as the arithmetic mean of the sensor readings when it is stationary. This provides a baseline correction to compensate for any inherent bias in the measurements.

However, gyroscope sensitivity and misalignment present the same challenge as with accelerometer calibration. These parameters require specialized equipment to be accurately determined, as they depend on precise controlled movements and external references that are difficult to achieve manually.

2.5 The need for postprocessing in orientation estimation

2.5.1 Filtering

Raw sensor outputs such as angular velocity (gyroscopes), linear acceleration (accelerometers), and magnetic field strength (magnetometers) present inherent limitations that must be addressed through filtering techniques to ensure reliable orientation estimation.

The accelerometer is a crucial sensor for estimating orientation because it can measure the direction of gravity. However, accelerometers are highly susceptible to noise, particularly in dynamic environments where external forces, sudden movements, or vibrations affect their readings [24]. To mitigate this issue, a low-pass filter is typically applied, which smooths out rapid variations but at the cost of introducing lag in the response. Furthermore, accelerometers cannot distinguish between gravitational acceleration and external translational forces, making them unreliable during rapid movements.

The gyroscope, on the contrary, is much more stable and less affected by environmental conditions. It provides precise measurements of angular velocity, which can be integrated over time to estimate orientation [24]. However, this integration process leads to drift, where small errors accumulate over time, resulting in significant deviations from the actual orientation.

The magnetometer provides the sensor's absolute orientation relative to geographic North, making it useful for determining heading. However, it is highly susceptible to interference from nearby magnetic sources, such as permanent magnets, ferromagnetic materials, or even electronic devices. These disturbances can significantly distort the readings, leading to inaccurate orientation data.

2.5.2 Sensor Fusion

Given the previous limitations, a filtering approach that combines accelerometer and gyroscope data is necessary. Since gyroscopes provide smooth, high-frequency data but drift over time, and accelerometers provide absolute orientation but are noisy, a fusion algorithm can leverage the strengths of both. By incorporating a small portion of the accelerometer's calculated

orientation into the gyroscope's estimate, it is possible to correct the drift of the gyroscope while maintaining fast and smooth updates.

In addition to combining gyroscope and accelerometer data, another variant includes the magnetometer in the fusion process. This integration is particularly beneficial as it enables not only relative orientation estimation but also absolute directional awareness. For example, it allows the system to determine that the movement occurred towards the East and then towards the West, rather than just indicating a generic right-left motion.

Chapter 3

Implementation

After introducing the key concepts of orientation, calibration, and sensor fusion in Chapter 2, this chapter focuses on the implementation of various algorithms used to estimate the Tait-Bryan angles by fusing raw data from different sensors. In addition, it describes the procedures applied for sensor calibration to improve measurement accuracy.

The chapter also presents the hardware used to collect the raw sensor data, detailing the setup and data acquisition process. Furthermore, it explains the algorithms implemented for gait tracking based on the processed data.

Following this, in Chapter 4, these implemented algorithms will be applied to different real-world scenarios. Their performance will be evaluated to demonstrate their strengths, limitations, and overall practicability in motion and orientation estimation tasks.

3.1 DAURIAN board

For the implementation, the hardware setup shown in Figure 3.1 was utilized. The board includes several key components: an ESP32-C3 microcontroller (highlighted in orange), integrated within the Mini-1 module; a status LED (in yellow); an IMU sensor located at the top of the board and a voltage level translator positioned above it (both in pink); voltage regulators supplying 1.8 V and 3.3 V (in light blue); a power management integrated circuit (PMIC) and a battery connector (both in dark blue), with the PMIC located at the top and the battery connector at the bottom. In addition, a USB interface (in green) is provided for communication and power, along with a QWIIC connector (in purple) that enables external I^2C sensor integration.

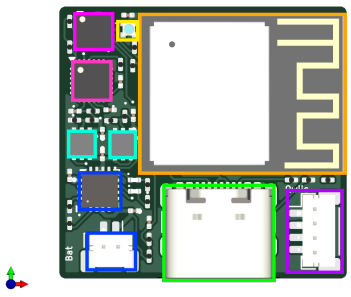

Figure 3.1: Design and components of DAURIAN board.

Communication between the IMU and the micro controller is established through the board, allowing the processor to retrieve sensor data for subsequent testing and analysis. The data

acquisition was programmed using the Arduino development environment [57].

3.2 ICM-20948

The IMU, identified as the ICM-20948, is located on the board and highlighted in pink in Figure 3.1.

The ICM-20948 is presented as the world's lowest power 9-axis motion-tracking device, designed for integration into smartphones, tablets, wearable sensors, and IoT applications [50]. It combines a 3-axis gyroscope, a 3-axis accelerometer, a 3-axis compass shown in Figure 3.2 and in Figure 3.3 within a compact 3 x 3 x 1 mm 24-pin QFN package.

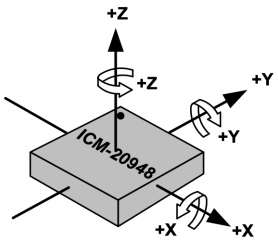

Figure 3.2: Orientation of Axes of Sensitivity and Polarity of Rotation [50].

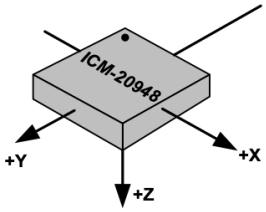

Figure 3.3: Orientation of Axes of Sensitivity for Magnetometer [50].

A key feature is its integrated Digital Motion Processor (DMP), which offloads the computational burden of motion processing algorithms from the host processor, resulting in improved system power performance. One of the main goals of this project is to compare several algorithms that compute the Tait-Bryan angles from raw sensor data with the angles provided directly by the DMP, which performs this computation internally as part of the IMU.

The ICM-20948 also provides an auxiliary I2C interface to facilitate connections with external sensors. It incorporates on-chip 16-bit Analog-to-Digital Converters (ADCs), programmable digital filters, an embedded temperature sensor, and programmable interrupts. The device operates within a voltage range down to 1.71V. Communication ports include I2C and high-speed SPI, capable of operating at up to 7 MHz [50].

To interface with the ICM-20948, this project utilizes the SparkFun ICM-20948 Arduino Library [16], a library developed by SparkFun Electronics, which provides an easy-to-use API to configure and retrieve data from the sensor. The communication between the micro-controller and the sensor is handled using the I2C (Inter-Integrated Circuit) protocol, which allows multiple devices to communicate over a shared two-wire bus.

3.2.1 I2C communication and sensor initialization

The I2C protocol is a two-wire serial communication standard that consists of a data line (SDA) and a clock line (SCL). It is widely used to connect low-speed peripheral devices to microcontrollers, such as sensors.

To set up I2C communication, the SDA and SCL pins are defined and the clock speed is set to 400 kHz for fast data transfer. A 1.8V voltage regulator is also enabled since the ICM-20948 operates at a lower voltage level than many micro-controllers.

When the sensor is initialized, the micro-controller repeatedly attempts to establish communication. If initialization fails, the system retries until a successful connection is made, printing status messages to the serial monitor.

3.2.2 Reading and Processing Sensor Data

Three different methods were implemented for acquiring sensor data. Depending on the specific application, one approach was selected over the others. However, all methods provide the same core information: each data frame includes measurements of linear acceleration, angular velocity (gyroscope), and the magnetic field, all recorded along the x, y, and z axes. In Chapter 5, a study about the power consumption of each method will be discussed.

Digital Motion Processor (DMP)

The Digital Motion Processor (DMP) is an onboard hardware engine that performs real-time sensor fusion and internal processing of accelerometer, gyroscope, and magnetometer data. This offloads computational effort from the microcontroller by directly outputting quaternions, Euler angles, and calibrated sensor readings.

The DMP supports continuous background calibration, ensuring consistently accurate outputs from both raw and fused sensor data. This contributes to enhanced performance and reliability across sensor-based applications over the device's lifetime.

Once activated, the DMP enables multiple operating modes [50], such as raw accelerometer data, raw gyroscope data, and uncalibrated magnetometer data.

The ICM-20948 also features a 512-byte FIFO (First-In, First-Out) buffer that temporarily stores data until it is read by the microcontroller. This mechanism preserves the correct data sequence and prevents data loss during periods when the processor is occupied. After reading, the FIFO is reset to maintain data integrity.

The maximum sampling frequency for the DMP is 60 Hz.

Polling

Another method for retrieving sensor data involves bypassing the DMP and directly accessing the raw values from the sensors. This approach allows for a higher data acquisition rate, reaching up to 400 Hz. Using the provided library, the implementation simply checks if new data is available and retrieves it if so.

Interrupts

This method builds on the polling approach but uses hardware interrupts instead of continuous checking. When new sensor data is available, an interrupt signals the microcontroller,

reducing unnecessary processor activity.

The interrupt system is configured through the Interrupt Configuration Register. Adjustable settings include interrupt pin behavior, latching/clearing methods, and specific trigger events.

Measurement Units

Sensor measurements vary slightly depending on the extraction method:

- **DMP (raw units):**
  - Accelerometer: raw integers, converted to g by dividing by 8192.
  - Gyroscope: raw integers, converted to degrees per second (°/s) by dividing by 16.4.
  - Magnetometer: raw integers, converted to μT by multiplying by 0.15.
- **Polling and Interrupt modes:**
  - Accelerometer: values in milligravity (mg).
  - Gyroscope: degrees per second (°/s).
  - Magnetometer: microteslas (μT).

3.3 Calibration

As described in Section 2.4, calibration procedures were implemented for the gyroscope, accelerometer, and magnetometer to obtain their respective correction parameters.

3.3.1 Gyroscope

For gyroscope calibration, only the offset needs to be estimated, as explained in Section 2.4.4. Other calibration parameters require specialized equipment and were not considered in this implementation.

To compute the offset, sensor data was collected while the device remained completely stationary. Since the gyroscope should ideally read zero under these conditions, the bias was estimated as the average of all readings. This was done using a simple Python script with the `pandas` library to compute the mean across each axis.

3.3.2 Magnetometer

To correct for both hard iron and soft iron distortions, a Python script developed by S. James Remington (`jremington` on GitHub) was used [28]. This implementation follows the ellipsoid fitting method outlined in Section 2.4.2. The only required configuration parameter is $\mathcal{F}$, which, based on the DMP units, was set to 250.

It's worth noting that the magnitude of the magnetometer vector is not critical; only the direction matters. Therefore, the value of $\mathcal{F}$ is not essential as long as all measurements are in consistent units.

To ensure a high-quality dataset for ellipsoid fitting, 3000 samples were collected. The data was acquired while rotating the sensor steadily and uniformly across all three axes. For each axis, both positive and negative directions were covered, and the sensor was rotated within a range of $[-90°, 90°]$ for each position to capture a comprehensive distribution.

3.3.3 Accelerometer

The accelerometer calibration process is similar to that of the magnetometer. To determine the sensitivity and offset, the key is to set $\mathcal{F} = 8192$, which corresponds to the raw data scaling used by the DMP for accelerometer measurements.

3.4 Transformation of raw sensor data into Tait-Bryan angles

In motion tracking and inertial navigation, raw sensor data from IMUs must be processed to obtain meaningful orientation information. The accelerometer, gyroscope, and magnetometer provide raw measurements that require techniques to estimate the sensor's orientation in terms of Tait-Bryan angles (roll, pitch, and yaw).

This section explores different methods used to transform the raw sensor data into Tait-Bryan angles, including: the Complementary Filter, which blends accelerometer and gyroscope data using a simple weighting factor; the Kalman Filter, a probabilistic approach that optimally estimates orientation while reducing noise; and the `Fusion` library, which implements sensor fusion algorithms to refine orientation estimation.

Each method has its own strengths and trade-offs in terms of computational efficiency, noise robustness, and real-time applicability. The following sections describe their implementations.

3.4.1 Complementary Filter

The complementary filter is one of the simplest and most efficient alternatives. It relies on a weighted combination of gyroscope and accelerometer data, providing a fast and reliable estimation of orientation with minimal computational overhead [24].

Let $\varphi = (\varphi_{\text{pitch}}, \varphi_{\text{roll}})$ represent the angles estimated from the accelerometer, and let $\sigma = (\sigma_{\text{pitch}}, \sigma_{\text{roll}})$ denote the angular velocity-derived estimates from the gyroscope, after applying bias correction. Additionally, let $\beta_n = (\phi_n, \theta_n)$ be the estimated roll and pitch angles at time step n. The complementary filter is then defined by:

$$\beta_n = \alpha\varphi + (1 - \alpha)(\beta_{n-1} + \sigma\Delta t) \tag{3.1}$$

Here, $\alpha \in [0, 1]$ is a weighting factor that balances the contribution of accelerometer and gyroscope data, and Δt is the time interval between successive measurements.

The accelerometer-based estimates of roll and pitch are computed as follows:

$$\varphi_{\text{pitch}} = \arctan\left(\frac{-\text{Acc}_x}{\sqrt{\text{Acc}_y^2 + \text{Acc}_z^2}}\right) \tag{3.2}$$

$$\varphi_{\text{roll}} = \arctan\left(\frac{\text{Acc}_y}{\sqrt{\text{Acc}_x^2 + \text{Acc}_z^2}}\right) \tag{3.3}$$

The angular rate of change in roll and pitch is computed taking into account cross-axis dependencies:

$$\sigma_{pitch} = \cos(\phi_{n-1})\text{Gyr}_y - \sin(\phi_{n-1})\text{Gyr}_z \tag{3.4}$$

$$\sigma_{roll} = \text{Gyr}_x + \tan(\theta_{n-1}) \left(\sin(\phi_{n-1}) \text{Gyr}_y + \cos(\phi_{n-1}) \text{Gyr}_z \right) \tag{3.5}$$

It was implemented in Python, using raw data measurements from the accelerometer and gyroscope, excluding the magnetometer. First, the angles are computed using the accelerometer, following Equation (3.2) and Equation (3.3). To reduce noise and improve accuracy, the gyroscope was first calibrated by computing and subtracting the bias for each axis, as described in Section 3.3.1. Then, following Equation (3.4) and Equation (3.5), the angular rates of change are calculated. These rates are integrated over time to update the gyroscope-based orientation estimate. This process was repeated for each time step during the data extraction.

Subsequently, the complementary filter is applied as shown in Equation (3.1), setting a fixed α for all iterations. The time is then updated, and another iteration is performed. A simplified representation of the filter workflow is shown in Figure 3.4.

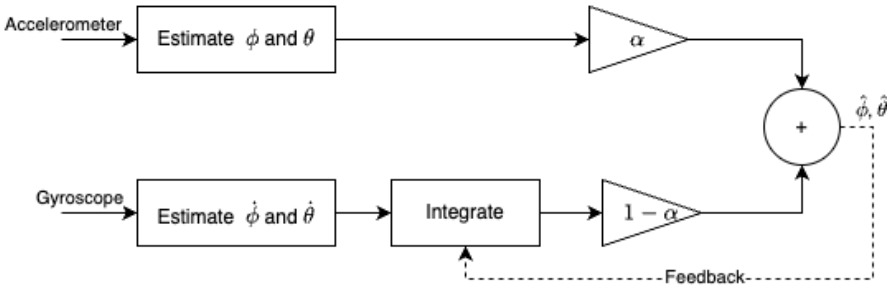

Figure 3.4: Flow of Complementary Filter.

A value of $\alpha = 0.02$ was chosen for all iterations, prioritizing the gyroscope data for short-term changes while using the accelerometer to correct long-term drift. Further evaluation of different α values is discussed in Section 4.2.

3.4.2 Kalman Filter

The Kalman filter is a sophisticated algorithm that provides an optimal estimation of orientation by continuously updating state variables based on probabilistic models. However, it is computationally expensive and may be excessive for applications with limited processing power.

The Kalman filter follows the form of the classic observer where K represents the Kalman gain. A state in a Kalman filter is the vector to estimate, in this case, the state represents the Tait-Bryan angles and the bias of the gyro. As shown in [17], the filter takes as input the current state and predicts the future state. Let t represent the time and $\mathbf{x}_t$ the state at time t.

$$\mathbf{x}_t = \begin{bmatrix} \phi_t \\ \theta_t \\ \psi_t \\ \mathbf{b}_t \end{bmatrix} \tag{3.6}$$

The first step is to obtain the gyroscope and accelerometer measurements, which will be represented as w_t and a_t, respectively. The next predicted state is then computed using system dynamics. If $\hat{}$ represents estimations, μ_t the mean at time t, Σ_t the covariance at time t, $\mathbf{Q}_t$ the noise matrix that models the noise of the system dynamics models at time t, and $\mathbf{A}_{t+1}$ represents the system model that models how the state changes from t to $t+1$.

$$\hat{\mu}_{t+1} = \mathbf{A}_{t+1}\mu_t + \mathbf{B}_{t+1}\mathbf{u}_{t+1}\hat{\Sigma}_{t+1} = \mathbf{A}_{t+1}\hat{\Sigma}_t\mathbf{A}_{t+1}^T + \mathbf{Q}_{t+1} \tag{3.7}$$

$$\mathbf{A}_{t+1} = \begin{bmatrix} 1 & 0 & 0 & -\Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & -\Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & -\Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

In Equation (3.8), Δt represents the time elapsed between t and $t+1$. In addition, $\mathbf{u}_{t+1}$ represents the input vector and is given by:

$$\mathbf{u}_{t+1} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{R}^{-1} w_t = \begin{bmatrix} \cos(\theta) & 0 & -\cos(\phi)\sin(\theta) \\ 0 & 1 & \sin(\phi) \\ \sin(\theta) & 0 & \cos(\phi)cos(\theta) \end{bmatrix}^{-1} w_t \tag{3.9}$$

$$\mathbf{B}_{t+1} = \begin{bmatrix} \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.10}$$

Here, $\mathbf{B}_{t+1}$ denotes the mapping of the input vector to the state vector.

The third and last step is to compute the attitude using the measurements and use them to obtain the corrected state.

$$\begin{aligned} \mathbf{K}_{t+1} &= \hat{\Sigma}_{t+1}\mathbf{C}^T(\mathbf{C}\hat{\Sigma}_{t+1}\mathbf{C}^T + \mathbf{R})^{-1} \\ \mu_{t+1} &= \hat{\mu}_{t+1} + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{C}\hat{\mu}_{t+1}) \\ \Sigma_{t+1} &= \hat{\Sigma}_{t+1} - \mathbf{K}_{t+1}\mathbf{C}\hat{\Sigma}_{t+1} \end{aligned} \tag{3.11}$$

In here, $\mathbf{z}_{t+1}$ represents the observable state of the sensor. Then the accelerometer is used to obtain the angles as follows:

$$\phi = \arctan(\frac{-acc_x}{\sqrt{acc_y^2 + acc_z^2}}) \tag{3.12}$$

$$\theta = \arctan(\frac{acc_y}{\sqrt{acc_x^2 + acc_z^2}}) \tag{3.13}$$

Here, $\mathbf{C}$ denotes the mapping from observed state to full state and is given by:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.14}$$

Note that the value of ψ is not calculated from the accelerometer readings as it is generally inaccurate.

This algorithm was also implemented using Python and following the mentioned equations.

3.4.3 AHRS - Sensor Fusion algorithm

`Fusion` [71] is a C-based sensor fusion library, also available as the Python package `imufusion`, specifically designed for IMUs and optimized for use in embedded systems.

At its core is an Attitude and Heading Reference System (AHRS) algorithm, which combines data from gyroscopes, accelerometers, and magnetometers to provide a single estimate of orientation relative to the Earth. The algorithm is based on the revised AHRS model presented in Chapter 7 of Madgwick's PhD thesis [34].

The AHRS algorithm calculates orientation by integrating gyroscope measurements and adding a feedback term. This feedback term is proportional to the error between the current orientation estimate and the one inferred from accelerometer and magnetometer data, scaled by a gain factor. Effectively, this forms a complementary filter: high-pass filtering the gyroscope measurements and low-pass filtering the accelerometer and magnetometer data. The cutoff frequency of this filter is controlled by the gain:

- A low gain gives more weight to the gyroscope, which can lead to drift over time.
- A high gain increases reliance on accelerometer and magnetometer data, which may introduce errors due to external accelerations or magnetic disturbances.
- A gain of zero disables feedback entirely, relying solely on gyroscope integration.

This revised version improves on the quaternion-based complementary filter originally proposed by Mahony [21, 35]. Mahony's algorithm assumes a fixed magnetic inclination, limiting its use in dynamic environments. Additionally, its use of integral feedback for gyroscope bias compensation can lead to issues like integral wind-up. Madgwick's revised approach introduces several modifications to address these limitations and enhance performance [34].

Algorithm Settings

The parameters used in this implementation are shown in Table 3.1.

Parameter	Value	Explanation
`convention`	FusionConventionNwu (North-West-Up)	Specifies the Earth axes convention .
`gain`	0.5	Controls the balance between gyroscope integration and sensor correction. A value of 0 disables initialization and rejection features.
`gyroscopeRange`	2000	Maximum angular rate (in degrees per second) measurable. Values above 98% of this range trigger angular rate recovery.
`accelerationRejection`	10	Threshold (in degrees) for enabling acceleration rejection. Set to 0 to disable.
`magneticRejection`	10	Threshold (in degrees) for enabling magnetic rejection. Set to 0 to disable.
`recoveryTriggerPeriod`	5 seconds	Period (in samples) after which recovery is triggered for both acceleration and magnetic rejection. A value of 0 disables this feature.

Table 3.1: Configuration parameters for the Fusion algorithm.

The algorithm can operate in two modes: one using only gyroscope and accelerometer data, and another that additionally incorporates the magnetometer. The magnetometer-enabled version yields absolute orientation estimates and is more effective at mitigating long-term drift.

The results of both configurations, including a comparative evaluation across several data sets, are analyzed in Section 4.2.

3.4.4 DMP quaternions to angles using panda3d

Instead of obtaining raw sensor data and computing the Tait-Bryan angles externally, as done in the previous sections, this approach uses the quaternion values directly provided by the sensor, using DMP. These quaternions are then converted to Tait-Bryan angles in Python using the `panda3d.core` library.

`Panda3d` offers the `LQuaternionf` structure to represent unit quaternions, along with the `getHpr` method, which returns the corresponding Tait-Bryan angles.

This method of obtaining the Tait-Bryan angles will be used as the reference standard for evaluating the accuracy of the previously mentioned approaches. Since these angles are provided directly by the sensor's internal processing, they are considered to be reliable and will serve as the ground truth in the comparison.

3.5 Gait tracking

As discussed in Section 1.4.2, gait tracking is one of the key applications of IMUs and forms one of the main focus of this work. It serves as one of the primary frameworks for evaluating the performance and utility of IMUs, while also deepening our understanding of their behavior in practical scenarios.

An implementation example can be found in [72], where a Python-based algorithm tracks the position of an IMU attached to a subject's foot during walking. The algorithm utilizes `Fusion`, the C-based sensor fusion library described in Section 3.4.3, to transform gyroscope and accelerometer data into an acceleration estimate in the Earth coordinate frame. This acceleration is then integrated to obtain velocity. A zero-velocity detection algorithm is used to correct for drift, and the resulting velocity is further integrated to estimate position, as illustrated in Figure 3.5.

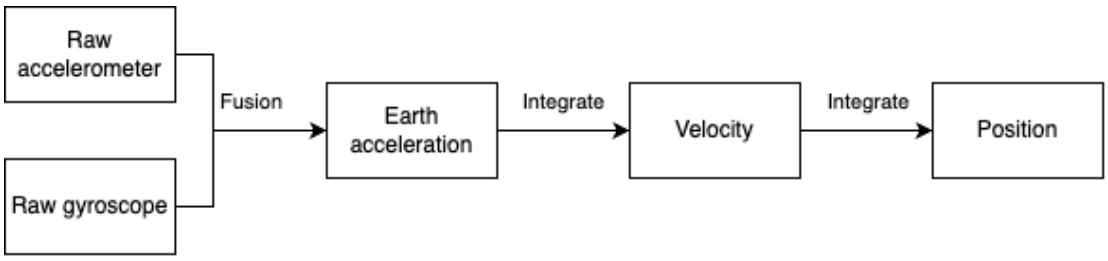

Figure 3.5: Flow diagram of the gait tracking algorithm without magnetometer input.

However, the version from [72] does not include calibration for the gyroscope and accelerometer. To address this limitation, a modified version was developed that applies proper sensor calibration prior to data processing.

Additionally, since the IMU used in this work also provides raw magnetometer readings, an enhanced version of the algorithm was implemented. This version incorporates calibrated magnetometer data, building on the previously calibrated sensor pipeline. The updated process

is shown in Figure 3.6.

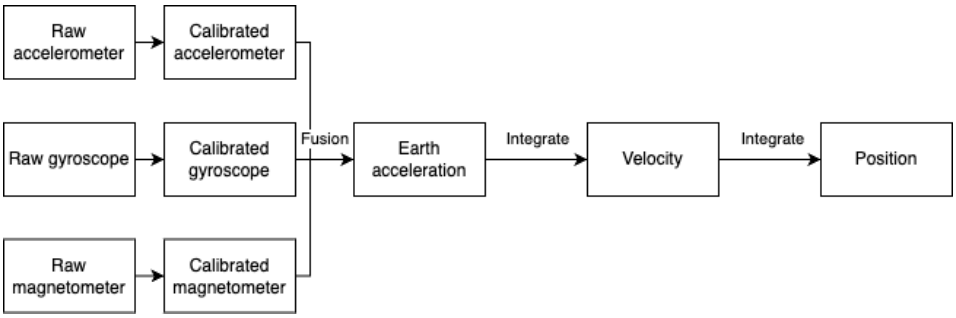

Figure 3.6: Flow diagram of the gait tracking algorithm using magnetometer and calibrated sensors.

Finally, a fourth, hybrid version of the algorithm was developed. This version combines the calibrated accelerometer and gyroscope-based pipeline with the magnetometer-enhanced approach. It selectively uses magnetometer data only when no external magnetic interference is detected, thereby improving robustness in magnetically noisy environments.

A detailed evaluation and comparison of the results from these different implementations will be presented in Section 4.3.

Chapter 4

Evaluation

After the implementation basis in the previous chapter, the results are discussed in this chapter. To have a better understanding of the results, graphics will be displayed where the following algorithms will be shown with the colors shown in Figure 4.1.

- **Complementary Filter:** This corresponds to the Complementary Filter algorithm.
- **Euler Direct:** This refers to the algorithm where quaternions are obtained directly from the IMU's internal motion processor, and then converted to Tait-Bryan angles. It is important to note that this algorithm will serve as the reference for evaluating the performance of the other algorithms.
- **Kalman Filter:** This refers to the Kalman Filter algorithm.
- **Simple Fusion:** This corresponds to the fusion algorithm that uses only the accelerometer and gyroscope, without a magnetometer.
- **Advanced Fusion:** This is the same as Simple Fusion but includes a magnetometer.
- **Advanced Fusion with Calibration:** This represents the Advanced Fusion algorithm previously mentioned but with additional calibration.

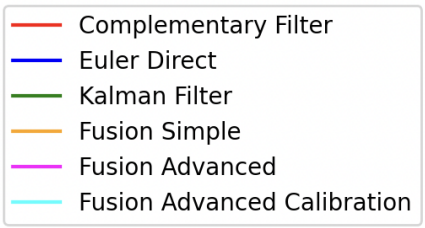

Figure 4.1: Legend of different algorithms in the graphics.

4.1 Calibration

All data sets required for sensor calibration were recorded as detailed in Section 3.3. These data sets serve as the basis for evaluating the impact of different calibration procedures on sensor accuracy and performance.

In the following section, a comprehensive analysis will be presented regarding the effects of applying or omitting individual calibration steps, including gyroscope calibration, magnetometer

calibration, and accelerometer calibration. The results will be examined to determine how each calibration process influences the precision and reliability of the sensor measurements, highlighting potential improvements and limitations introduced by each method.

All comparisons will be made using Fusion, which means using **Fusion Advanced** and **Fusion Advanced With Calibration**.

4.1.1 Gyroscope

After calculating the gyroscope offset, the following values were obtained:

$$\text{gyroscopeOffset} = [-0.672\overline{56097}, -0.0\overline{39837}, -0.465\overline{670731}] \tag{4.1}$$

These results indicate that the offset is more pronounced along the x-axis. However, the values are relatively small, meaning that they do not cause significant differences in the calibration process.

To analyze the impact of gyroscope calibration, a comparison was performed using the Fusion algorithm with and without calibration in a stationary scenario, where gyroscope calibration is expected to have the most noticeable effect. The results are illustrated in Figure 4.2.

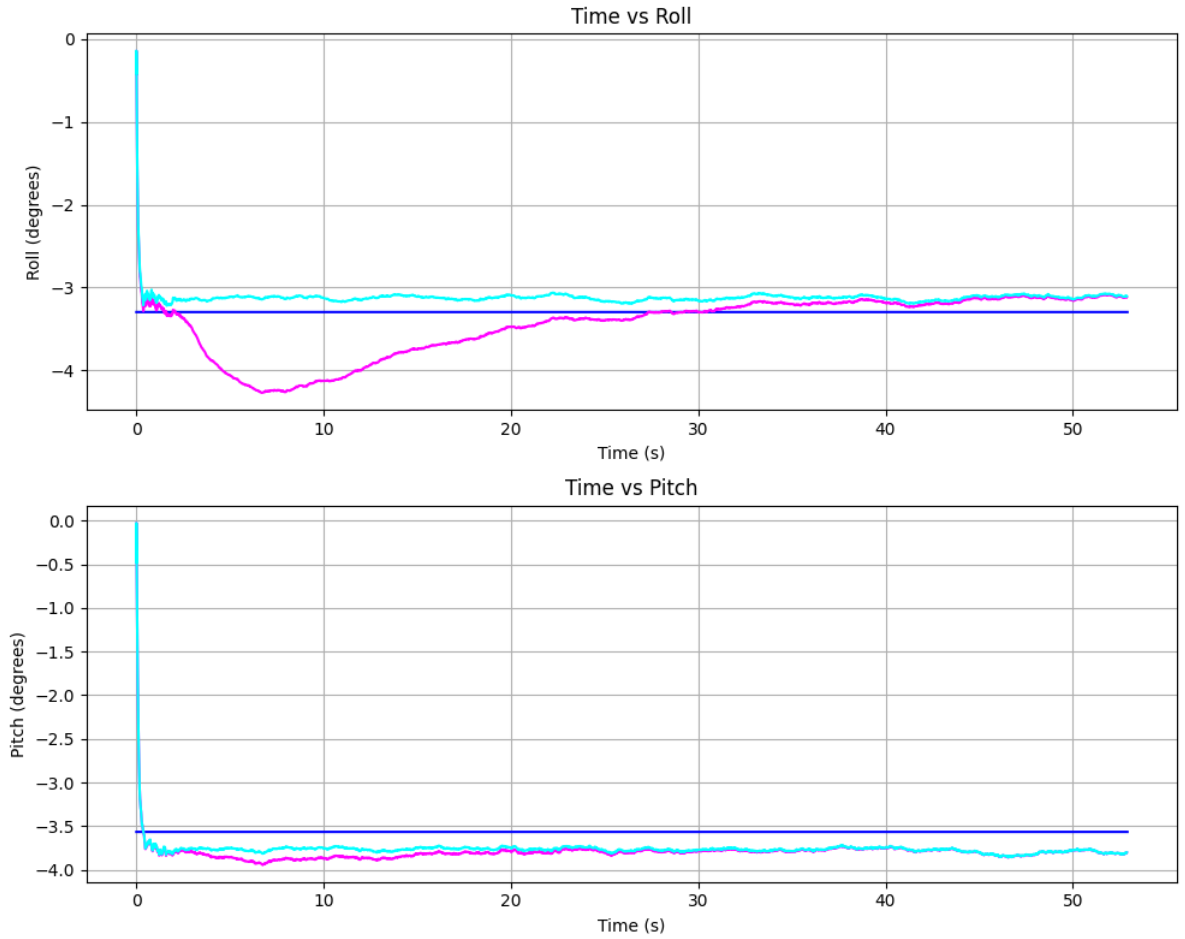

Figure 4.2: Comparison of gyroscope calibration versus no calibration in a stationary scenario.

In Figure 4.2, the reference values are approximately stable at roll $\approx -3.2°$ and pitch $\approx -3.5°$. When gyroscope calibration is not applied, the roll takes 40 seconds to stabilize having

a maximum one degree of error in the first ten seconds, while the pitch, despite being relatively stable, takes about 20 seconds to fully converge.

Although the gyroscope offset calibration in any of the axes is less than one degree, it affects stabilization. Notably, even without calibration, the system eventually stabilizes, though it initially drifts and takes time to correct itself. This is because the Fusion library inherently compensates for drift over time, preventing long-term accumulation.

4.1.2 Magnetometer

Applying the ellipsoid fitting calibration method, the magnetometer data was corrected, resulting in the sphere depicted in Figure 4.3. The corresponding calibration parameters, including the soft iron matrix and hard iron offset, are presented in Equation (4.2) and Equation (4.3), respectively.

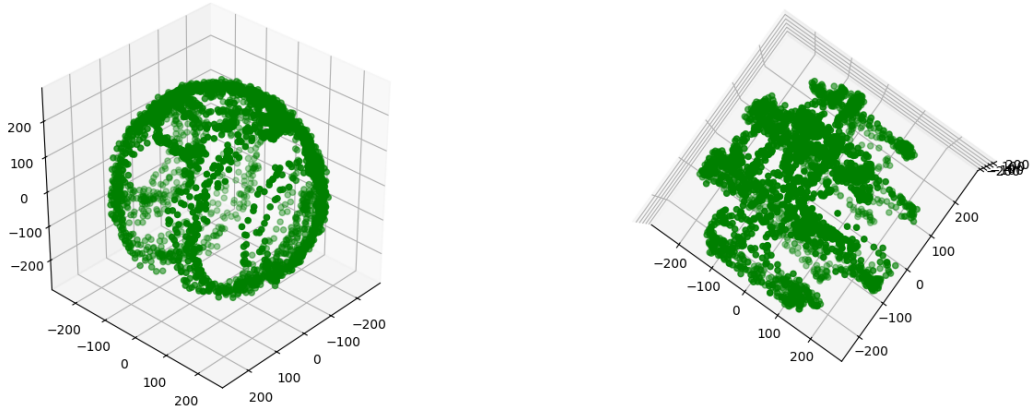

Figure 4.3: Magnetometer sphere with $\mathcal{F} = 250$.

$$\text{Soft iron} = \begin{bmatrix} 1.08366655 & -0.0187489 & 0.01229909 \\ -0.0187489 & 1.04376805 & -0.00784824 \\ 0.01229909 & -0.00784824 & 1.05946304 \end{bmatrix} \tag{4.2}$$

$$\text{Hard iron} = [12.55537129, -68.81162222, -57.65193058] \tag{4.3}$$

When applying only magnetometer calibration in a stationary scenario, the roll and pitch angles remain unaffected. This is evident in Figure 4.4, where the Fusion Advanced method and its calibrated counterpart yield identical results, as indicated by the overlapping pink and light blue lines.

This finding suggests that magnetometer calibration does not influence the computation of roll and pitch angles in the stationary state. Consequently, it raises the question of whether yaw remains similarly unaffected, which would imply that the magnetometer is not critical for determining the Tait-Bryan angles.

To investigate this, an experiment was conducted using both a physical compass and a smartphone compass application. The compass was manually aligned with geographic North, while the phone, although pointing in the same direction, displayed an approximate deviation of 10°. The sensor was also aligned with both the phone and the compass, as illustrated in Figure 4.5. However, due to the IMU's specific position and orientation within the sensor module, even when it is visually aligned with geographic north, as shown in the figure, the IMU

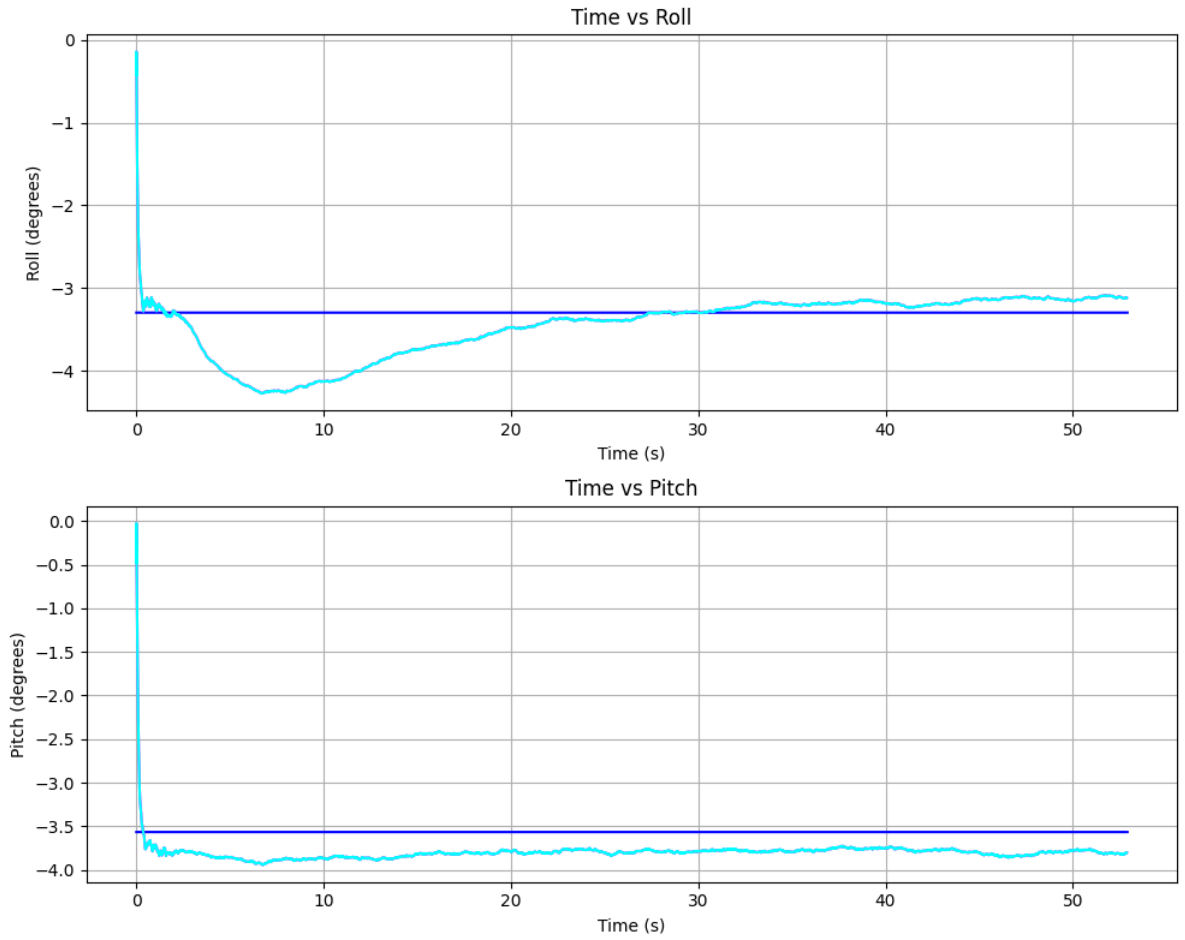

Figure 4.4: Comparison of magnetometer calibration versus no calibration in a stationary scenario.

internally registers a heading of $-90°$. To have the IMU report a heading of $0°$, the sensor would need to be rotated $90°$ to the right.

Figure 4.5: Relative positioning of the phone, compass, and sensor during the alignment experiment.

Figure 4.6 shows that:

- The Fusion Advanced With Calibration method yielded a yaw angle of approximately 80°, which corresponds to the 10° degrees that the phone is reading.
- The Fusion Advanced method produced a yaw of approximately −120°, introducing an error of around 40°.
- The Euler Direct algorithm, which derives yaw directly from the quaternion using the Python library `panda3d` (as referenced in Section 3.4.4), resulted in a significantly different value of −160°.

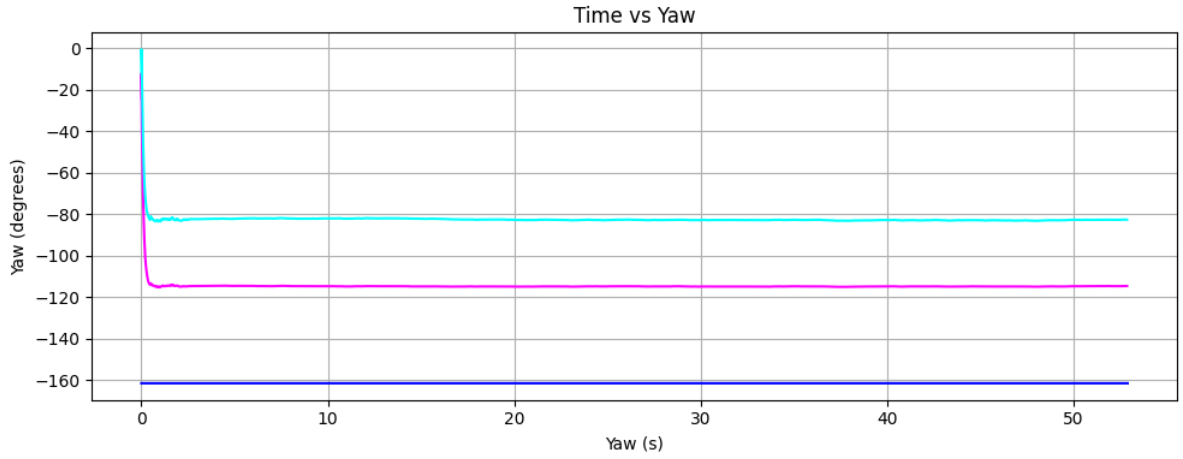

Figure 4.6: Comparison of magnetometer calibration versus no calibration in a stationary scenario for the yaw angle.

Based on the observed values, it appears that the phone's digital compass provides the most accurate reference, suggesting that the physical compass contains a slight error. The Fusion Advanced With Calibration algorithm yields an exact yaw estimation with no noticeable error. In contrast, the Fusion Advanced algorithm deviates by approximately 40°, and the Euler Direct algorithm exhibits a larger error of about 80°.

Since the magnetometer axes should remain equal or be rotationally proportional by a multiple of π, the −160° reading, when it should be −80°, suggests a misalignment in the Euler Direct algorithm.

4.1.3 Accelerometer

Using the same dataset employed for magnetometer calibration, the accelerometer calibration yields a significantly different ellipsoid, as shown in Figure 4.7. One of the most noticeable differences is that this ellipsoid appears much less uniformly distributed on its surface compared to the magnetometer ellipsoid in Figure 4.3. Instead, most data points are concentrated along two intertwined rings, leaving large empty areas.

This discrepancy can be explained by the fact that the gravity vector and the geomagnetic north vector occupy different positions in space. Since the ellipsoids are constructed based on these reference vectors, differences in their distribution are expected.

The obtained calibration parameters are as follows:

$$\text{Accelerometer misalignment} = \begin{bmatrix} 1.02497 & -0.00726 & 0.01056 \\ -0.00726 & 1.0165 & -0.00169 \\ 0.01056 & -0.00169 & 0.94969 \end{bmatrix} \tag{4.4}$$

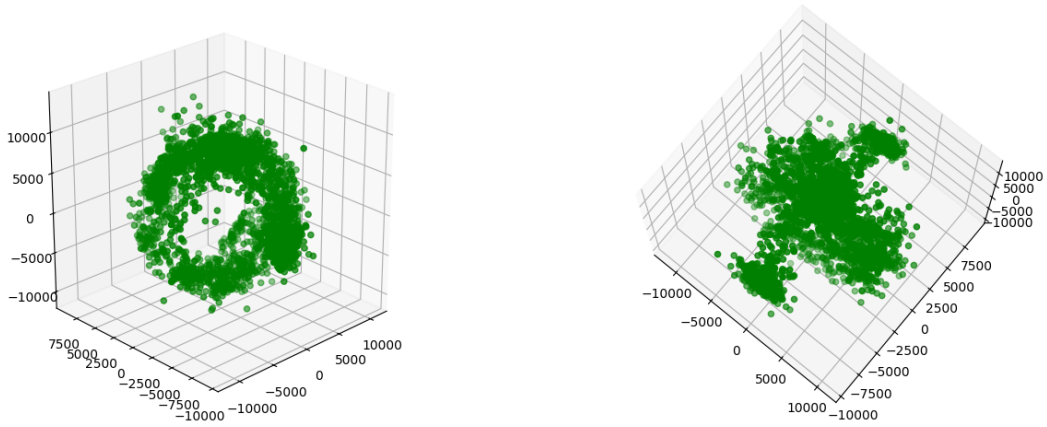

Figure 4.7: Accelerometer ellipsoid.

$$\text{Accelerometer offset} = [0.02287475586, -0.02744873047, 0.013828125] \tag{4.5}$$

When applying accelerometer calibration in the Fusion algorithm, in Figure 4.8, it is observed that the roll follows the same trend as without calibration but with a slight offset. The pitch exhibits a similar behavior, though it is slightly closer to the expected values, thereby reducing the error.

However, an important observation is that when the Fusion algorithm without calibration stabilizes around the 35-second mark, it actually produces better results than the calibrated version. The calibrated output exhibits an error of approximately 1.25°, which, although it converges, does not reach the correct value.

This suggests that if the calibration is not performed with high precision, which is likely in this case, as it was done manually and the ellipsoid lacks full surface coverage, the results may actually be more accurate without calibration.

In an effort to obtain a more uniform spherical distribution, an alternative approach was adopted. The initial issue stemmed from significant gaps across the surface of the sphere used for calibration. Rather than using a controlled set of rotations around the sensor's six axes, this method involved collecting data through random movements performed over an extended period. Instead of acquiring 3,000 samples, the dataset was increased to 30,000 samples. This increase not only provided a larger volume of data but also introduced greater directional variability due to the randomness of the motion.

Given that the data collection was performed manually, achieving uniform coverage across the entire spherical surface was not feasible. To address this, a post-processing algorithm was implemented to reduce the density in over sampled regions. The algorithm selected a subset of 3,000 points, maximizing the spatial separation between them. This filtering process resulted in a more evenly distributed set of points across the sphere, as can be seen in Figure 4.9, effectively minimizing empty regions and producing a more ideal spherical calibration reference.

The calibration parameters obtained using the improved spherical data distribution are as follows:

$$\text{Accelerometer misalignment} = \begin{bmatrix} 0.95213335 & 0.00826761 & 0.01260375 \\ 0.00826761 & 1.04511509 & 0.02500396 \\ 0.01260375 & 0.02500396 & 0.89267858 \end{bmatrix} \tag{4.6}$$

$$\text{Accelerometer offset} = [0.03085829069, -0.1084436947, 0.04438916199] \tag{4.7}$$

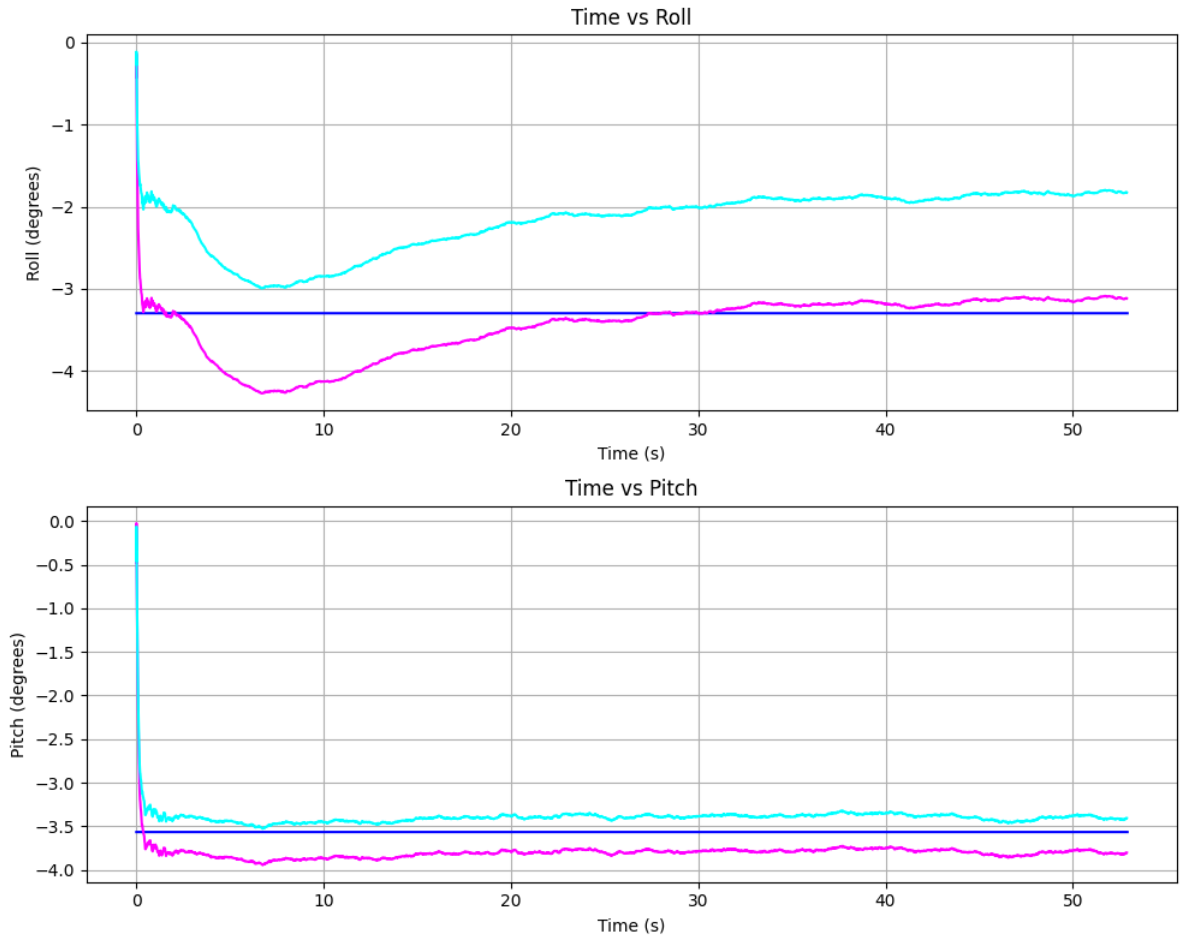

Figure 4.8: Comparison of accelerometer calibration versus no calibration in a stationary scenario.

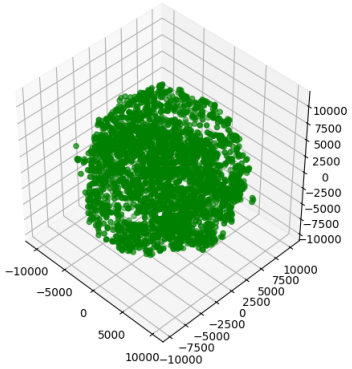

Figure 4.9: Accelerometer ellipsoid with 30,000 samples.

Using these updated parameters, the results shown in Figure 4.10 were obtained. The pitch angle exhibited an error of 0.5°, which is comparable to the previous calibration results. However, the roll angle showed a larger deviation, with an error of 8°, indicating a decline in performance compared to the earlier outcomes.

One possible explanation for the reduced accuracy is visible in Figure 4.9, where the presence of a second, smaller sphere inside the expected outer one can be observed. Ideally, the calibration should result in a single well-defined sphere at $\mathcal{F} = 8192$. The presence of two distinct spherical shapes suggests that, due to the extended duration and randomness of the motion during data

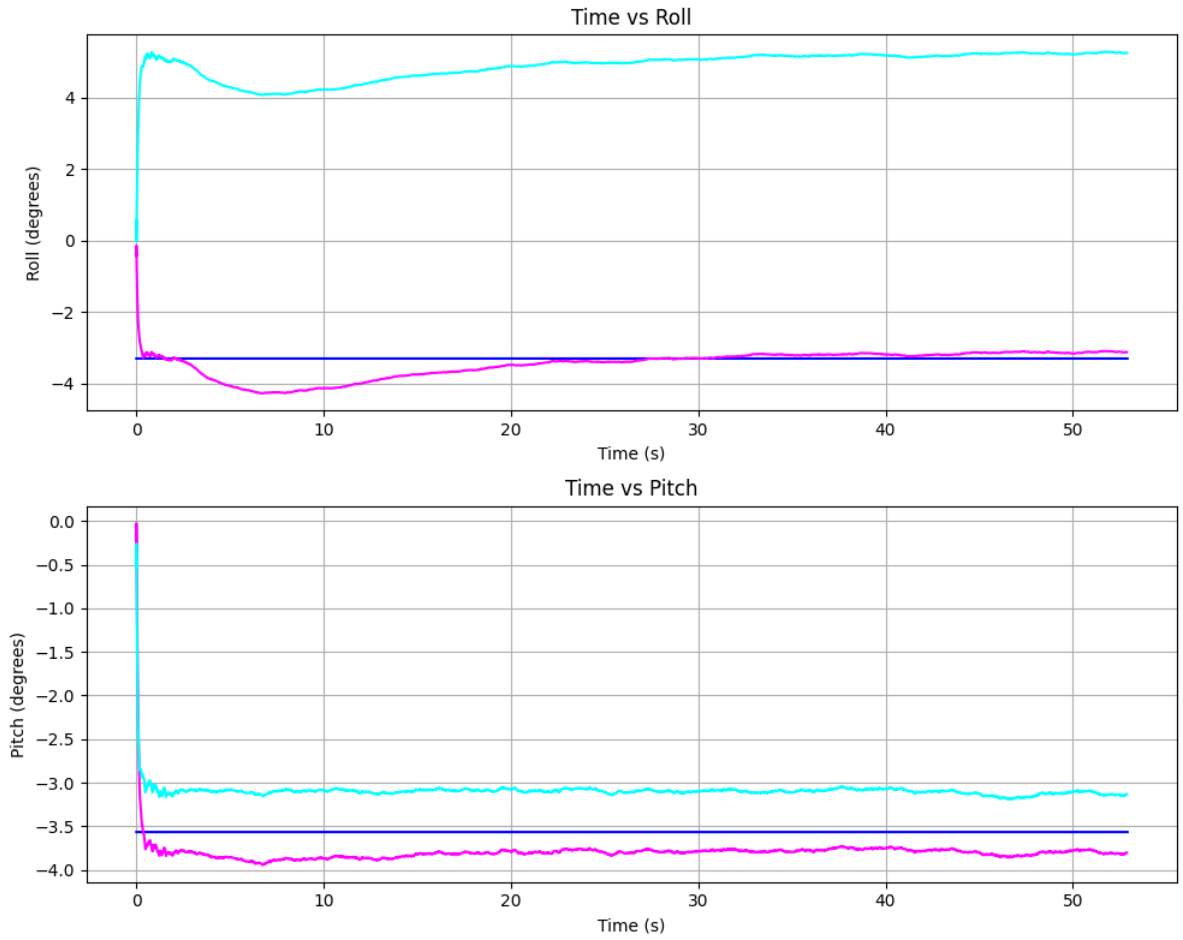

Figure 4.10: Comparison of accelerometer calibration versus no calibration in a stationary scenario.

collection, centrifugal accelerations may have been introduced. These nonlinear accelerations could have distorted the measurement data, leading to less accurate calibration results.

4.1.4 Summary of calibration

A brief summary of all the results mentioned in the previous sections about the calibration of the magnetometer, gyroscope, and accelerometer will be provided.

- **Gyroscope calibration** significantly improved the stabilization time during stationary scenarios. Although the raw offset values were relatively small, their correction led to faster convergence of roll and pitch, reducing initial drift. Despite this, the Fusion algorithm could still stabilize over time without calibration, but slower.
- **Magnetometer calibration** had little effect on roll and pitch in stationary scenarios, but showed a clear impact on the yaw angle. The calibration process, particularly ellipsoid fitting, corrected substantial heading errors, reducing yaw deviation from up to 80° down to an almost perfect alignment.
- **Accelerometer calibration** showed mixed results. The initial calibration using a small dataset introduced minor improvements in pitch estimation but introduced errors in roll. Another attempt using a bigger and filtered dataset improved pitch accuracy but degraded roll accuracy. In general, results were not that good compare to the uncalibrated version, so

the conclusion for the accelerometer is that it is hard to obtain good results for calibration doing it by hand or without any more sophisticated hardware.

In conclusion, while each calibration step addresses different aspects of sensor error, their effectiveness strongly depends on the quality of data and precision of the procedure. Gyroscope and magnetometer calibrations clearly improved accuracy in their respective domains. However, accelerometer calibration, unless carefully performed, may not guarantee better results and can potentially introduce additional error.

4.2 Results of different motions

It is essential to note that the sensor exhibits distinct behaviors depending on its motion. Consequently, different types of movement were recorded to analyze these variations. Each data set comprises measurements from the accelerometer, gyroscope, and magnetometer along all three axes, in addition to quaternion data.

- **Slow movement:** This data set was recorded by moving the sensor gradually across the surface of a table over short distances and at a very slow pace.
- **Rapid movement:** The rapid movement data set was obtained by simulating typical phone movements, such as rotating it between horizontal and vertical orientations, pointing it towards the sky, and mimicking actions such as capturing a panoramic photograph.
- **Stationary state:** In this scenario, the sensor remained completely still on the table for the duration of the recording.
- **Walking motion:** The sensor was affixed to a shoe to capture the data while walking.
- **Pendulum motion:** This data set was acquired by suspending the sensor using a cable and allowing it to oscillate in a pendulum-like motion.
- **Infinite movement:** This dataset simulates holding the sensor in hand and moving it in a figure-eight pattern.

4.2.1 Slow movement

As illustrated in Figure 4.11, all the evaluated algorithms provide a reasonable approximation of the actual motion of the sensor, effectively capturing the expected roll and pitch behavior.

It is worth noting that among all the algorithms, the Kalman Filter introduces the most noise into the signal, whereas the other methods exhibit a more stable response. This suggests that while Kalman filtering is a powerful approach, it may require further tuning to reduce noise in this specific application.

Throughout the experiment, there were periods in which the sensor remained completely stationary. One such interval can be observed in Figure 4.11, between approximately seconds 14 and 17. To better analyze this stationary phase, a zoomed-in version of this segment is presented in Figure 4.12.

In this detailed view, it is evident that the Euler Direct algorithm, along with the Kalman Filter, the Complementary Filter, and the Fusion method with calibration, correctly maintain a stable state as expected. However, Fusion algorithms that do not incorporate the magnetometer or calibration exhibit a drift effect, where they continue to move until they eventually stabilize at zero. This behavior indicates that the absence of additional corrections, such as magnetometer integration or calibration, can lead to a delayed convergence to the correct position.

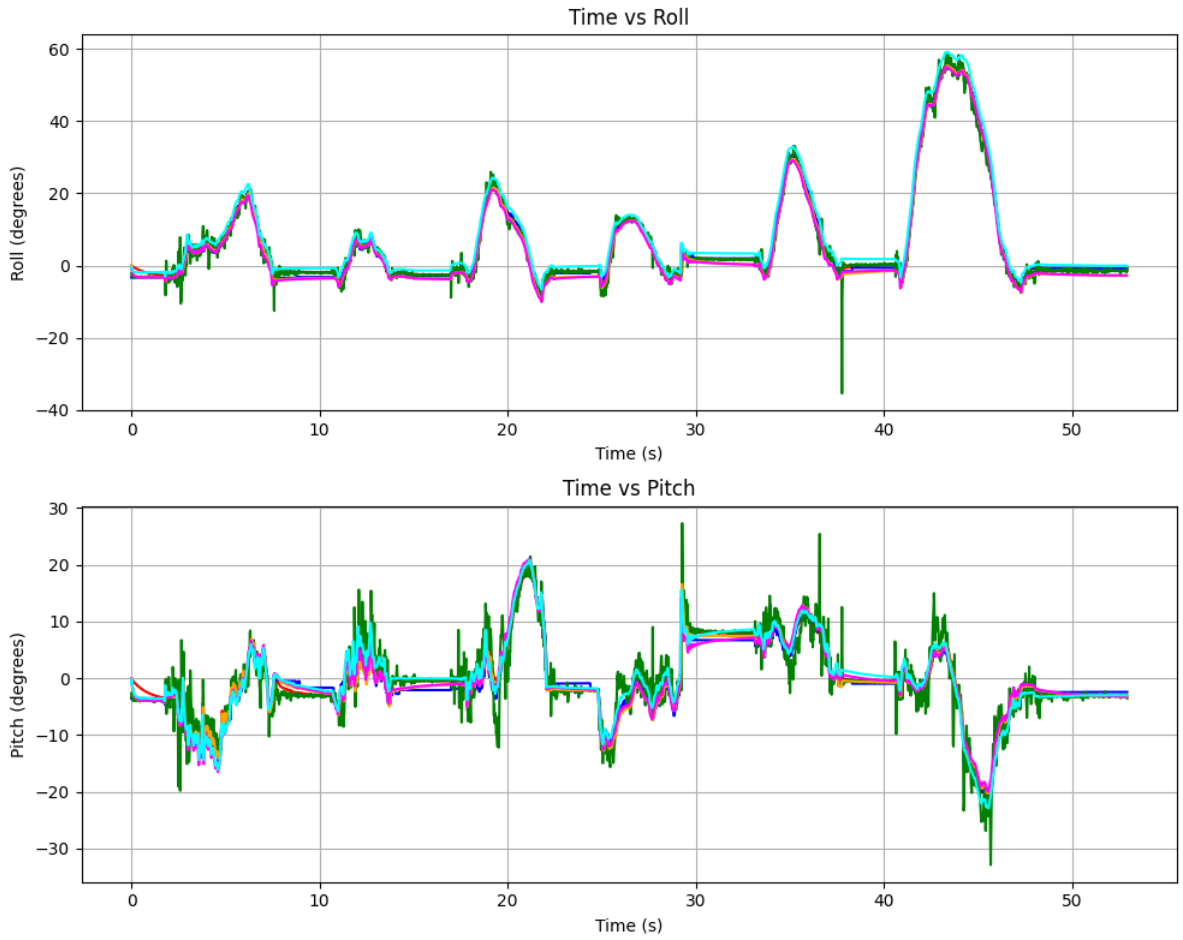

Figure 4.11: Roll and pitch during slow movement.

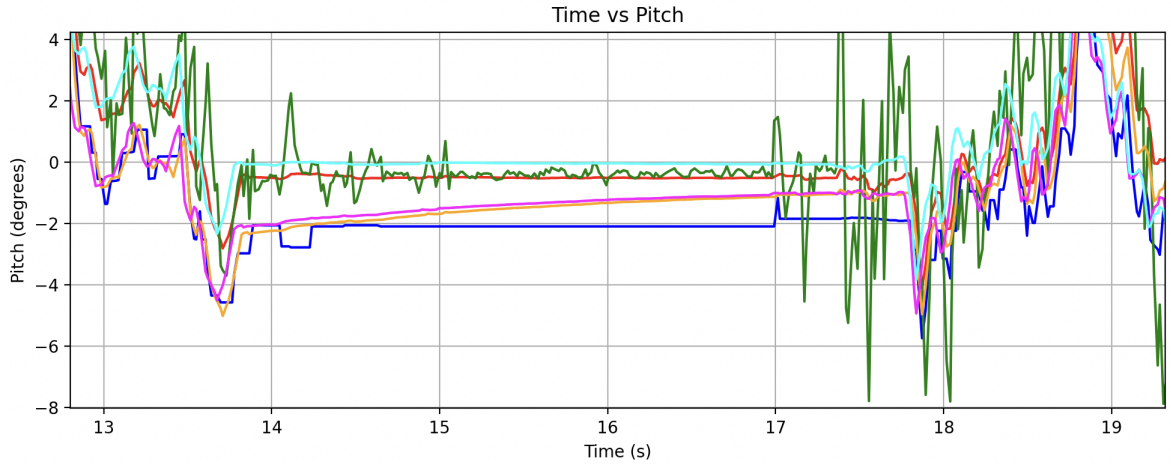

Figure 4.12: Roll and pitch during a stationary phase of slow movement.

4.2.2 Rapid movement

When maintaining the same parameters as in the slow movement during rapid movement, it becomes evident that not all algorithms produce results that closely align with each other. Unlike in slow-motion scenarios, where most algorithms follow a similar trajectory, high-speed movements introduce greater variations among them.

Notably, the Kalman Filter, which exhibited significant noise in slow movements, demonstrates a reduction in noise during rapid motion. However, it still remains noisier compared to the other algorithms, suggesting that while its performance improves with increased dynamics, it does not completely eliminate unwanted fluctuations.

Among all the tested methods, the three Fusion-based algorithms, both with and without calibration and magnetometer, provide the most accurate estimation of the real angles, closely following the expected trajectory. Their stability and consistency indicate that the fusion approach is well suited for handling rapid motion scenarios.

In contrast, both the Kalman Filter and the Complementary Filter initially track the motion well but encounter significant errors due to a gimbal lock issue. This problem particularly affects roll estimation, causing both algorithms to diverge from the expected values. It takes approximately 20 seconds, from second 10 to second 30, for them to gradually recover and converge back to the correct orientation.

Additionally, in the case of pitch estimation, the Complementary Filter continues to register movements but introduces a noticeable delay before it is able to realign with the actual motion. This lag highlights a limitation in its ability to respond accurately in high-speed dynamic conditions.

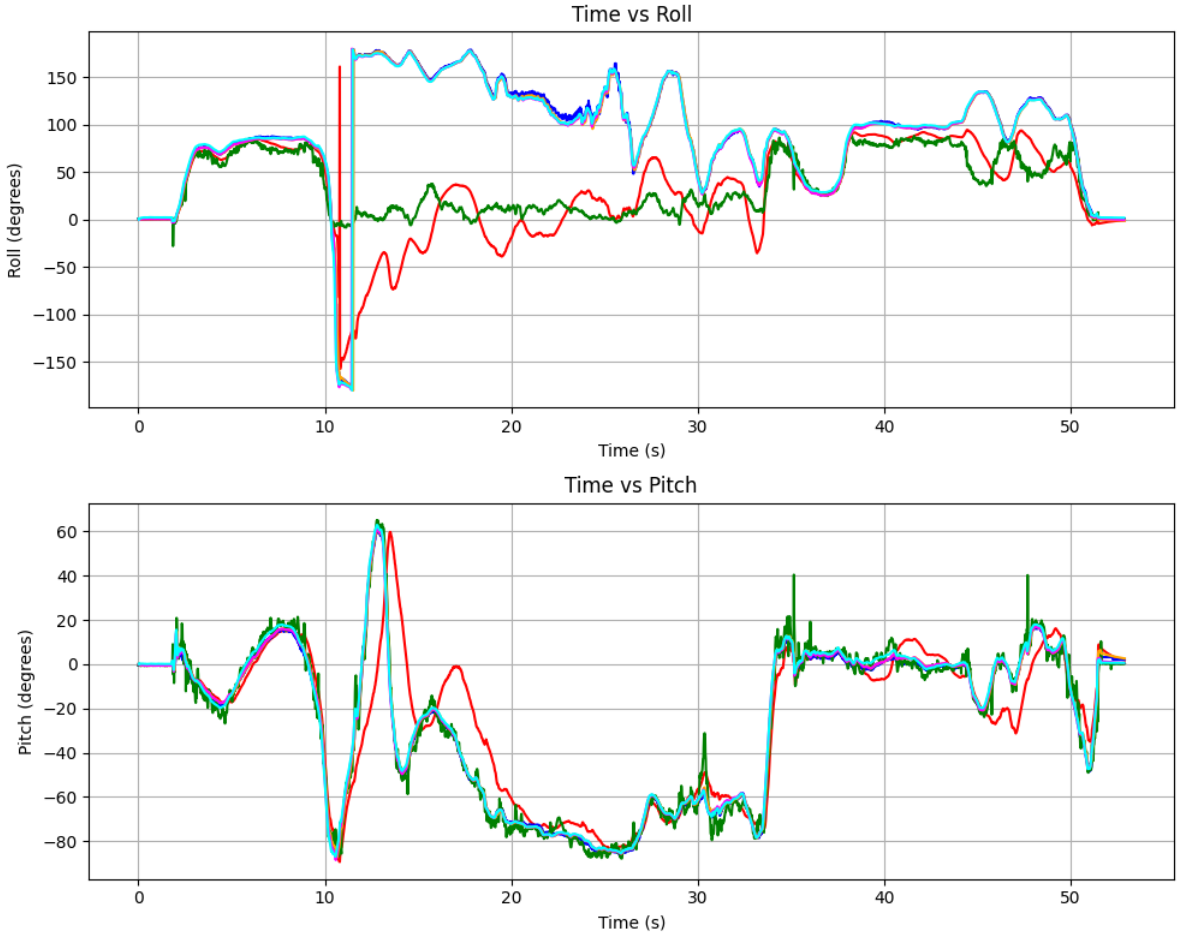

Figure 4.13: Roll and pitch during rapid movements.

To determine the optimal value of α for this type of movement, a study was carried out testing some values of α within the range $[0, 1]$. The resulting performance of the Complementary Filter under these varying conditions is shown in Figure 4.14.

Interestingly, while previous tests indicated that $\alpha = 0.02$ provided the best results in other

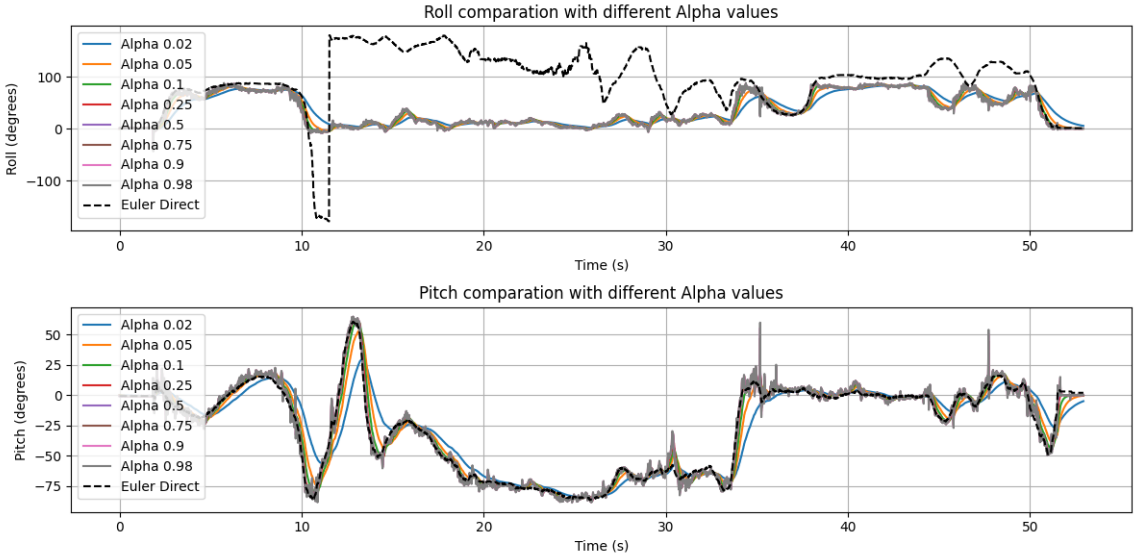

Figure 4.14: Effect of different α values during rapid movements.

scenarios, this is not the case for rapid movements. In fact, $\alpha = 0.02$ performs the worst under these conditions, while setting $\alpha = 0.5$ yields the most accurate results. After adjusting α accordingly, new data was collected, as illustrated in Figure 4.15.

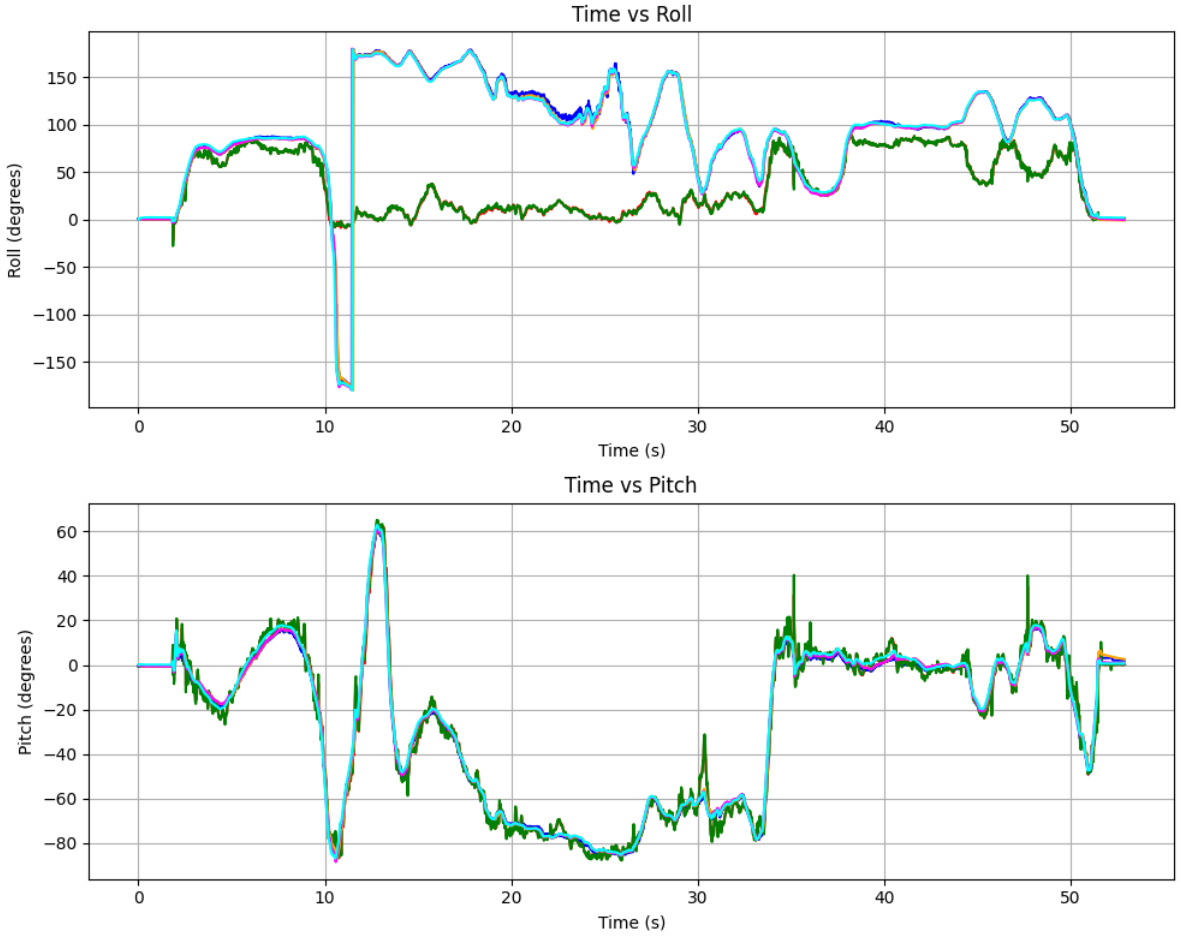

Figure 4.15: Roll and pitch during rapid movements with $\alpha = 0.5$.

From Figure 4.15, it can be observed that with $\alpha = 0.5$, the Complementary filter and the Kalman filter exhibit very similar behavior. However, increasing α introduces more noise in the Complementary Filter compared to its usual performance. Although the gimbal lock issue persists, the adjustment successfully eliminates the delay previously observed in the pitch angle estimation.

4.2.3 Stationary State

Results for the stationary state of the sensor can be seen in Figure 4.16.

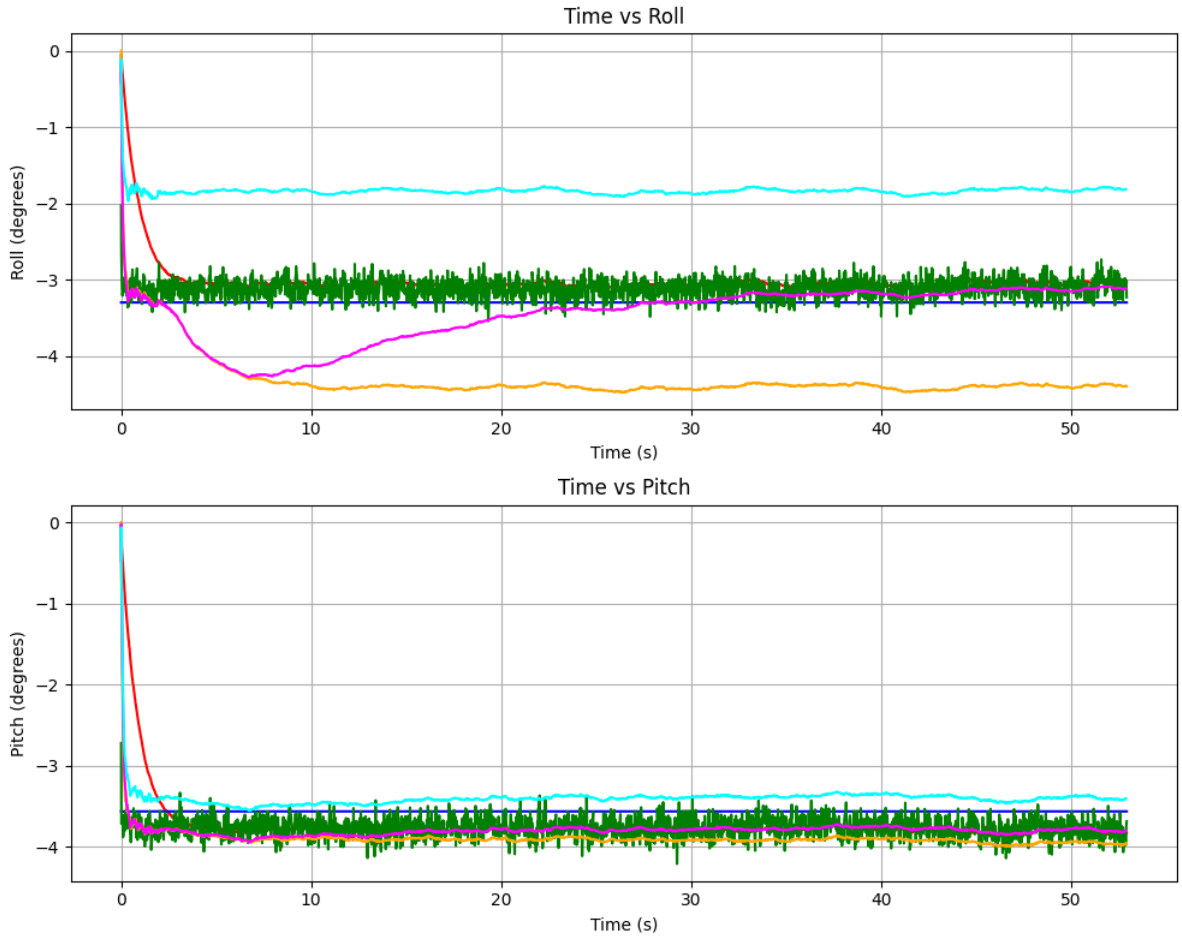

Figure 4.16: Roll and pitch during stationary state.

When the sensor is in a stationary state, all the values tested of α for the Complementary Filter produce very accurate results, with only minor differences between them. However, the best performance is observed at $\alpha = 0.25$. In this type of movement, the Complementary Filter performs very well, as it converges rapidly to the stationary state and remains the closest to the Euler Direct algorithm compared to the other methods.

Additionally, it can be observed that the Fusion Advanced algorithm takes about 22 seconds to converge, while the Fusion Advanced with Calibration converges straight. The Fusion Simple algorithm also exhibits delayed convergence but deviates more from the Euler Direct reference. The Kalman Filter continues to introduce significant noise but still provides results that are close to the expected values.

Regarding Fusion Advanced Calibration, a slight discrepancy can be seen in the roll angle compared to the Euler Direct. This raises the question of whether the Fusion Advanced Cali-

bration is actually more accurate than the Euler Direct method or if this small deviation is due to the inherent limitations of manual calibration, since achieving a perfectly precise calibration is nearly impossible by hand; these small errors could become noticeable. However, the observed deviation is minimal, approximately within a one-degree margin.

4.2.4 Walking Motion

The walking steps can be observed in Figure 4.17, where the movement pattern is clearly visible. Notably, at second 40, the walking motion ceases.

Most of the algorithms successfully capture the steps without significant deviations. However, the Kalman Filter introduces considerable noise, making its trajectory less stable.

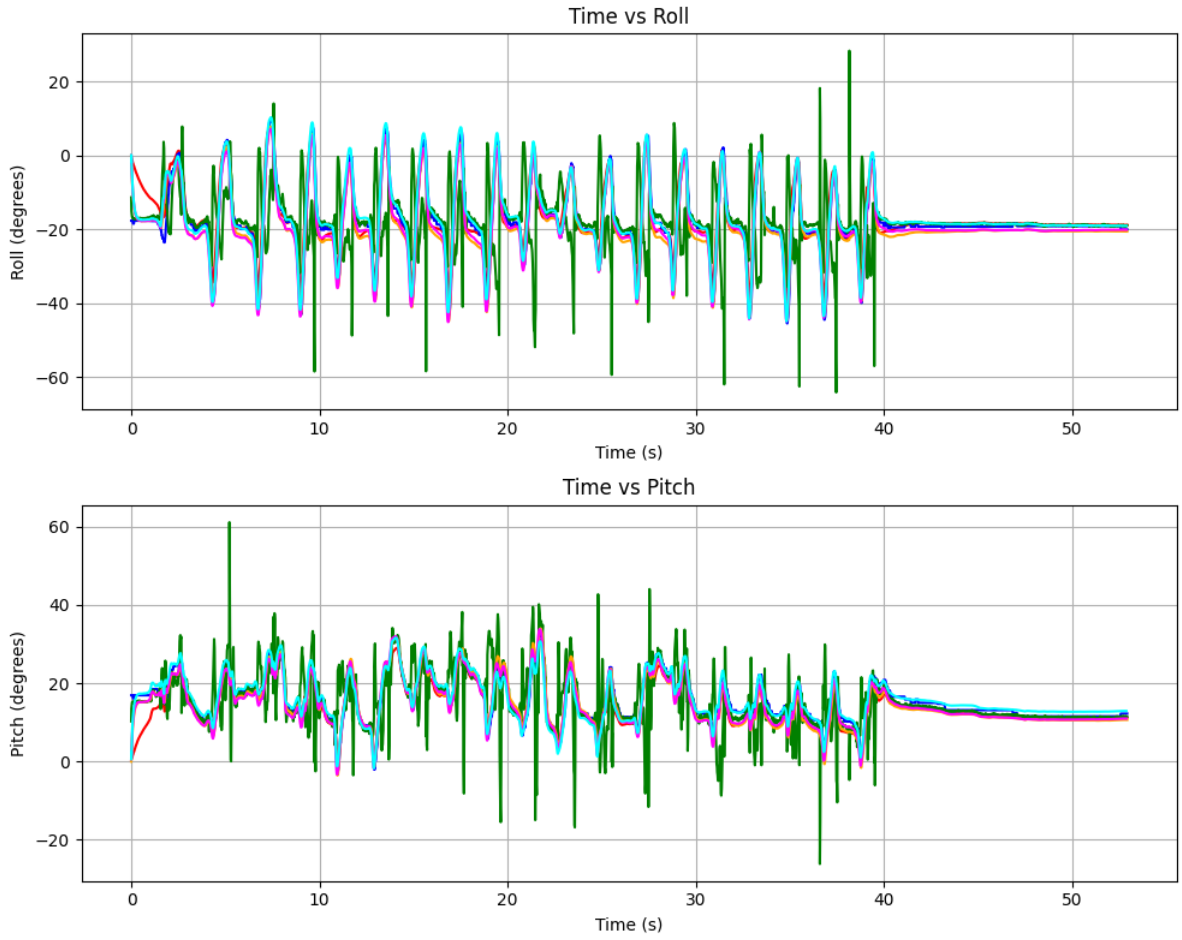

Figure 4.17: Roll and pitch during walking movement.

When zooming in on seven consecutive steps, as shown in Figure 4.18, it becomes evident that the Kalman Filter experiences a slight delay of a few seconds, failing to detect the steps at the correct moments.

Further analyzing a single step, illustrated in Figure 4.19, it is evident that the Fusion Advanced Calibration algorithm provides the most accurate and stable result, closely aligned with the Euler Direct algorithm. In fact, it demonstrates even greater stability in comparison.

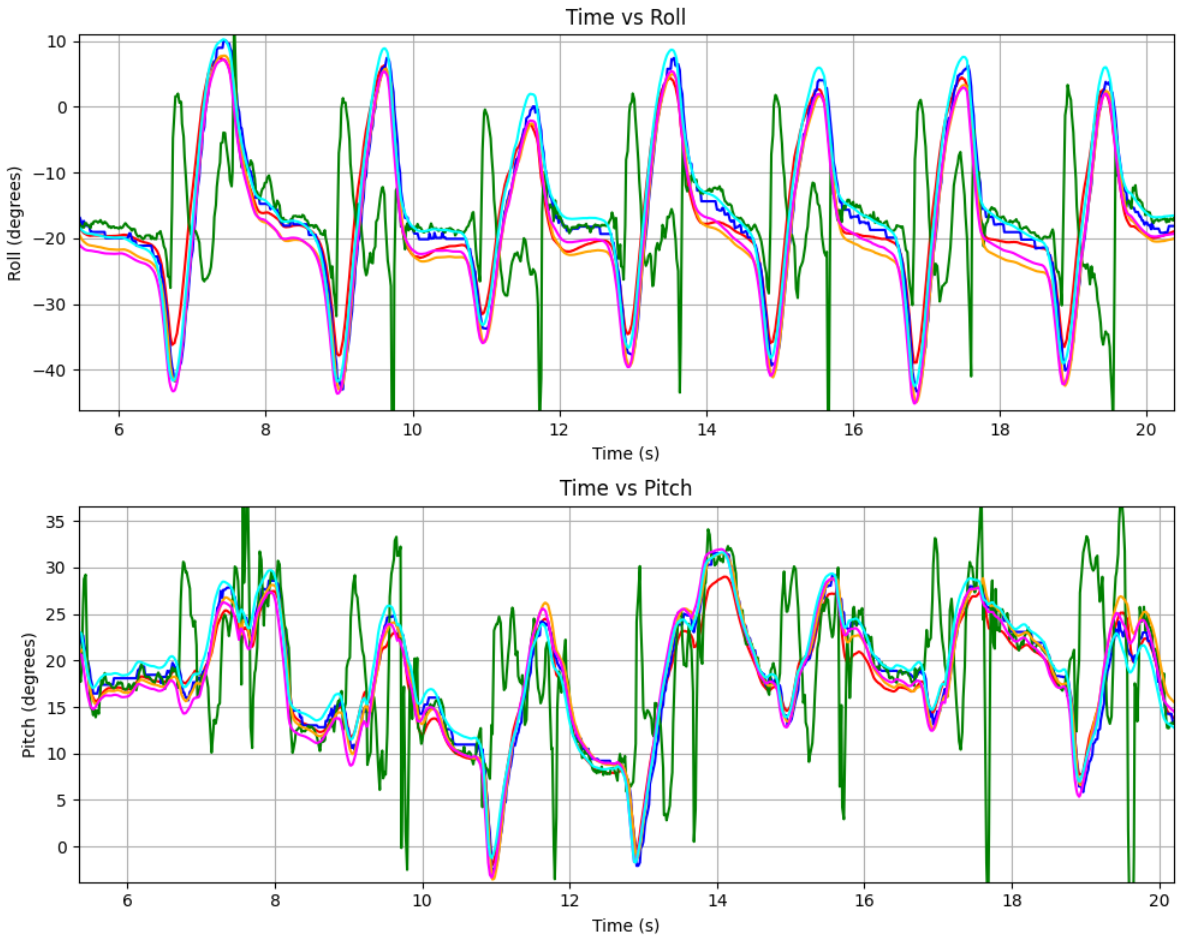

Figure 4.18: Roll and pitch during seven steps of walking movement.

4.2.5 Pendulum motion

The pendulum motion is illustrated in Figure 4.20, where it initially exhibits large oscillations that gradually decrease over time.

At the beginning of the motion, the sensor may experience disturbances due to being manually released. Therefore, it is advisable to analyze the data after the motion stabilizes, approximately around the 5-second mark.

It can be observed that the Complementary filter converges, but not to the correct value, showing a significant discrepancy in both roll and pitch. Additionally, the Fusion Advance filter takes until around the 30-second mark to converge, as can be seen in Figure 4.21, whether the Fusion Advance with Calibration achieves convergence almost immediately. Unlike the other data sets, in this case, the Kalman filter appears highly stable, resembling a stationary state with minimal noise. Interestingly, it does not seem to detect the pendulum's motion.

4.2.6 Infinite movement

When the sensor follows a figure-eight trajectory, the corresponding pitch and roll measurements are obtained, as illustrated in Figure 4.22.

It can be observed that all the evaluated algorithms yield relatively accurate results, with the pitch estimation demonstrating greater precision compared to the roll estimation. Furthermore, it is noticeable that the Kalman filter introduces a higher level of noise compared to the other

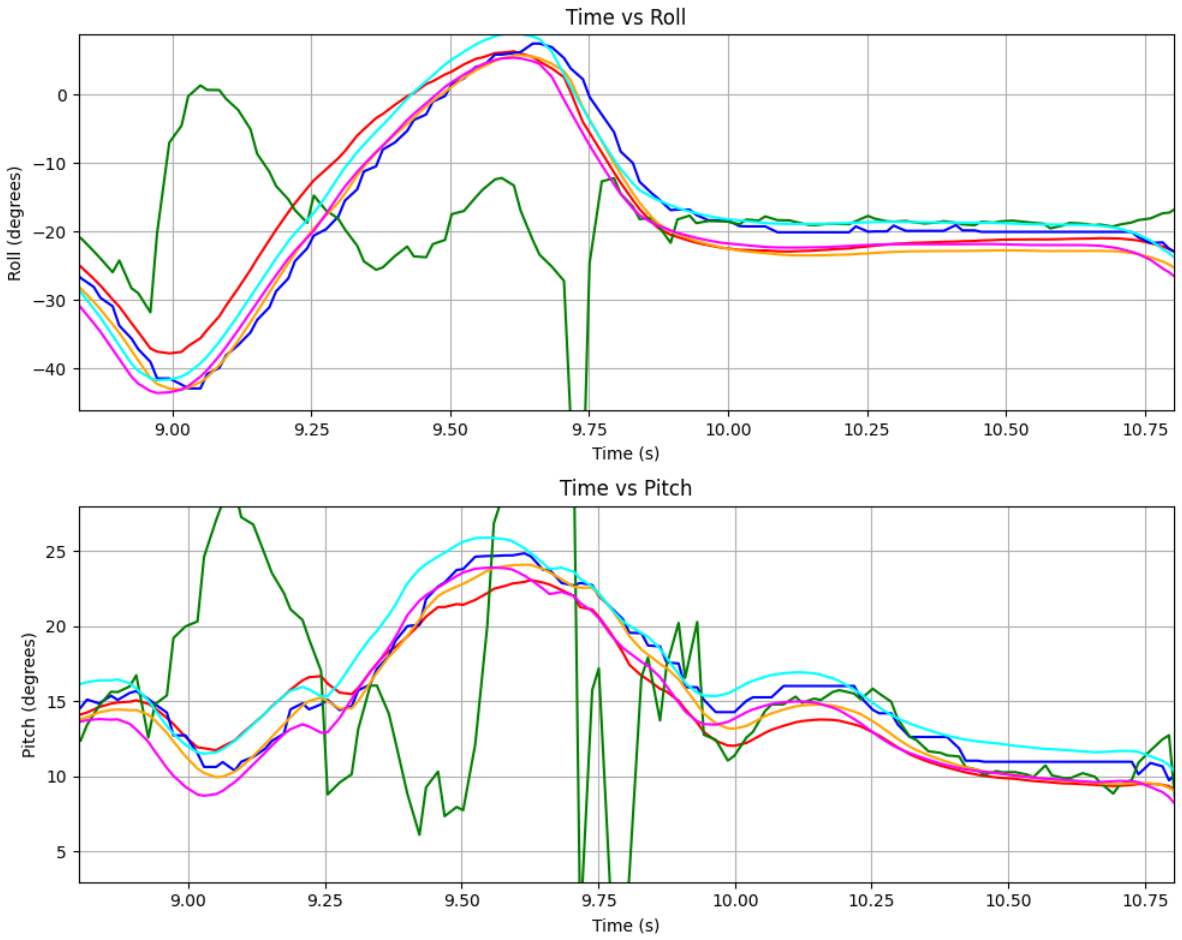

Figure 4.19: Roll and pitch during a single step of walking movement.

algorithms.

A more detailed examination, as depicted in Figure 4.23, particularly within the time interval between 20 and 30 seconds, reveals that although the Kalman filter does not lose track of the movement, it tends to underestimate the amplitude of roll variations. A similar behavior is observed with the Complementary Filter.

4.2.7 Summary of results

This section provides a summary of the results presented in the previous sections. These results are shown in Table 4.1, which includes the mean roll error over time, the mean pitch error, and the total error.

According to Table 4.1, the `Fusion Advanced Calibration` algorithm yields the best overall performance. All versions of `Fusion` generally outperform the other methods, being selected as the best in 5 out of 6 motion scenarios. Although the Kalman Filter achieves the best result in the stationary state, Figure 4.16 shows that it introduces considerable noise and is identified as the worst-performing algorithm in 3 out of 6 cases.

Based on these conclusions, the `Fusion` algorithm is considered the most suitable choice for orientation estimation. Consequently, all subsequent experiments will be conducted using this algorithm in its variants.

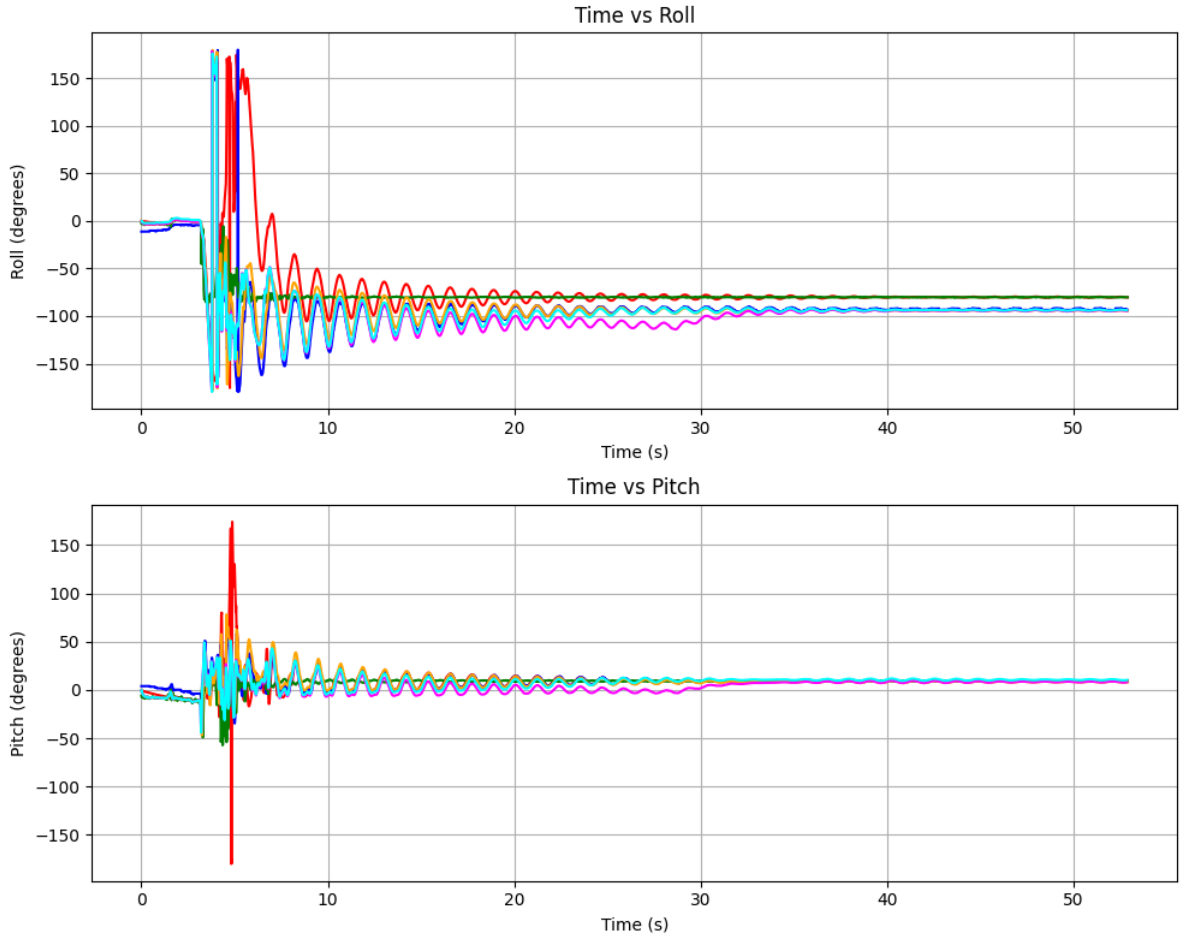

Figure 4.20: Roll and pitch during a pendulum motion.

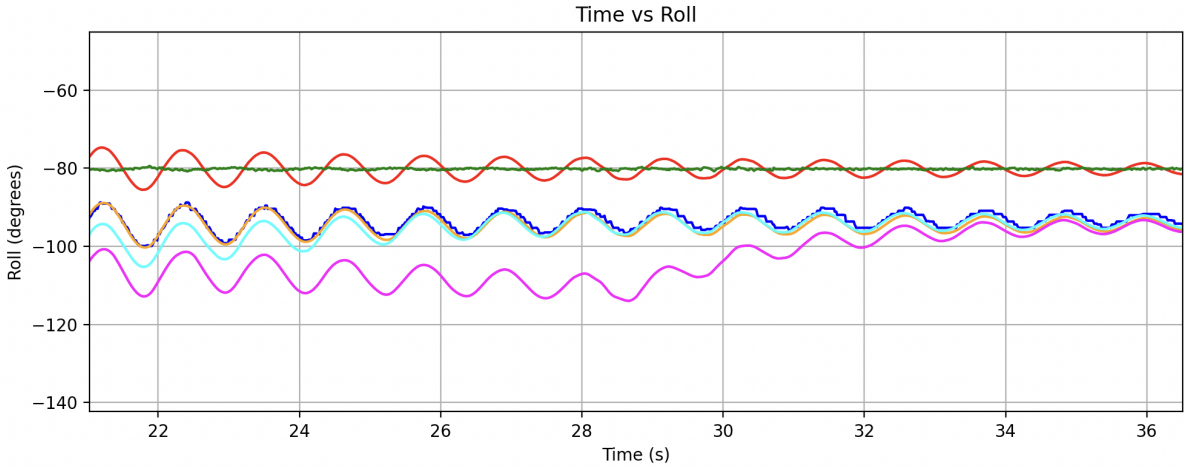

Figure 4.21: Closer look at roll and pitch during a pendulum motion.

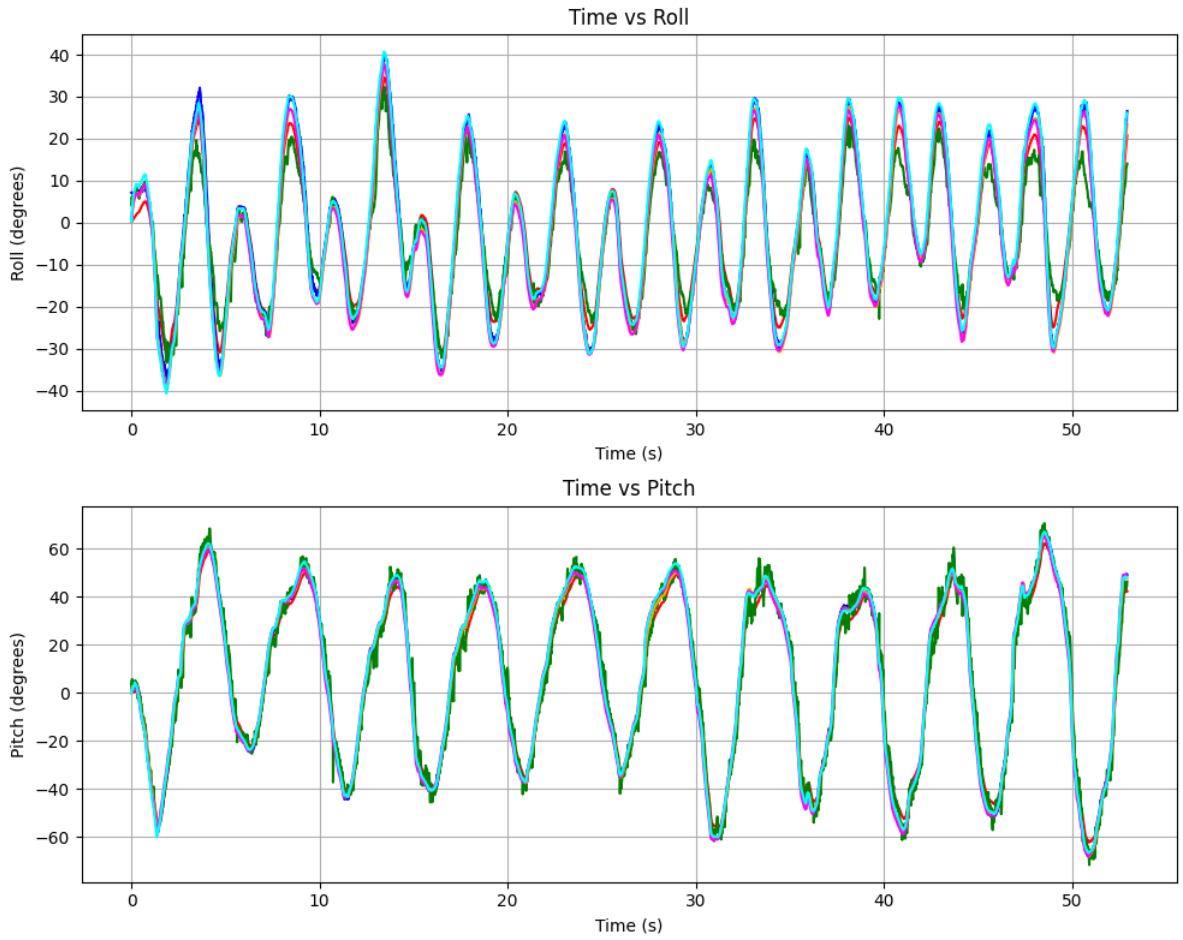

Figure 4.22: Pitch and roll variations during a figure-eight motion pattern.

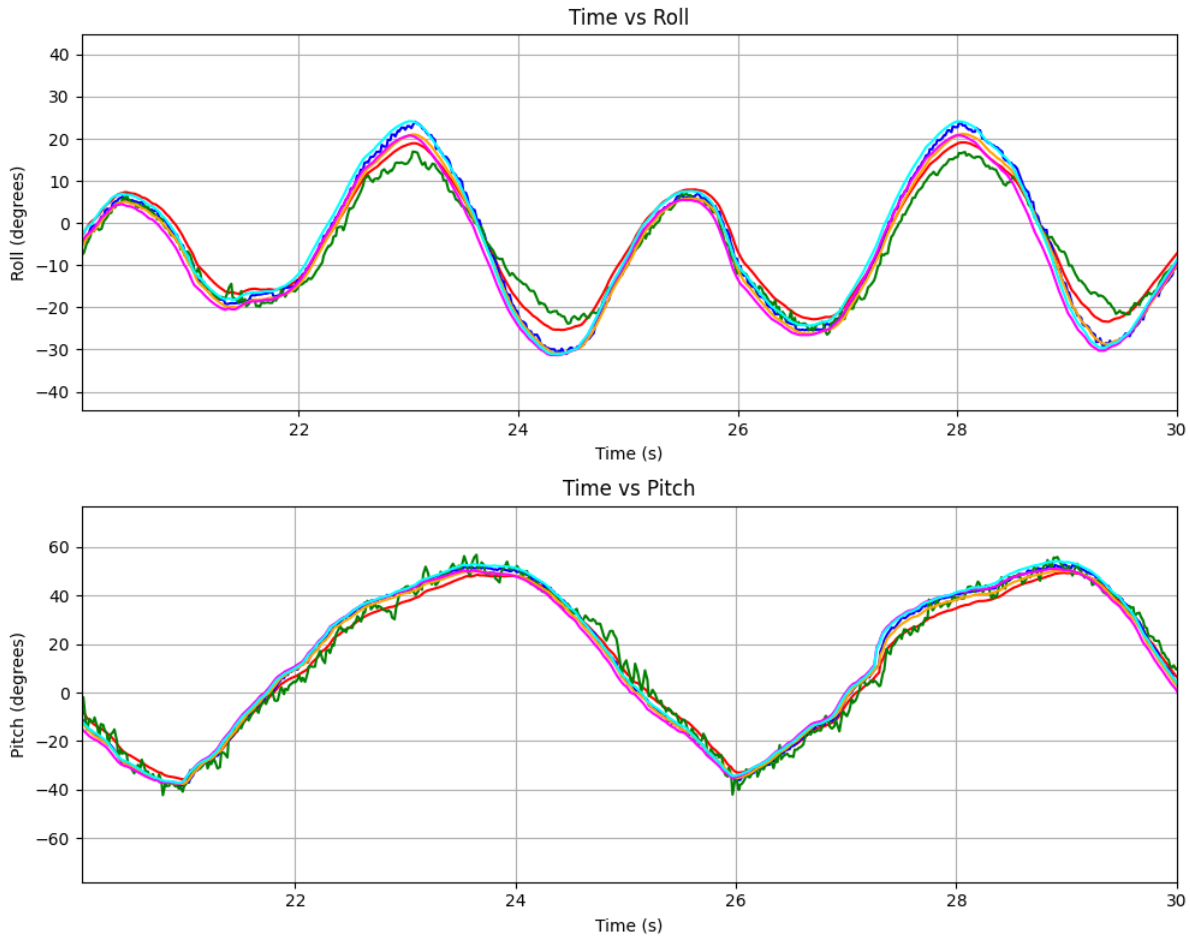

Figure 4.23: Detailed analysis of a 10-second interval of pitch and roll variations during a figure-eight motion pattern.

Motion Type	Algorithm	Mean Roll Error (°)	Mean Pitch Error (°)	Total Error (°)
Stationary state	Complementary Filter	0.232	0.237	0.235
	Kalman Filter	0.189	0.206	0.197
	Fusion Simple	1.023	0.347	0.685
	Fusion Advanced	0.279	0.247	0.263
	Fusion Adv. Calibration	1.460	0.200	0.830
	Euler Direct (Reference)	0	0	0
Rapid movement	Complementary Filter	63.153	9.136	36.145
	Kalman Filter	61.566	2.803	32.185
	Fusion Simple	1.602	0.891	1.247
	Fusion Advanced	1.600	0.829	1.214
	Fusion Adv. Calibration	1.746	1.256	1.501
	Euler Direct (Reference)	0	0	0
Slow movement	Complementary Filter	0.625	1.015	0.820
	Kalman Filter	1.143	1.935	1.539
	Fusion Simple	0.913	0.627	0.770
	Fusion Advanced	0.876	0.622	0.749
	Fusion Adv. Calibration	1.982	1.052	1.517
	Euler Direct (Reference)	0	0	0
Walking motion	Complementary Filter	2.369	1.719	2.044
	Kalman Filter	7.040	3.668	5.354
	Fusion Simple	1.855	1.540	1.697
	Fusion Advanced	1.542	1.393	1.468
	Fusion Adv. Calibration	0.932	0.888	0.910
	Euler Direct (Reference)	0	0	0
Pendulum motion	Complementary Filter	20.357	3.419	11.888
	Kalman Filter	18.222	4.683	11.452
	Fusion Simple	4.758	2.353	3.556
	Fusion Advanced	7.772	5.305	6.539
	Fusion Adv. Calibration	4.633	2.930	3.781
	Euler Direct (Reference)	0	0	0
Infinite movement	Complementary Filter	2.608	2.879	2.743
	Kalman Filter	4.386	3.486	3.936
	Fusion Simple	1.527	1.012	1.269
	Fusion Advanced	1.819	1.076	1.447
	Fusion Adv. Calibration	1.099	1.139	1.119
	Euler Direct (Reference)	0	0	0

Table 4.1: Comparison of mean pitch, roll, and total errors for each algorithm under different motion types, relative to Euler Direct.

4.3 Gait tracking

Four versions of gait tracking were implemented, all based on the approach described in [72], which utilizes the Fusion algorithm [71]. The first version involves performing gait tracking using only an accelerometer and gyroscope, with no calibration. However, it is important to note that Fusion itself calibrates the gyroscope automatically. The second version adds calibration to both the accelerometer and the gyroscope, while the third version incorporates a magnetometer and calibrates all three sensors: accelerometer, gyroscope, and magnetometer. The fourth version is a hybrid model of the second and third versions.

Each version applies the Fusion algorithm to compute the Tait-Bryan angles, and further uses it to estimate the Earth's acceleration. This acceleration is then integrated to obtain velocity, and velocity is subsequently integrated to determine position.

In addition, various flags and error values were computed and visualized in graphs.

To evaluate the performance of the algorithms, a walking sample of approximately 50 seconds was recorded. The error was assessed by starting and finishing at the same location and measuring the distance between the corresponding points.

4.3.1 Gait tracking using gyroscope and accelerometer data

This section presents a gait tracking algorithm primarily inspired by the implementation provided by xioTechnologies [72]. The algorithm utilizes data from the gyroscope and accelerometer to estimate the walking trajectory of a person.

As shown in Figure 4.24, the reconstructed walking path closely resembles the actual trajectory, demonstrating the effectiveness of the approach. However, there is a noticeable offset at the final position, which is approximately 63 cm away from the true endpoint. This displacement could be attributed to accumulated integration errors and sensor drift over time.

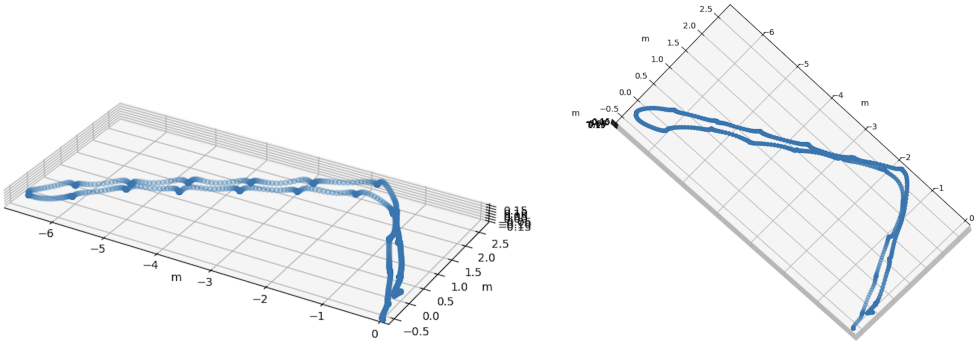

Figure 4.24: Reconstructed walking path using gyroscope and accelerometer data.

A more detailed analysis is provided in Figure 4.25, where acceleration, velocity, and position are plotted over time. From these signals, it is possible to identify individual steps, approximately 21 in total, based on the periodic increases and decreases in acceleration and velocity. Each step is characterized by a peak in acceleration and velocity during the swing phase, followed by a sharp drop when the foot contacts the ground.

The position graph indicates that the halfway point of the walk occurred around the 20-second mark, while the endpoint was reached near second 35. At this point, all variables—acceleration, velocity, and position—converge to values close to zero, consistent with the end of the motion. In addition, a binary activity indicator reveals when the subject was in motion, which aligns with the periodicity of the gait cycle.

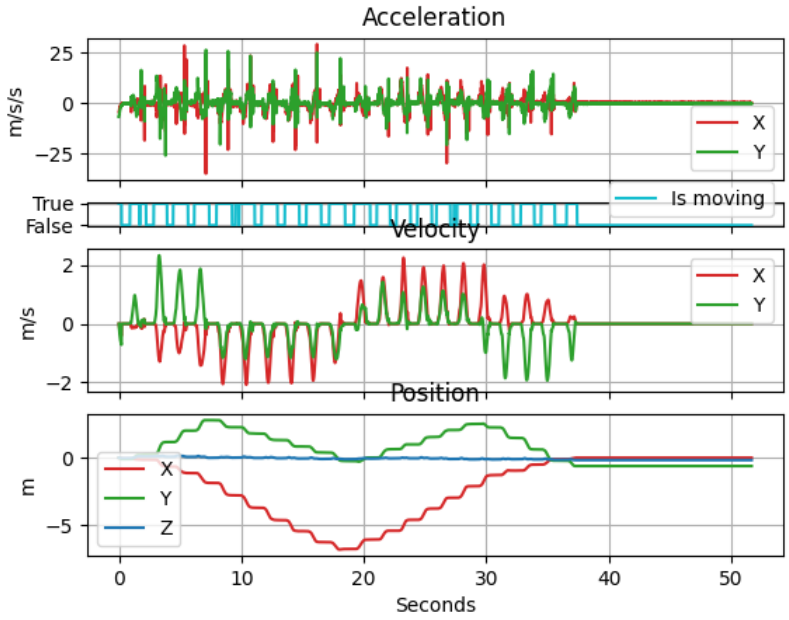

Figure 4.25: Acceleration, velocity, and position over time during the gait sequence.

Further insights are provided in Figure 4.26, which includes the raw gyroscope and accelerometer readings, the Tait-Bryan orientation angles, and various internal status flags such as acceleration error, accelerometer ignored, and recovery triggers.

A close examination of the raw data confirms the presence of the 21 steps, each associated with spikes in sensor values corresponding to the foot movements. The Tait-Bryan angles show consistent patterns for roll and pitch during each step, while the yaw angle reflects changes in the walking direction. Initially, the yaw remains close to 0°, then increases to approximately 90° during the first turn, stabilizes, and shifts to 180° or −180° during the turnaround. Finally, it settles around −90° before reaching −180° at the final 90° turn, indicating proper capture of heading changes.

The acceleration error metric reaches values as high as 100, particularly during active steps, signifying deviations from expected accelerometer behavior. The 'accelerometer ignored' flag frequently activates, indicating that during many phases of motion, the algorithm relies predominantly on gyroscope data due to unreliable accelerometer input.

4.3.2 Gait tracking using calibrated gyroscope and accelerometer data

When the calibration parameters discussed in Section 4.1 are applied to the raw gyroscope and accelerometer data, the resulting final position error is reduced to approximately 61.1 centimeters. This represents an improvement of about 2 centimeters compared to the non-calibrated case, thus reinforcing the importance and effectiveness of sensor calibration.

However, the remaining 61.1 centimeters of error may suggest two important considerations. First, the calibration performed may not be sufficiently precise, especially considering that it was manually estimated. Second, it may indicate the intrinsic limitations of inertial measurement units (IMUs) under these conditions, implying that such an error margin could be the best achievable outcome with the given hardware and environment.

The reconstructed walking trajectory, shown in Figure 4.27, closely resembles the trajectory obtained without calibration (see Figure 4.24). The main difference appears in the final straight

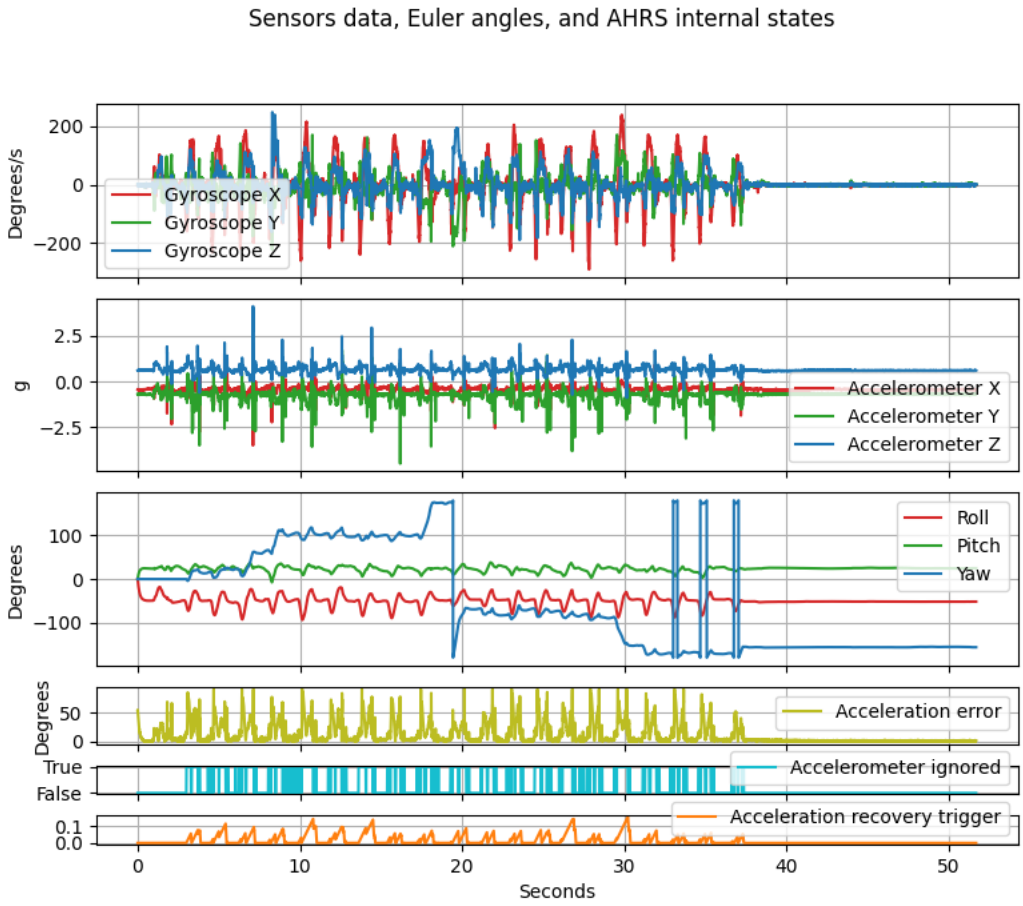

Figure 4.26: Raw sensor data, orientation angles, and status flags during the walk.

segment, where a small deviation is observed.

Figure 4.28 and Figure 4.29 illustrate the evolution of acceleration, velocity, and position over time, as well as the raw sensor measurements. The same sensor data is used as in the non-calibrated case; however, the application of calibration improves the orientation estimation and final displacement. These figures also include the Tait-Bryan orientation angles and various status flags that give insight into the filter behavior during the walking sequence.

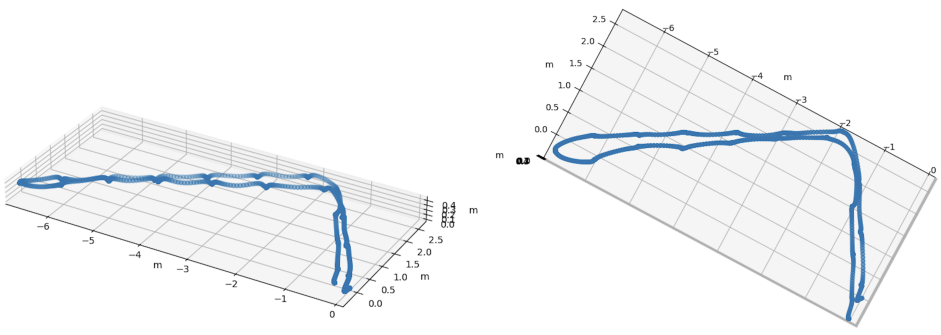

Figure 4.27: Reconstructed walking path using calibrated gyroscope and accelerometer data.

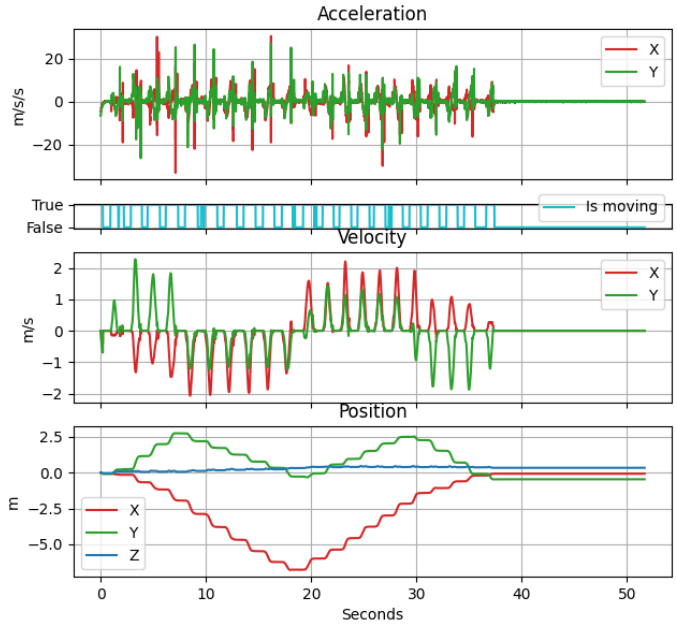

Figure 4.28: Acceleration, velocity, and position over time during the gait sequence with calibration.

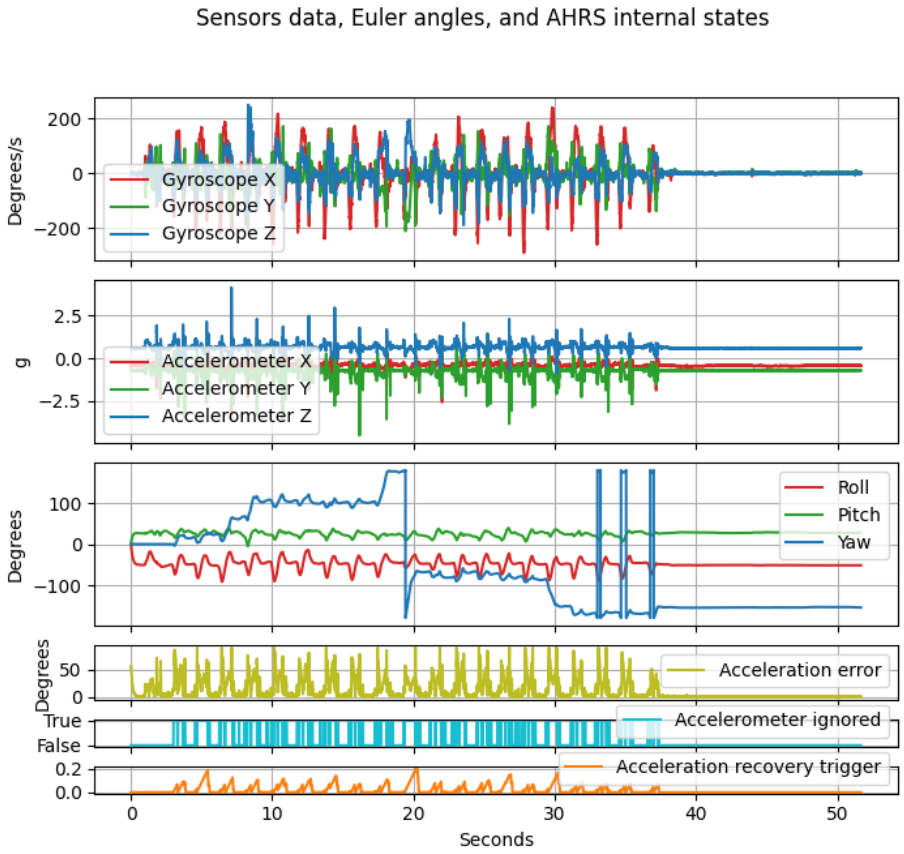

Figure 4.29: Raw sensor data, orientation angles, and status flags during the walk with calibration.

4.3.3 Gait tracking using magnetometer

In this experiment, in addition to the calibrated accelerometer and gyroscope, a calibrated magnetometer was also utilized. Based on the results observed in previous experiments, the Fusion algorithm [71] consistently provided better performance when incorporating magnetometer data. Therefore, similar improvements were expected in this case.

Following an approach closely inspired by the algorithm developed by xioTechnologies [72], a new implementation was developed that also integrates magnetometer readings. The raw values from the magnetometer, as recorded during the walking sequence, are presented in Figure 4.30.

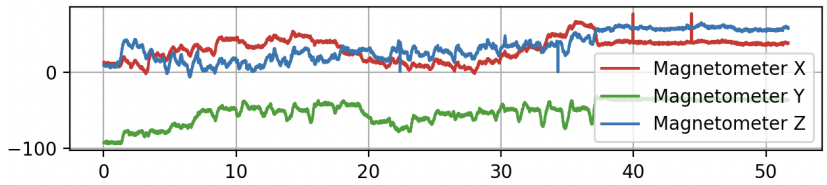

Figure 4.30: Raw magnetometer data during the walking experiment.

The reconstructed walking path is shown in Figure 4.31. While the overall trajectory still resembles the intended path, the reconstruction appears notably more curved compared to the four straight segments observed in previous experiments. The final position shows a deviation of approximately 74.1 centimeters from the expected endpoint, around 13 centimeters more than the error observed without the use of the magnetometer.

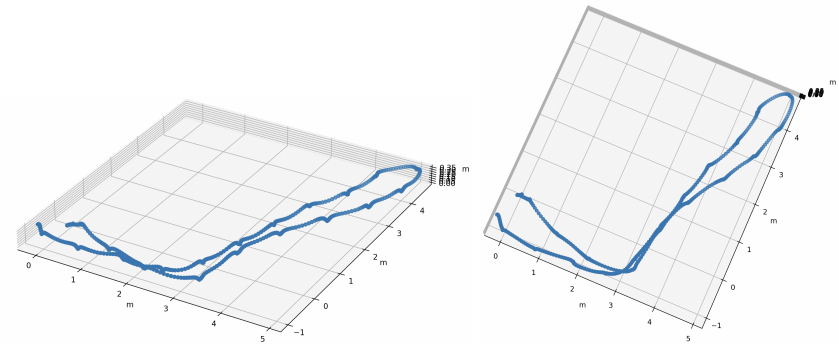

Figure 4.31: Reconstructed walking path using calibrated accelerometer, gyroscope, and magnetometer.

As shown in Figure 4.33, the pitch and roll angles are consistent with those from previous trials, but the yaw angle exhibits a substantially different trajectory. In addition, several magnetometer-related status flags are activated during the trial. Notably, the 'magnetic error' value reaches high levels, particularly after the 180° turn, and the 'magnetometer ignored' flag remains active for approximately 20 seconds thereafter.

The increased positioning error when using the magnetometer suggests that either the magnetometer itself or its calibration may be inadequate in this scenario. As discussed in Section 2.4.2, the calibrated magnetometer data should ideally form an ellipsoidal shape centered at the origin. However, the distribution of magnetometer values collected during the walk, shown in Figure 4.34, lacks the expected ellipsoidal symmetry, indicating possible magnetic interference.

To verify whether the issue was caused by sensor malfunction or environmental interference, an additional test was performed. The sensor was moved freely in space in a controlled environment, and the resulting data, shown in Figure 4.35, formed a well-defined sphere, indicating proper sensor behavior under ideal conditions.

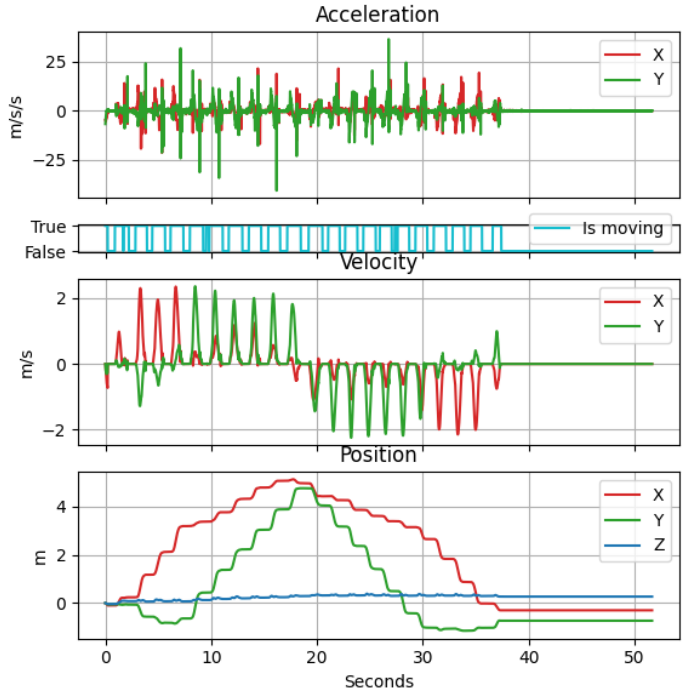

Figure 4.32: Acceleration, velocity, and position over time during the gait sequence with magnetometer.

This discrepancy suggests that the sensor might behave differently when attached to the foot compared to being held in the hand. While this behavior is counterintuitive, it could result from magnetic disturbances present near the ground or in the environment, emphasizing the sensitivity of magnetometers to external interferences.

To verify this hypothesis, an additional experiment was conducted. The sensor was pointed in the same direction at all times, while being placed in different parts of the room: on the table, on the floor, and in the air while held by hand. If there were no magnetic interferences, the magnetometer readings should remain consistent regardless of the sensor position, as its orientation was unchanged. The recorded magnetometer values for the three axes are presented in Section 4.3.3.

Sensor Position	**Mag X**	**Mag Y**	**Mag Z**
On the table	3	20	-13
Above the table leg (2 cm)	5	8	-35
Above the table leg (4 cm)	4	15	-30
Above the table leg (5 cm)	5	17	-26
Above the table leg (6 cm)	6	20	-25
On the floor	15	10	-13
On the floor (other tile)	13	4	-4
On the floor (another tile)	8	7	-15
In the air	0	27	-23
In the air (other side of the room)	1	26	-27

Table 4.2: Magnetometer readings at different sensor positions in the room.

When the sensor was placed on a random point on the table, it returned consistent values.

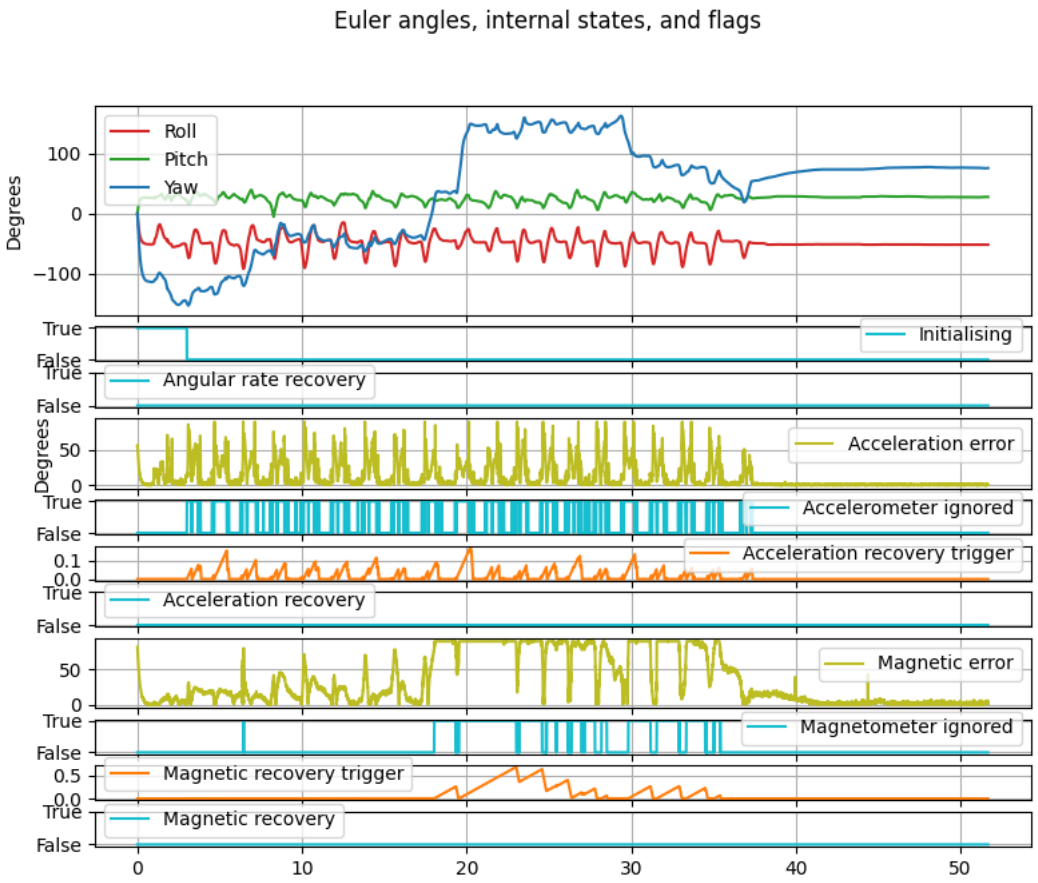

Figure 4.33: Orientation angles and status flags during the walking sequence using magnetometer data.

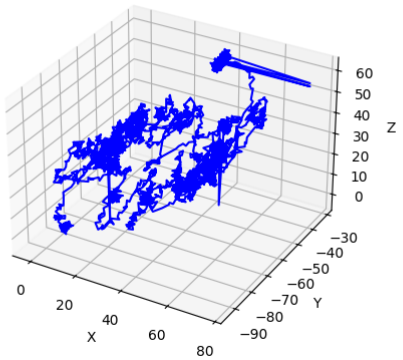

Figure 4.34: Magnetometer data represented as a 3D point cloud for the walking data set.

However, once positioned above the leg of the table, approximately 2 cm from it, which corresponds to the thickness of the table, the values changed significantly. To determine the minimum distance required to avoid interference from the leg, further measurements were taken at 4, 5, and 6 cm. The readings gradually converged to the initial measurement, where it is assumed that there was no interference.

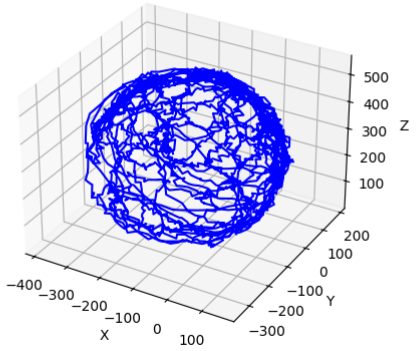

Figure 4.35: Magnetometer data for a random test sequence with arbitrary hand movements.

The data varied significantly again when the sensor was placed on the floor, suggesting interference there as well. Moving the sensor from one floor tile to another caused changes in the readings, indicating that magnetic interference varies across the room. Lastly, when the sensor was held in the air by hand, the magnetometer values differed from those on the floor or table, but remained consistent regardless of the sensor's location in the room.

In conclusion, there appear to be magnetic interferences both on the table and on the floor. This explains why the sphere shown in Figure 4.34 deviates from a true spherical shape, in contrast to the result in Figure 4.35. For accurate gait tracking using a magnetometer, it is essential to ensure that the sensor is not affected by environmental magnetic interferences.

4.3.4 Influence of calibration and use of the magnetometer for Gait tracking

Another experiment was conducted by combining the gait tracking algorithm that uses the magnetometer with the one that does not. The main challenge of this hybrid approach was determining when to rely on each version of the algorithm. The proposed solution was to discard the magnetometer data whenever its magnitude deviated significantly from the expected spherical form.

Initially, using a fixed magnitude value from the beginning led to rarely using the magnetometer based algorithm, since the magnetometer's magnitude can vary while its spherical shape is the key indicator of reliability.

A more effective approach was to periodically recalculate the expected magnitude of the magnetometer. A variable was introduced to define how much deviation from the expected magnitude would be tolerated. After testing various values, this threshold was set to 0.4, which resulted in an error of 0.564 meters, the best performance among the three previously tested algorithms.

Out of the $19,000$ samples used to evaluate the algorithm, $18,870$ used the version with the magnetometer, while only 130 used the version without it.

As seen in Figure 4.36, the reconstructed path still appears somewhat curved compared to Figure 4.24, but it ends closer to the starting point. However, Figure 4.37 and Figure 4.38 show nearly identical results when compared to the same plots from previous algorithm versions.

However, this method required extensive preprocessing, including computing the mode of the magnetometer's module every $1,000$ samples and tuning the discard threshold. Therefore, while effective in this case, it cannot be considered a generalizable solution; for different data sets, the

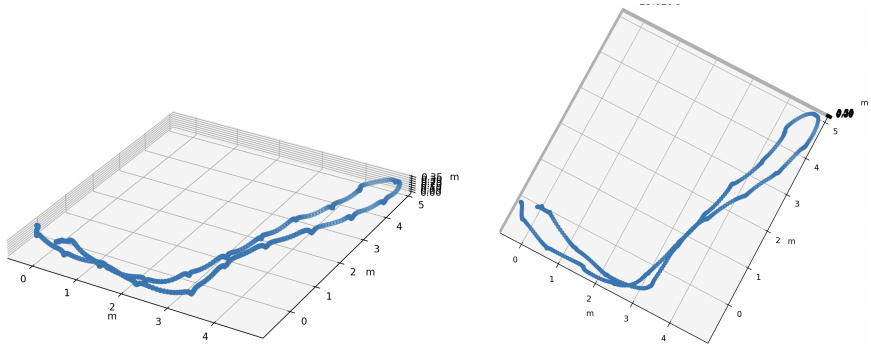

Figure 4.36: Reconstructed walking path using calibrated accelerometer, gyroscope, and selectively the magnetometer.

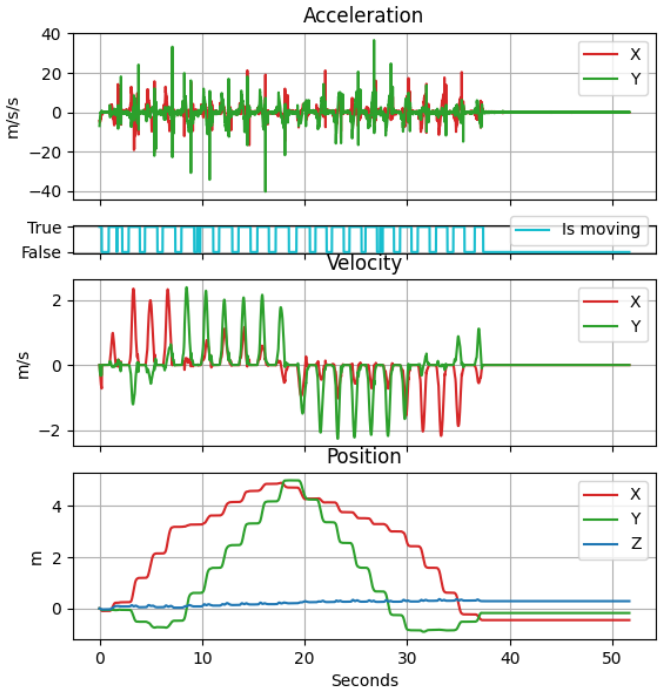

Figure 4.37: Acceleration, velocity, and position over time during the gait sequence with conditional magnetometer usage.

optimal threshold may differ from 0.4.

A promising direction for future improvement would be to periodically and automatically estimate the radius of the magnetometer sphere. Despite variations in the magnetic field's magnitude, as long as its shape remains consistent and the magnitude is accurately tracked, the magnetometer can continue to provide reliable orientation data.

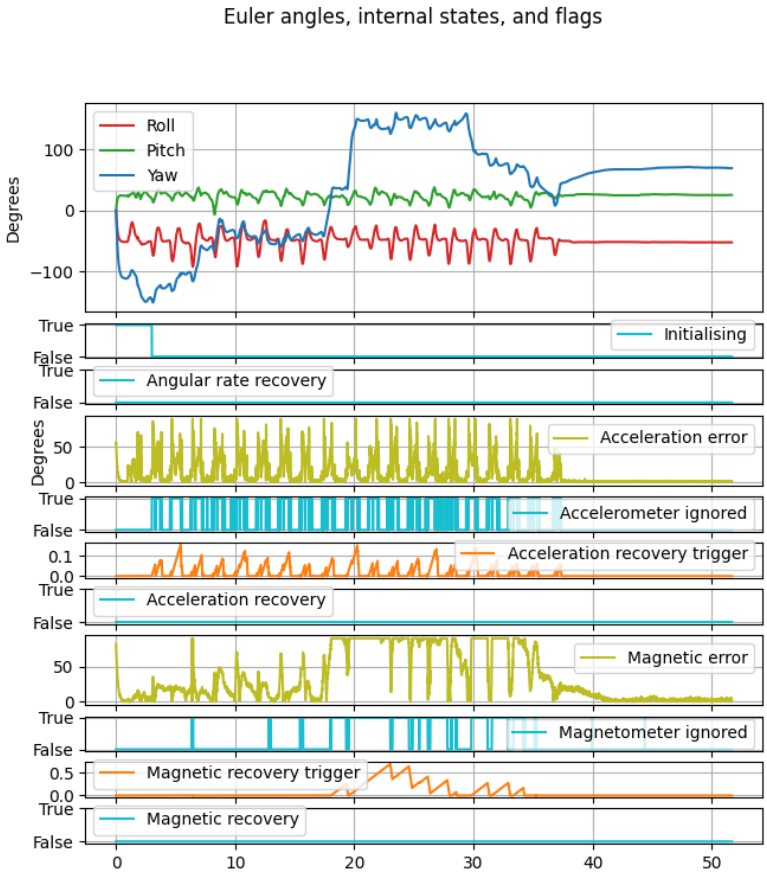

Figure 4.38: Orientation angles and status flags during the walking sequence, selectively using magnetometer data.

Chapter 5

Power Consumption

Another key aspect explored in this project was the power consumption of the IMU. Understanding how different data acquisition methods affect power usage is crucial, especially in applications where battery life is a critical constraint. Although the differences may appear minimal in small-scale experiments, in real-world scenarios, where extended operation is necessary, these differences can significantly impact system performance and usability.

This chapter studies the differences of power consumption between the magnetometer, accelerometer and gyroscope, but also compares the advantages and disadvantages of each data acquisition method with a focus on their power consumption characteristics.

Various implementations were considered, as mentioned in Section 3.2.2, including: (1) a polling-based method that continuously checks for new data, (2) an interrupt-driven method that reacts only when data is available, and (3) the use of the DMP (Digital Motion Processor) for onboard computation.

5.1 Hardware and technologies

Data acquisition from the IMU was performed using the SparkFun `ICM_20948` library [16], as detailed in Section 3.2.

For an accurate evaluation of power consumption, it is important to account for the fact that the development board includes an onboard LED, as shown in Figure 3.1, which continuously draws approximately 1 mA. As a result, even when the IMU and other components are deactivated, the baseline power consumption remains around 1 mA due to the LED alone. All of this will be addressed in Section 5.2.

Power consumption measurements were conducted using the Power Profiler Kit II, shown in Figure 5.1. This affordable and versatile tool provides hardware and software developers with a simple and effective means of measuring both average and dynamic power consumption in embedded systems. The measurements were acquired using the nRF Desktop application.

5.2 Power consumption of the accelerometer, gyroscope, and magnetometer

To gain a better understanding of the power consumption of each component in the ICM-20948 IMU (including the onboard LED, accelerometer, gyroscope, and magnetometer), a set of experiments was conducted.

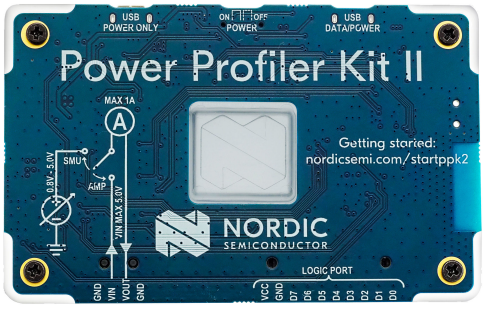

Figure 5.1: Power Profiler Kit II [40].

The accelerometer and gyroscope were individually enabled and disabled under different configurations using the modes `ICM_20948_Sample_Mode_Continuous` and `ICM_20948_Sample_Mode_Cycled`, and the magnetometer was only turned on and off. A total of eight scenarios were evaluated to determine the relative power draw of each component.

For simplicity, the sampling frequency was fixed at $562.5Hz$ during all experiments, but due to the limitations it represents to $400Hz$.

The following variables represent the individual power contributions:

- C_{acont}: Accelerometer in continuous mode.
- C_{acycl}: Accelerometer in cycled mode.
- C_{acontd}: Accelerometer in continuous mode (idle/delay).
- C_{acycld}: Accelerometer in cycled mode (idle/delay).
- C_{gcont}: Gyroscope in continuous mode.
- C_{gcycl}: Gyroscope in cycled mode.
- C_{gcontd}: Gyroscope in continuous mode (idle/delay).
- C_{gcycld}: Gyroscope in cycled mode (idle/delay).
- C_{led}: Power consumption of the LED.
- C_{mag}: Magnetometer during active data extraction.
- C_{magd}: Magnetometer in delay mode.
- C_{magon}: Magnetometer powered on but not extracting data.
- C_{sleep}: Consumption of the board when all its components are in sleep mode. Excludes the LED.

These variables led to the following system of approximate equations (all values in mA):

$$
\begin{cases}
C_{acont} + C_{gcont} + C_{mag} + C_{led} = 7.83 \\
C_{acontd} + C_{gcontd} + C_{magd} + C_{led} = 6.39 \\
C_{acont} + C_{gcycl} + C_{mag} + C_{led} = 7.31 \\
C_{acontd} + C_{gcycld} + C_{magd} + C_{led} = 5.79 \\
C_{acycl} + C_{gcont} + C_{mag} + C_{led} = 7.72 \\
C_{acycld} + C_{gcontd} + C_{magd} + C_{led} = 6.27 \\
C_{acycl} + C_{gcycl} + C_{mag} + C_{led} = 7.18 \\
C_{acycld} + C_{gcycld} + C_{magd} + C_{led} = 5.90 \\
C_{acont} + C_{gcont} + C_{led} = 7.04 \\
C_{acontd} + C_{gcontd} + C_{led} = 5.58 \\
C_{acont} + C_{gcycl} + C_{led} = 6.54 \\
C_{acontd} + C_{gcycld} + C_{led} = 5.08 \\
C_{acycl} + C_{gcont} + C_{led} = 6.96 \\
C_{acycld} + C_{gcontd} + C_{led} = 5.46 \\
C_{acycl} + C_{gcycl} + C_{led} = 6.41 \\
C_{acycld} + C_{gcycld} + C_{led} = 4.94 \\
C_{sleep} + C_{led} = 0.979 \\
C_{magon} + C_{led} = 2.17
\end{cases}
\tag{5.1}
$$

After completing all experiments, the LED was removed from the board to eliminate its power consumption. Following this modification, a sleep current (C_{sleep}) of 43 μA was measured. This value includes approximately 8 μA drawn by the IMU, 10 μA by the voltage regulator, with the remaining current attributed to losses from voltage level conversions.

Substituting this value directly in Equation (5.1) simplifies the system. Then, using substitution and Gaussian elimination, the following estimated values were obtained:

$$
\begin{aligned}
C_{sleep} &\approx 0.043 \\
C_{led} &\approx 0.936 \\
C_{magon} &\approx 1.191 \\
C_{mag} &\approx 1.051 \\
C_{magd} &\approx 0.761 \\
C_{acont} &\approx 1.312 \\
C_{acontd} &\approx 0.928 \\
C_{acycl} &\approx 1.202 \\
C_{acycld} &\approx 0.829 \\
C_{gcont} &\approx 4.518 \\
C_{gcontd} &\approx 3.473 \\
C_{gcycl} &\approx 3.558 \\
C_{gcycld} &\approx 2.622
\end{aligned}
\tag{5.2}
$$

5.2.1 Comparison with datasheet values

The ICM-20948 datasheet [50] provides reference power consumption values for various operating modes of the IMU. These values are summarized in Table 5.1. In this section, the datasheet values are compared with the experimental results obtained in this work.

Parameter	Conditions	Value (mA)
9-axis	No DMP, Continuous Mode, Low noise mode	3.11
Gyroscope only	No DMP, Low power mode, 102.3 Hz	1.23
Accelerometer only	No DMP, Low power mode, 102.3 Hz	0.068
Magnetometer only	8 Hz	0.09
Full chip sleep mode	—	0.008

Table 5.1: Power consumption values reported in the ICM-20948 datasheet [50].

To enable a fair comparison, the experimental power consumption values from Equation (5.2) were scaled to match the operating frequencies listed in Table 5.1. This adjustment assumes that power consumption is directly proportional to sampling frequency.

In the experimental setup, the gyroscope and accelerometer were originally configured for a sampling frequency of 562.5 Hz, and the magnetometer for 100 Hz. The values presented in Table 5.2 have been adjusted to reflect operation at 102.3 Hz for the gyroscope and accelerometer, and 8 Hz for the magnetometer, in alignment with the datasheet conditions.

Parameter	Conditions	Value (mA)
9-axis	No DMP, Continuous Mode, Low noise mode	1.2421
Gyroscope only	No DMP, Low power mode, 102.3 Hz	0.82167
Accelerometer only	No DMP, Low power mode, 102.3 Hz	0.2386
Magnetometer only	8 Hz	0.08408
Full chip sleep mode	—	< 0.033

Table 5.2: Adjusted experimental power consumption values for comparison.

It is important to note that several assumptions were made during this comparison, particularly regarding operating modes and configurations not explicitly detailed in the datasheet. As a result, the experimental values, especially those for the 9-axis, gyroscope, and accelerometer, should be interpreted with caution.

Despite these uncertainties, the experimental value obtained for the magnetometer is notably close to the datasheet specification, indicating a high degree of accuracy in that measurement. In contrast, the other measurements show significant deviations, likely due to differences in mode configurations, component tolerances, and estimation errors in the system-level power modeling presented in Equation (5.1).

5.3 Power consumption of the polling approach

This method relies on a continuous loop that checks whether the IMU has new data available. Different tests were carried out by varying the sampling frequency and toggling between normal, low power, and sleep modes.

The test cycle was as follows:

- Collect data at a high frequency of 400 Hz (maximum possible in this mode) for 5 seconds.
- Wait a 5-second delay.
- Switch to a low frequency of 20 Hz and collect data for another 5 seconds.
- Enable sleep mode for 5 seconds. (Note: the default sleep mode does not disable the magnetometer, so it had to be manually turned off.)

- Resume operation with the magnetometer active and activate low power mode, again collecting data at 20 Hz for 5 seconds.

- Introduce a 5-second delay before restarting the cycle.

This setup allows a detailed comparison of how various configurations affect power consumption over time.

As illustrated in Table 5.3, the polling method, using the mode `ICM_20948_Sample_Mode_Continuous`, yields the following results: during high-frequency data collection (400 Hz), the system consumes approximately 7.73 mA. Surprisingly, at a lower frequency (20 Hz), the current draw increases to 8.73 mA. This counterintuitive behavior could be attributed to the increased internal processing time for sensor filtering when operating at lower sampling rates, particularly for the gyroscope and accelerometer.

When the IMU is placed into sleep mode and the magnetometer is manually disabled, the consumption drops to 0.984 mA, a value consistent with the LED-only power draw. If sleep mode is activated but the magnetometer remains active, the system consumes approximately 2.17 mA, implying that the magnetometer alone accounts for around 1.183 mA of current draw.

Enabling the IMU's low power mode, however, did not yield a measurable reduction in power consumption compared to normal operation. This suggests that simply toggling this mode might not be sufficient; further investigation is needed to determine whether the sensor is already operating near its lowest power state by default, or whether additional configurations are required.

During delay intervals, when no data collection is taking place, the system consumes approximately 6.36 mA, which falls between the active sampling modes and the full sleep state, as expected.

Approach	**Frequency (Hz)**	**Mode**	**Consumption (mA)**
Polling	400	Continuous	7.73
Polling	20	Continuous	8.73
Polling	-	Sleep mode	0.984
Polling	20	Low power mode	8.74
Polling	-	Delay	6.36

Table 5.3: Schematic timeline of the cycle during continuous mode with its corresponding power consumption. Each phase lasts 5 seconds.

If both the accelerometer and gyroscope are configured to the `ICM_20948_Sample_Mode_Cycled` mode, the resulting power consumption values differ, as illustrated in Table 5.4.

Approach	**Frequency (Hz)**	**Mode**	**Consumption (mA)**
Polling	400	Cycled	7.13
Polling	20	Cycled	6.93
Polling	-	Sleep mode	0.974
Polling	20	Low power mode	6.84
Polling	-	Delay	5.78

Table 5.4: Schematic timeline of the cycle during cycled mode with its corresponding power consumption. Each phase lasts 5 seconds.

As shown in Table 5.4, the power consumption values observed in cycled mode are consistently lower than those recorded in continuous mode, discussed previously in Section 5.2. Although the timeline structure remains similar to that of the continuous mode, the lower frequency settings

yield reduced power consumption, as expected. The difference between low power mode and low frequency operation in cycled mode is minimal but greater than in the continuous mode, only about 0.09 mA, which reinforces the conclusion that low power mode has a negligible impact in this specific configuration.

5.4 Power consumption of the interrupt-driven approach

This approach also uses raw data acquisition, but instead of constantly checking for data availability, the system sleeps until it is awakened by an interrupt signal from the IMU. This can significantly reduce power usage, especially when data is not required at high rates.

The same test was conducted using both high-frequency (400 Hz) and low-frequency (20 Hz) settings to compare the impact of sample rate on power savings.

In this method, the system remains in a low-power state until awakened by an interrupt signal indicating that new data is available. The measured power consumption during sleep and delay phases was consistent with the polling approach results described in Section 5.3. However, differences were observed during active data acquisition:

When operating in continuous mode, as seen in Table 5.5, at a high sampling rate of 400 Hz, the system consumed approximately 7.05 mA, which is about 0.68 mA less than the consumption observed with the polling method. At a lower sampling rate of 20 Hz, still in continuous mode, the power consumption stayed in 8.75 mA as seen in Table 5.3.

Approach	**Frequency (Hz)**	**Mode**	**Consumption (mA)**
Interruptions	400	Continuous	7.05
Interruptions	20	Continuous	8.75
Interruptions	-	Sleep mode	0.984
Interruptions	-	Delay	6.36

Table 5.5: Power consumption of the interruption-driven approach during continuous mode.

In cycled mode with a high sampling rate of 400 Hz, the system consumed around 6.36 mA, approximately 0.77 mA less than when using polling. Notably, this value corresponds to the current drawn in delay mode in the continuous mode, suggesting that the system remains in a quasi-idle state between interrupts. At the lower frequency of 20 Hz in cycled mode, power consumption didn't decrease and stayed at 6.94 mA, which is identical to the value observed without using interrupts in Table 5.4. All the values commented can be seen in Table 5.6.

Approach	**Frequency (Hz)**	**Mode**	**Consumption (mA)**
Interruptions	400	Cycled	6.36
Interruptions	20	Cycled	6.94
Interruptions	-	Sleep mode	0.974
Interruptions	-	Delay	5.78

Table 5.6: Power consumption of the interruption-driven approach during cycled mode.

These observations lead to the conclusion that interrupt-driven data acquisition is effective for reducing power consumption at high sampling rates, but it does not provide any significant benefit at lower frequencies.

5.5 Power consumption of the DMP-based approach

The final method utilized the IMU's Digital Motion Processor (DMP), a built-in module that processes sensor data internally and outputs quaternions and other computed values. While this offloads computation from the host processor, it also changes the power profile of the IMU.

The maximum frequency supported in this mode is 60 Hz, which was used as the high-frequency configuration. A second test at a low frequency of 2 Hz was also conducted. Additional experiments were performed by toggling the DMP on and off to observe the corresponding changes in power consumption.

This method leverages the Digital Motion Processor (DMP) embedded in the IMU, which computes higher-level data such as quaternions onboard. Three configurations were tested:

1. DMP enabled at its maximum supported frequency of 60 Hz.
2. DMP enabled at a lower frequency of 2 Hz.
3. DMP disabled (sensor powered on, but without onboard processing).

As seen in Table 5.7, with the DMP disabled, the system drew approximately 4.93 mA. Enabling the DMP and running it at 60 Hz increased the consumption slightly to 5.37 mA, while the 2 Hz configuration resulted in 5.15 mA. These results show that enabling the DMP adds only a modest increase in power usage compared to raw data acquisition.

Approach	**Frequency (Hz)**	**Mode**	**Consumption (mA)**
DMP	60	-	5.37
DMP	2	-	5.15
DMP	-	DMP off	4.93

Table 5.7: Power consumption of the DMP approach.

From a performance perspective, the DMP provides not only raw sensor data but also processed quaternions, which have previously been shown to closely approximate Tait-Bryan angles. In contrast, the polling and interrupt-based methods only output raw data, requiring additional onboard computation to derive orientation; this would add to the total power consumption. .

5.6 Evaluation and interpretations

From the results shown in Equation (5.2), the following interpretations can be made:

- The gyroscope is the most power-hungry component, especially in continuous mode.
- Switching to cycled mode reduces power consumption by approximately 20–25%.
- The accelerometer consumes significantly less than the gyroscope.
- The magnetometer, even when active, has a relatively low power footprint.
- Power consumption is measurably lower when sensors are not reading data.

Based on the results shown in Table 5.8, several conclusions can be drawn regarding the power consumption of the IMU under different operating modes and frequencies:

Approach	Frequency (Hz)	Mode	Consumption (mA)
Polling	400	Continuous	7.73
Interruptions	400	Continuous	7.05
Polling & Interruptions	20	Continuous	8.73
Polling	400	Cycled	7.13
Interruptions	400	Cycled	6.36
Polling & Interruptions	20	Cycled	6.94
Polling	20	Low power + Continuous	8.73
Polling	20	Low power + Cycled	6.84
Polling & Interruptions	-	Delay + Continuous	6.36
Polling & Interruptions	-	Delay + Cycled	5.78
Polling & Interruptions	-	Sleep mode	0.984
DMP	60	-	5.37
DMP	2	-	5.15
DMP	-	DMP off	4.93

Table 5.8: Results evaluating the consumption of the IMU in different modes. All values include the LED value which is 1mA approximately.

- Polling and interruption-driven modes are inefficient at low frequencies. This suggests that lower frequencies in these modes do not necessarily lead to lower consumption and may keep internal circuits, or filters, active for longer durations.
- Low power mode has no effect in polling at any frequency. The low power setting may not be effective or impactful in this context.
- Delay mode reduces consumption in general.
- Sleep mode is about 0,984 mA, which corresponds to the LED.
- Interrupt-based acquisition is more efficient than polling.
- DMP usage results in consistent savings. This shows that using the DMP provides stable and efficient consumption across different sampling frequencies.

In summary, if the application does not demand very high sampling rates and can function effectively at 60 Hz, the DMP-based method offers an efficient balance. It delivers high-level orientation data while consuming less power overall, making it suitable for power-sensitive applications that still require reliable motion tracking.

The most effective strategy for reducing IMU power consumption includes transitioning to sleep or delay modes during idle periods.

In the case of this experiment, such as the gait tracking scenario discussed in Section 4.3, a high sampling frequency was required. Therefore, it was preferable not to use the Digital Motion Processor (DMP), as it is limited to a maximum frequency of 60 Hz. Among the available high-frequency, no-DMP configurations, the best result was achieved using the interrupt-driven approach in cycled mode, yielding a power consumption of 6.36 mA.

Although the configuration using the magnetometer yielded slightly better numerical results, the reconstructed trajectory appeared more realistic when using only the accelerometer and

gyroscope. Consequently, the optimal configuration excludes the magnetometer. Additionally, the onboard LED can be removed, as it is not necessary for the application.

Taking this into account, the estimated total power consumption becomes:

$$6.36 - C_{\text{mag}} - C_{\text{led}} \approx 4.373\,\text{mA} \tag{5.3}$$

Assuming the use of a 100 mAh battery, the estimated system uptime can be calculated using the following formula:

$$\text{Time (hours)} = \frac{\text{Battery capacity (mAh)}}{\text{Power consumption (mA)}} \tag{5.4}$$

Substituting the values:

$$\text{Time} = \frac{100}{4.373} \approx 22.87\,\text{hours} \tag{5.5}$$

Thus, under these conditions, the system could operate continuously for approximately 22.87 hours.

Chapter 6

Conclusions and Future Work

To summarize, Inertial Measurement Units (IMUs) are devices capable of capturing motion-related data. They typically provide three types of measurements: magnetic field, acceleration, and angular velocity (gyroscope), each along three axes and at high sampling rates.

To obtain useful and meaningful data, raw measurements must be processed. First, orientation must be computed using appropriate algorithms, followed by the integration of this information to estimate the absolute position. This process is ideally done in real-time to provide responsive feedback to the user. A common challenge is that many of these algorithms are embedded within micro controllers, turning the system into a "black box".

The main objective of this project was not only to estimate orientation, but more importantly, to evaluate whether the designed "black box" system is reliable, flexible, and suitable for real-world applications. A key focus was developing robust and adaptable algorithms capable of being tuned for different use cases, such as gait tracking. This required verifying the quality of the raw data acquisition, assessing various sampling and power modes, and determining the optimal configuration that balances performance and energy efficiency.

Through a series of experiments, multiple sensor configurations were tested and evaluated their impact on power consumption.

Conclusions

Regarding the orientation estimation algorithms, the Complementary Filter proved to be intuitive and easy to implement, producing smooth and reasonably accurate results with minimal noise. The Kalman Filter, in contrast, performed the worst among the tested algorithms. It introduced considerable noise and tended to lose track during large movements. The Fusion algorithm delivered the most reliable results, closely matching the expected orientation even under dynamic conditions. Also, in some specific cases (e.g., when the sensor was stationary point to the geographic North), it gave better results than the quaternion-based algorithm.

Four different versions of the Fusion algorithm were evaluated: using the magnetometer with calibration, not using the magnetometer with and without calibration, and a hybrid version that selectively uses the magnetometer based on its module. As anticipated, incorporating the magnetometer generally led to improved absolute positioning accuracy.

However, when comparing the calibrated and uncalibrated versions, the results were inconsistent — in some cases, calibration led to improvements, while in others, it made little difference or even introduced more drift. Therefore, only the calibration parameters that improved the algorithm were introduced, and no definitive advantage could be attributed solely to the presence or absence of calibration.

It can be concluded that while the algorithms using the magnetometer tend to provide more accurate final position estimations (absolute performance), the versions that exclude the magnetometer often result in walking paths that appear more natural and realistic (relative performance), closely resembling the actual trajectory followed by the subject.

In terms of calibration, the gyroscope offset was the simplest and most effective parameter to correct, significantly improving the results. Magnetometer calibration had little impact on roll and pitch angles, but it greatly improved yaw estimation. Calibration was performed by manually rotating the sensor in all directions to approximate a spherical trajectory. Although the same method was attempted for accelerometer calibration, the results were not as accurate. In some cases, uncalibrated data yielded better performance, highlighting the importance of precise calibration. Manual calibration can introduce errors, and proper calibration should ideally be performed using specialized machinery.

The experiments also revealed that the calibration results depended heavily on the environment. For example, measurements taken near metallic structures or magnetic sources (like table legs or floors) showed significant deviations. This further emphasizes the sensitivity of magnetometers and the importance of an interference-free environment during calibration.

As for gait tracking, the results showed that it is generally more reliable without magnetometer data, as floors may introduce magnetic interference. Although calibration improved the results, achieving optimal accuracy requires high-precision calibration tools.

Future Work

IMUs, orientation algorithms, and gait tracking remain active areas of research, with numerous unexplored opportunities beyond the scope of this project. Some directions for future exploration include the following.

- Enhancing calibration accuracy using professional equipment, such as gimbals, to evaluate the improvement in algorithm performance.
- Developing algorithms capable of real-time self-calibration to maintain accuracy during motion.
- Extending gait tracking to full-body motion capture using multiple sensors placed across different body parts to reconstruct complete human movement.
- Improving energy efficiency or reducing the cost of sensor systems to enable other applications.

Chapter 7

Annex

All the code used in the evaluation of this project is available on GitHub at `https://github.com/luciaalonsomozo/IMUAlgorithms` [39]. The repository is organized into folders, each of which is explained below for better clarity.

7.1 ICM_20948

The `ICM_20948` folder contains C code used to program the sensor and microcontroller. It consists of two main components: one for data extraction and another for measuring power consumption. The latter version is similar but excludes IMU-CPU communication and UART transmission.

- `DMP:` This code configures the sensor to use the DMP (Digital Motion Processor). It outputs raw sensor data and quaternions in the following format:

$$Time, AccX, AccY, AccZ, GyrX, GyrY, GyrZ, MagX, MagY, MagZ, Qw, Qx, Qy, Qz \tag{7.1}$$

  Units: seconds for time, $8192 \cdot g$ for the accelerometer, $16.4 \cdot$ degrees/s for the gyroscope, and $0.15 \cdot \mu T$ for the magnetometer. The last four components represent the quaternion as $q = (q_w, q_x, q_y, q_z)$.

- `Simple_angles:` This code directly configures the sensor without using the DMP. It outputs only raw sensor data in the format:

$$Time, AccX, AccY, AccZ, GyrX, GyrY, GyrZ, MagX, MagY, MagZ \tag{7.2}$$

  Units: seconds for time, mg for the accelerometer, dps (degrees per second) for the gyroscope, and μT for the magnetometer.

- `Interruptions:` This is a modified version of `Simple_angles`, using hardware interrupts so the microcontroller only wakes when data is available. The output format and units remain the same as Equation (7.2). This code was used to measure the power consumption of this approach.

- `Power_consumption_simple_angles`: This follows the same approach as `Simple_angles`, but it changes the frequency, it goes on sleep mode, on low power mode and on delay mode to test the power consumption of each mode.

- `Power_consumption_DMP:` This code is very similar to `DMP` but has some more extra code lines for turning on and off the DMP. This code was used to measure the power consumption of the DMP.

- `Power_consumption_mag_gyr_acc`: This follows the same approach as `Power_consumption_simple_angles`, but it was specially created not to measure the power consumption of the whole system, but for the gyroscope, accelerometer and magnetometer separately.

The sampling frequency is configured based on the formula:

$$value = \frac{Running\_sample}{ODR} - 1 \tag{7.3}$$

Different values of `value` are used depending on whether `DMP` or `Simple_angles` is employed.

7.2 Readings_to_csv

This part of the project handles data storage. Once the IMU is programmed, data is saved in CSV format for easy processing. All related scripts are contained in the `readings_to_csv` folder.

There are two main Python scripts:

- `reading_data.py`: Saves data collected using `Simple_angles`. It captures 19,000 samples (around 52 seconds at 400Hz), generating five files:
  - `data/acc3_raw.csv`: Accelerometer data on x, y, and z axes.
  - `data/gyr3_raw.csv`: Gyroscope data on x, y, and z axes.
  - `data/mag3_raw.csv`: Magnetometer data on x, y, and z axes.
  - `data/data.csv`: Consolidated data in the format shown in Equation 7.2.
- `reading_data_dmp.py`: Saves data collected using `DMP`. It captures 3,000 samples (also 52 seconds at 60Hz), producing the same files as above, with an additional:
  - `data/quaternions.csv`: Contains time and quaternion data in the format:

$$Time, Qw, Qx, Qy, Qz \tag{7.4}$$

  The file `data/data.csv` now follows the format in Equation 7.1, as it includes quaternion data.

7.3 Filters

The `filters` folder includes algorithms for computing Tait-Bryan angles from the raw sensor data stored in the CSV files.

- `filters/Fusion-main`: Implements the AHRS algorithm using the Fusion library, located in `Fusion-main/Fusion`. The `Examples` subfolder includes three variants:
  - `Examples/Simple`: Uses only accelerometer and gyroscope data, without calibration. It generates the results as `fusion_simple_results.csv`.
  - `Examples/Advanced`: Incorporates magnetometer data. Results are saved in `fusion_advanced_nocalibration.csv`.
  - `Examples/AdvancedCalibration`: Adds calibration to the previous version. The results are saved in `fusion_advanced_calibration.csv`.

- `filters/complementary_filter.py`: Reads raw data from `data.csv` and computes Tait-Bryan angles using a complementary filter with $\alpha = 0.02$. It generates the results in `complementary_filter_results.csv`.
- `filters/euler_direct.py`: Reads quaternion data from `quaternions.csv` and computes Tait-Bryan angles using the `panda3d.core` module. Results are saved in `euler_direct_results.csv`.
- `filters/kalman_filter.py`: Computes Tait-Bryan angles from raw sensor data in `data.csv` using a Kalman filter approach. Results are in `kalman_filter_results.csv`.

The files `fusion_simple_results.csv`, `complementary_filter_results.csv` and `kalman_filter_results.csv` have the following format:

$$Time, Phi(degrees), Theta(degrees) \tag{7.5}$$

However, `fusion_advanced_nocalibration.csv`, `fusion_advanced_calibration.csv` and `euler_direct_results.csv` are as follows:

$$Time, Phi(degrees), Theta(degrees), Yaw(degrees) \tag{7.6}$$

7.4 Calibration

This section refers to the `calibration` folder, which contains algorithms for estimating offsets and calibration parameters.

- `calibration/calibrate_gyr.py`: This script reads gyroscope data from `gyr3_raw.csv` and computes the mean, which serves as the gyroscope offset. It should be executed using data recorded while the sensor is stationary, specifically `data/still/gyr3_raw.csv`.
- `calibration/calibrate_mag_acc.py`: This script estimates the offset and misalignment parameters for both the magnetometer and the accelerometer. For the accelerometer, the datasets `data/calibration/acc3_raw.csv` and `data/calibration_30000_samples/acc3_filtered.csv` can be used. For the magnetometer, use `data/calibration/mag3_raw.csv`. The parameter $MField$ should be set depending on the sensor and data source:
  - 8192 for the accelerometer,
  - 250 for the magnetometer using DMP,
  - 39.4784 for the magnetometer without DMP.
- `calibration/reduction_of_points.py`: This script reduces the original 30,000 accelerometer samples from `data/calibration_30000_samples/acc3_raw.csv` to 3,000 representative points. It filters out regions of high density to ensure a uniform distribution over the sphere. The reduced dataset is saved to `data/calibration_30000_samples/acc3_filtered.csv`.
- `calibration/calibrate_euler.py`: This script calculates the Tait-Bryan angle offsets while the sensor is stationary.

It is important to note that if a new data set is recorded, all the calibration parameters will be different and need to be recalculated. The steps to record and compute the new calibration parameters are:

1. Program the IMU.
2. Run `readings_to_csv/reading_data.py` or `readings_to_csv/reading_data_dmp.py` (depending on how the IMU was programmed) while moving the sensor through all possible orientations.
3. Set `MField` to `8192`, use the newly recorded accelerometer data file, and run `calibration/calibrate_mag_acc.py`. Save the returned values.
4. Change `MField` to the correct value for the magnetometer (either `250` with DMP or `39.4784` without DMP), set the input file to the magnetometer data, and run `calibration/calibrate_mag_acc.py` again. Save the returned values.
5. Record data with the sensor completely still using `readings_to_csv/reading_data.py` or `readings_to_csv/reading_data_dmp.py`.
6. Open `calibration/calibrate_gyr.py`, update the file path to the newly recorded gyroscope data, and save the resulting offset values.
7. Update all relevant files and algorithms with the new calibration parameters.

7.5 Data

This section corresponds to the `data` folder, which contains all datasets referenced in Section 4.2, as well as the datasets used for calibration and gait tracking experiments.

- `data/calibration`: Contains datasets used to calibrate the accelerometer and magnetometer. The sensor was moved through various orientations to capture values across the full range of motion (i.e., covering the sphere).
- `data/calibration_30000_samples`: Similar to `data/calibration`, but includes 30,000 samples, which are later reduced to 3,000 samples for calibration purposes.
- `data/fast_movement`: Contains data captured during fast sensor movements.
- `data/infinite`: Contains data recorded while the sensor moved in a horizontal figure-eight (infinity symbol) pattern.
- `data/pendulum`: Contains data from pendulum-like swinging motion of the sensor.
- `data/slow_movement`: Contains data from slow movements of the sensor.
- `data/still`: Contains data recorded while the sensor was stationary. This dataset is also used for gyroscope calibration.
- `data/walking`: Contains walking motion data where the sensor is attached to the foot, but the user does not return to the starting position.
- `data/walking_gait_tracking`: Contains walking motion data where the sensor is attached to the foot, and the user returns to the starting point. This dataset is specifically used for gait tracking experiments.

All datasets were recorded using the `DMP`, except for `data/walking_gait_tracking`, which was recorded using `Simple_angles`.

7.6 Results

This section corresponds to the `results` folder, which contains a subfolder for each dataset in the `data` folder, excluding the gait tracking and calibration datasets.

Each file within these subfolders contains the results of applying the filters described in Section 7.3, and they follow the formats outlined in Equation (7.5) and Equation (7.6).

7.7 Util

This section corresponds to the `util` folder, which contains utility scripts used throughout the project.

- `util/best_alpha_compl_filter.py`: Executes the complementary filter algorithm (as in `filters/complementary_filter.py`) for various values of α: $0.02, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98$. It then computes the error in the roll and pitch angles and returns the optimal α for each.
- `util/comparison.py`: After the results are stored in `results/`, this script generates time-series plots of the roll and pitch angles. Algorithm colors are defined in Figure 4.1.
- `util/frequency_mean_mode.py`: Given a file like `data.csv`, this script calculates the mean and mode of the sampling frequency.
- `util/norm_mean_mode.py`: Given raw sensor files such as `acc3_raw.csv`, `gyr3_raw.csv`, or `mag3_raw.csv`, this script computes the mean and mode of the magnitude across all three axes.
- `util/compare_errors.py`: This script reads the output results from all the different algorithms and computes the corresponding pitch and roll errors, and its mean.

7.8 Plot

This section refers to the `plot` folder, primarily used to generate visual examples for this report. The key scripts are:

- `plot/plot_raw.py`: Given a sensor file in three-axis format (e.g., `acc3_raw.csv`, `gyr3_raw.csv`, or `mag3_raw.csv`), this script generates 2D plots for each axis.
- `plot/plot_3d.py`: Similar to `plot_raw.py`, but generates 3D plots of the raw sensor data.

7.9 Gait Tracking

The `gait_tracking` folder contains four implementations of the gait tracking algorithm:

- `gait_tracking/gait_tracking.py`: The basic version of the gait tracking algorithm, using only uncalibrated accelerometer and gyroscope data, based on the approach from [72].
- `gait_tracking/calibrated_gait_tracking.py`: An enhanced version of the basic algorithm, incorporating accelerometer and gyroscope calibration.

- `gait_tracking/magnetometer_gait_tracking.py`: Builds upon the calibrated version by including magnetometer data and its calibration.
- `gait_tracking/combined_gait_tracking.py`: Combines the logic of the previous two versions, using the magnetometer conditionally based the magnetic field values. Two important variables control this behavior:
  - *magneticPercentage*: Defines the allowable deviation from the expected magnetometer norm. If the deviation exceeds this percentage, the version without magnetometer is used. For the dataset `data/walking_gait_tracking`, the best performing value was $magneticPercentage = 0.4$.
  - *magnetometer_norms*: Calculated using `util/norm_mean_mode.py`, this contains the mode of magnetometer norms sampled every 1,000 data points.

Each algorithm prints the error between the starting and ending positions—so to obtain values that make sense, they require datasets where the path returns to the starting point. They also output internal flags, parameters related to the Fusion algorithm, and a 3D GIF animation of the walking motion to visually interpret the results.

7.10 Files and Script

A key component of the project is the `files.csv` file, which maps the relationships between raw data files and the corresponding processed results. Each line in the CSV corresponds to a dataset and includes the following format:

```
Acc, Mag, Gyr, Quat, Data, ComplementaryFilter, EulerDirect,
FusionAdvancedCalibration, FusionAdvanced, FusionSimple, KalmanFilter
```

For example, the line for the *still* dataset is:

```
data/still/acc3_raw.csv,data/still/mag3_raw.csv,data/still/gyr3_raw.csv,data/still/
quaternions.csv,data/still/data.csv,results/still/complementary_filter_results.csv,
results/still/euler_direct_results.csv,results/still/fusion_advanced_calibration.csv,
results/still/fusion_advanced_nocalibration.csv,results/still/fusion_simple_results.
csv,results/still/kalman_filter_results.csv
```

The first line of `files.csv` always specifies the currently active dataset to be used by scripts.

Additionally, the file `run.sh` automates the entire workflow (assuming the IMU is already configured), executing the following steps:

1. Reads data from the IMU and converts it into a CSV file.
2. Runs all filtering algorithms to generate Tait-Bryan angle results.
3. Plots the results for comparison.

Bibliography

[1] X Aguado, F Funollet, and C Giralt. "Orientarse. De los sentidos a los instrumentos de orientación." In: *Apunts. Educación física y deportes* 4.18 (1989), pp. 05–07.

[2] Simegnew Yihunie Alaba. "GPS-IMU Sensor Fusion for Reliable Autonomous Vehicle Position Estimation". In: *arXiv preprint arXiv:2405.08119* (2024).

[3] Bashar Alsadik and Samer Karam. "The simultaneous localization and mapping (SLAM): An overview". In: *Surveying and geospatial engineering journal* 1.2 (2021), pp. 1–12.

[4] Maya Ameliasari, Aji Gautama Putrada, and Rizka Reza Pahlevi. "An evaluation of svm in hand gesture detection using imu-based smartwatches for smart lighting control". In: *Jurnal Infotel* 13.2 (2021), pp. 47–53.

[5] Steve Arar. *An Introduction to MEMS Vibratory Gyroscopes - Technical Articles — allaboutcircuits.com*. https://www.allaboutcircuits.com/technical-articles/introduction-to-mems-gyroscopes-vibratory-gyroscope/. [Accessed 17-04-2025]. 2022.

[6] Chaiyawan Auepanwiriyakul et al. "Accuracy and acceptability of wearable motion tracking for inpatient monitoring using smartwatches". In: *Sensors* 20.24 (2020), p. 7313.

[7] Bram JC Bastiaansen et al. "An inertial measurement unit based method to estimate hip and knee joint kinematics in team sport athletes on the field". In: *Journal of Visualized Experiments* 2020.159 (2020), pp. 1–8.

[8] Esmat Bekir. *Introduction to modern navigation systems*. World scientific, 2007.

[9] Binoy Bera and Madhumita Das Sarkar. "Piezoelectric effect, piezotronics and piezophototronics: a review". In: *Imperial Journal of Interdisciplinary Research (IJIR)* 2.11 (2016), pp. 1407–1410.

[10] Evandro Bernardes and Stéphane Viollet. "Quaternion to Euler angles conversion: A direct, general and computationally efficient method". In: *Plos one* 17.11 (2022), e0276302.

[11] Gaudenz Boesch. *Evolution of Motion Tracking: From Manual to Automated - viso.ai — viso.ai*. https://viso.ai/deep-learning/evolution-of-motion-tracking-from-manual-tracking-to-deep-learning/. [Accessed 31-03-2025].

[12] Johann Borenstein et al. "Mobile robot positioning: Sensors and techniques". In: *Journal of robotic systems* 14.4 (1997), pp. 231–249.

[13] Paul E Ceruzzi. *GPS*. MIT Press, 2018.

[14] EF Chehab, TP Andriacchi, and J Favre. "Speed, age, sex, and body mass index provide a rigorous basis for comparing the kinematic and kinetic profiles of the lower extremity during walking". In: *Journal of biomechanics* 58 (2017), pp. 11–20.

[15] Dongyao Chen, Kyong-Tak Cho, and Kang G Shin. "Mobile IMUs reveal driver's identity from vehicle turns". In: *arXiv preprint arXiv:1710.04578* (2017).

[16] Paul Clark. *GitHub - sparkfun/SparkFun_ICM-20948_ArduinoLibrary: Arduino support for ICM_20948 w/ portable C backbone — github.com*. https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary?tab=readme-ov-file. [Accessed 01-05-2025].

[17] NJS colorlib. *Personal — nitinjsanket.github.io*. https://nitinjsanket.github.io/tutorials/attitudeest/kf. [Accessed 20-02-2025].

[18] Eric R Craine. "Altitude measuring instruments: The emergence of the sextant". In: *Astronomy Quarterly* 2.7 (1978), pp. 115–136.

[19] Nathan DeVrio, Vimal Mollyn, and Chris Harrison. "Smartposer: Arm pose estimation with a smartphone and smartwatch using uwb and imu data". In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 2023, pp. 1–11.

[20] Aldebaran Documentation. *Joints; Aldebaran 2.4.3.28-r2 documentation — doc.aldebaran.com*. http://doc.aldebaran.com/2-4/family/robots/joints_robot.html. [Accessed 20-02-2025]. 2014.

[21] Mark Euston et al. "A complementary filter for attitude estimation of a fixed-wing UAV". In: *2008 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2008, pp. 340–345.

[22] *File:XYZ-DIS magnetic field coordinates.svg - Wikimedia Commons — commons.wikimedia.org*. https://commons.wikimedia.org/wiki/File:XYZ-DIS_magnetic_field_coordinates.svg. [Accessed 02-05-2025].

[23] Brice Guignard et al. "Validity, reliability and accuracy of inertial measurement units (IMUs) to measure angles: Application in swimming". In: *Sports Biomechanics* 23.10 (2024), pp. 1471–1503.

[24] https://www.facebook.com/sean.boerhout. *Complementary Filters for IMU Fusion — seanboe.com*. https://seanboe.com/blog/complementary-filters. [Accessed 02-05-2025].

[25] Lucia F Jacobs et al. "Olfactory orientation and navigation in humans". In: *PLoS One* 10.6 (2015), e0129387.

[26] John. *Accelerometer-Sensor, Working, Types, Specification, Selection, Applications — instrumentationtoday.com*. https://www.instrumentationtoday.com/accelerometer/2011/08/. [Accessed 17-04-2025]. 2011.

[27] Lauren Johnson. *Evolution of Navigation Systems - Inventionland — inventionland.com*. https://inventionland.com/blog/evolution-of-navigation-systems/. [Accessed 04-02-2025].

[28] Jremington. *AltIMU-AHRS/calibrate3.py at master · jremington/AltIMU-AHRS — github.com*. https://github.com/jremington/AltIMU-AHRS/blob/master/calibrate3.py. [Accessed 31-03-2025].

[29] Monish Katari et al. "Driving towards safety: the role of ecus and imus in advanced driver-assistance systems (ADAS)". In: *International Journal for Multidisciplinary Research* 6.2 (2024), pp. 1–14.

[30] Sung-Kyun Kim, Seokmin Hong, and Doik Kim. *A walking motion imitation framework of a humanoid robot by human walking recognition from imu motion data. In 2009 9th IEEE-RAS International Conference on Humanoid Robots*. 2009.

[31] Lefteris Kotsonis. *What Is Sensor Calibration And Why It Matters — botasys.com*. https://www.botasys.com/post/sensor-calibration. [Accessed 25-05-2025].

[32] Samuel A Lawoyin, Ding-Yu Fei, Ou Bai, et al. "A novel application of inertial measurement units (IMUs) as vehicular technologies for Drowsy driving detection via steering wheel movement". In: *Open Journal of Safety Science and Technology* 4.04 (2014), p. 166.

[33] Tian Lei, Abduallah A Mohamed, and Christian Claudel. "An IMU-based traffic and road condition monitoring system". In: *HardwareX* 4 (2018), e00045.

[34] Sebastian OH Madgwick. "AHRS algorithms and calibration solutions to facilitate new applications using low-cost MEMS". PhD thesis. University of Bristol, 2014.

[35] Robert Mahony, Tarek Hamel, and J-M Pflimlin. "Complementary filter design on the special orthogonal group SO (3)". In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE. 2005, pp. 1477–1484.

[36] Renato de Souza Melo et al. "Balance and gait evaluation: comparative study between deaf and hearing students". In: *Revista Paulista de Pediatria* 30 (2012), pp. 385–391.

[37] Rebecca Morelle. *Astrolabe: Shipwreck find 'earliest navigation tool' — bbc.com*. https://www.bbc.com/news/science-environment-41724022. [Accessed 04-02-2025].

[38] Hamid Didari Khamseh Motlagh et al. "Position Estimation for Drones based on Visual SLAM and IMU in GPS-denied Environment". In: *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*. IEEE. 2019, pp. 120–124.

[39] Lucía Alonso Mozo. *GitHub - luciaalonsomozo/IMUAlgorithms — github.com*. https://github.com/luciaalonsomozo/IMUAlgorithms. [Accessed 24-05-2025]. 2025.

[40] *Nordic Semiconductor - nRF-PPK2 — RS — es.rs-online.com*. https://es.rs-online.com/web/p/kits-de-desarrollo-de-alimentacion-motores-y-robots/2119949. [Accessed 06-05-2025].

[41] John D North. "The astrolabe". In: *Scientific American* 230.1 (1974), pp. 96–107.

[42] Boris Odehnal, Hellmuth Stachel, and Georg Glaeser. *The universe of quadrics*. Springer Nature, 2020.

[43] Vítor H Pinto et al. "Enhanced performance real-time industrial robot programming by demonstration using stereoscopic vision and an IMU sensor". In: *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE. 2020, pp. 108–113.

[44] Qniemiec. *By Qniemiec - Own work*. https://commons.wikimedia.org/w/index.php?curid=10893303. CC BY-SA 4.0. 2025. URL: https://commons.wikimedia.org/w/index.php?curid=10893303.

[45] Qniemiec. *East-North-Up (ENU) and North-East-Down (NED) Reference Frames*. https://commons.wikimedia.org/w/index.php?curid=10893168. Own work, CC BY-SA 3.0. 2008.

[46] Daniel Robert and Martin C Göpfert. "Novel schemes for hearing and orientation in insects". In: *Current opinion in neurobiology* 12.6 (2002), pp. 715–720.

[47] IW Scaysbrook, SJ Cooper, and ET Whitley. "A miniature, gun-hard MEMS IMU for guided projectiles, rockets and missiles". In: *PLANS 2004. Position Location and Navigation Symposium (IEEE Cat. No. 04CH37556)*. IEEE. 2004, pp. 26–34.

[48] Hatice Şeyma Selbesoğlu, Burak Barutçu, and Aytekin Çökelez. "The Brief History of Early Marine-Navigation". In: *Advanced Geomatics* 1.1 (2021), pp. 14–20.

[49] Martin Sipos et al. "Analyses of triaxial accelerometer calibration algorithms". In: *IEEE Sensors Journal* 12.5 (2011), pp. 1157–1165.

[50] TDK. *ICM-20948 Datasheet — TDK InvenSense — invensense.tdk.com*. https://invensense.tdk.com/download-pdf/icm-20948-datasheet/. [Accessed 10-02-2025].

[51] *Teslabs Engineering - A way to calibrate a magnetometer — teslabs.com*. https://teslabs.com/articles/magnetometer-calibration/. [Accessed 02-05-2025].

[52] Vectornav. *Learn more about magnetometer models and HSI calibration; VectorNav — vectornav.com*. https://www.vectornav.com/resources/inertial-navigation-primer/specifications--and--error-budgets/specs-hsicalibration. [Accessed 20-05-2025].

[53] YoungWonks Vidya Prabhu. *An introduction to magnetometers, where they are used and how they work — youngwonks.com*. https://www.youngwonks.com/blog/What-is-a-Magnetometer-and-How-Does-It-Work. [Accessed 17-04-2025].

[54] Yujue Wang et al. "FFCI: A Camera and IMU Sensors Based Multi-modal Neural Network for Activity Recognition in Smart Factory". In: *IEEE Transactions on Consumer Electronics* (2025).

[55] Darragh F Whelan et al. "Technology in rehabilitation: Comparing personalised and global classification methodologies in evaluating the squat exercise with wearable IMUs". In: *Methods of information in medicine* 56.05 (2017), pp. 361–369.

[56] Wikipedia. *Apollo PGNCS - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Apollo_PGNCS. [Accessed 04-04-2025].

[57] Wikipedia. *Arduino - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Arduino. [Accessed 04-04-2025].

[58] Wikipedia. *Axes conventions - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Axes_conventions. [Accessed 07-04-2025].

[59] Wikipedia. *Conversion between quaternions and Euler angles - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles. [Accessed 20-05-2025].

[60] Wikipedia. *Euler angles - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Euler_angles. [Accessed 20-05-2025].

[61] Wikipedia. *Gait analysis - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Gait_analysis. [Accessed 26-05-2025].

[62] Wikipedia. *Gimbal lock - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Gimbal_lock. [Accessed 17-04-2025].

[63] Wikipedia. *Gyroscope - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Gyroscope. [Accessed 20-05-2025].

[64] Wikipedia. *Hall effect - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Hall_effect. [Accessed 10-02-2025].

[65] Wikipedia. *Inertial measurement unit - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Inertial_measurement_unit. [Accessed 05-02-2025].

[66] Wikipedia. *Magnetometer - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Magnetometer. [Accessed 06-02-2025].

[67] Wikipedia. *MEMS - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/MEMS. [Accessed 20-05-2025].

[68] Wikipedia. *Quadric - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Quadric. [Accessed 20-05-2025].

[69] Wikipedia. *Quaternion - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Quaternion. [Accessed 20-05-2025].

[70] M Wylde et al. "Placement of inertial measurement units in Racket Sports: Perceptions of coaches for IMU use during training and competition". In: *International Journal of Racket Sports Science* 3.1 (2021), pp. 45–55.

[71] xioTechnologies. *GitHub - xioTechnologies/Fusion — github.com*. https://github.com/xioTechnologies/Fusion?tab=readme-ov-file. [Accessed 20-02-2025].

[72] xioTechnologies. *GitHub - xioTechnologies/Gait-Tracking — github.com*. https://github.com/xioTechnologies/Gait-Tracking. [Accessed 31-03-2025].

[73] Huiyu Zhou and Huosheng Hu. "Human motion tracking for rehabilitation—A survey". In: *Biomedical signal processing and control* 3.1 (2008), pp. 1–18.