CODEN:LUTEDX/(TEIE-5510)/1-84/(2024)

Industrial Electrical Engineering and Automation

Developing Stabilization System for Body-Worn Camera

Adam Bladh
Jakob Göransson

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Developing Stabilization System for Body-Worn Camera

Adam Bladh & Jakob Göransson

LUND
UNIVERSITY

Division of Industrial Electrical Engineering and Automation (LTH)

MSc Thesis TEIE-5510

Division of Industrial Electrical Engineering and Automation (LTH)
Lund University
Box 118
SE-221 00 LUND
Sweden

© 2024 by Adam Bladh & Jakob Göransson. All rights reserved.
Printed in Sweden.
Lund 2024

Abstract

This thesis focuses on the development of an image stabilization system for body-worn cameras. Body-worn cameras are frequently used in dynamic situations, exposing them to intense movements. Capturing sharp, high-quality footage in these conditions requires effective image stabilization systems. This investigation involved mapping the nature of the disturbances experienced by the body-worn camera through data collection and an external search for information. With a comprehensive literature study and the gathered data, a gimbal-based stabilization system was designed. The design process included selecting the necessary electrical components and developing custom software. CAD drawings of the design were created, followed by 3D printing of the parts. These parts were then assembled with the other components to construct a prototype. The prototype's performance was tested demonstrating an improvement in image stability. This study provides insight into image stabilization systems and their complexity. It also offers a practical solution to mitigate the unwanted effects of camera movement, enhancing the reliability of surveillance in dynamic settings.

Keywords: Image stabilization, body-worn camera, gimbal

Abbreviations

OIS - Optical Image Stabilization

EIS - Electronic Image Stabilization

MIS - Mechanical Image Stabilization

IMU - Inertial Measurement Unit

MCU - Micro controller Unit

FOC - Field-Oriented Control

DOF - Degrees of Freedom

BLDC - Brushless Direct Current

PID - Proportional–Integral–Derivative

Acknowledgements

First we would like to thank our academic supervisor Avo Reinap at the Division of Industrial Electrical Engineering and Automation (IEA), for all the help and support during this project.

We would also like to thank our company supervisor Albin Berggren for suggesting and guiding us through this project, and connecting us to the correct experts in the company. A big thank you to the SPE team at the company, especially Sebastian Göbel, Allan Hubble and Jonathan Hägerbrand for always having the time to answer our questions and discussing our ideas.

Preface

This work was done in collaboration with the Division of Industrial Electrical Engineering and Automation (IEA) at LTH, and an outside company. All parts of the thesis are a joint contribution between Adam Bladh and Jakob Göransson.

Great things are done by a series of small things brought together.

- Vincent Van Gogh

Contents

1 Introduction

Provides an introduction of this master thesis and states the background, objectives and limitations.

1.1 Background

Security concerns have become increasingly important in today's modern society. This sparks the necessity of developing new sophisticated surveillance technologies to protect properties, individuals and public places. Security cameras have taken a vital role in this development and are indispensable tools for threat detection and surveillance. In recent years, mobile body-worn cameras have emerged as essential technology for uniform occupations like police, military and security guards. This have introduced new challenges for surveillance camera technology. One of those challenges is the gathering of high quality footage under intense movements. The unwanted vibrations and shakes occurring from walking and running deteriorate the captured footage, making it harder to watch and to capture important situations. To solve this problem, different image stabilization technologies have been developed in recent years. Their goal is to reduce blurriness in photos and videos caused by camera movement, allowing for better image clarity and overall improved visual quality.

1.2 State of the Art

Image stabilization is a critical feature in modern cameras, essential for capturing sharp, high-quality footage. An effective image stabilization system is especially important for body-worn cameras used in uniform occupations, where dynamic situations are common. The demand for advanced image stabilization systems have therefore increased. Recent years have seen significant advancements in state-of-the-art image stabilization technology.

For body-worn cameras, the VM780 Body Camera Micro Gimbal Stabilizer by Hytera is a notable development. Its "mini-gimbal" structure, which tilts the image sensor and lens together, provides some anti-shake capabilities for wearable devices [Hytera, 2023].

Another product offering anti-shaking technology is the Ordro EP8 4K Head Mounted Camera, featuring a built-in two-axis gimbal anti-shake system [Ordro, n.d.].

Despite these advancements, current stabilization systems are often insufficient to handle larger vibrations, such as those caused by walking or running. This has led to ongoing research and development efforts to improve image stabilization systems for body-worn cameras.

1.3 Problem formulation and objectives

The purpose of this thesis is to provide an overview of the currently available stabilization methods for cameras, investigate which vibrations affect the footage from the body-worn camera, and evaluate each stabilization method's ability to reduce these vibrations. Furthermore, based on the results of this investigation, a solution shall be proposed and a prototype will be designed, constructed and tested to see if it reduces vibrations in the recorded footage.

The body-worn camera currently uses electronic image stabilization to help reduce the effects of high-frequency, low-amplitude vibrations. However, this stabilization system is insufficient when the camera experiences lower frequency, higher amplitude disturbances, such as those from walking and running. To address these types of vibrations, a new image stabilization system needs to be considered. The objective of this new image stabilization system is to counteract the larger disturbances resulting from external forces, ensuring better-captured footage.

To develop an image stabilization system for the body-worn camera, an understanding of the currently available stabilization methods is needed. The exact nature of the disturbances affecting the camera also needs to be investigated. A significant part of the work therefore involves a comprehensive literature study to gather information in the following areas:

- Types of stabilization methods for cameras.
- Existing solutions to stabilize body-worn cameras.
- The magnitude and types of disturbances affecting the body-worn camera.

Information about the disturbances affecting the body-worn camera is gathered through experimentation. This data maps the expected movement of the camera and sets the limitations of the stabilization system. These observations are then used to propose possible solutions that decrease the disturbances. Following this, an analysis is done to find a suitable solution, which is then developed into a prototype.

1.4 Limitations

The purpose of this thesis is to identify the disturbances caused by the wearer's upper body movement and propose solutions to counteract them. Therefore, unwanted disturbances caused by other factors are not considered.

In their thesis, *Image Stabilization for Body-Worn Cameras*, Samuel Bryngelsson and Jonathan Gustafsson looked at the stabilization of a body-worn camera by designing different mounts. Some of their findings serve as a basis for the work in this report. Based on their investigation, the following limitations have been set:

- The thesis compared fastening a body-worn camera with a magnet holster or a camera harness. By tracking an object in the recorded footage, the deviation of the object relative to the original image could be estimated. Using the worst-case estimation, an area of deviation was established to show how much the tracked object deviated compared to the original image size. The study found that the deviation area decreased from 65% to 21% of the original image size when using the harness instead of the holster [Bryngelsson and Gustafsson, 2023]. In other words, unwanted movements would occur if the camera was not well fastened. Therefore this thesis will use a similar camera harness, in order to minimize unwanted disturbances.
- In the thesis it was also found that the location of the body-worn camera impacted the vibrations affecting it [Bryngelsson and Gustafsson, 2023]. It is deemed outside the scope of this thesis to test and wear the body-worn camera in different locations on the body and only one position will be used.
- Finally, the thesis concluded that rotational movements of the body-worn camera had a larger impact on the destabilization of the recorded footage compared to translational movements [Bryngelsson and Gustafsson, 2023]. Therefore, the solution sought in this work will focus on eliminating rotational movements of the camera rather than addressing translational movements.

Furthermore, an electronic image stabilization system already exist in the body-worn camera, which is assumed to adequately handle small vibrations. The new image stabilization system shall therefore focus on larger vibrations, specifically the ones occurring during walking and running.

The goal of this work also includes the design, assembly, and evaluation of a functional prototype. In this context, the concept of the solution is prioritized over its size. Therefore, it is not a requirement that the finished prototype fits inside of the body-worn camera.

1.5 Course of work

The thesis began with a comprehensive literature study on the concept of image stabilization. Discussions with employees at the company were also conducted to understand practical implementations and current measures in the body-worn cameras. Following this, data on the movements affecting the camera was collected and analyzed, establishing the foundation for the stabilization system's requirements. Various concepts and systems were evaluated against these requirements, leading to the selection of a gimbal system as the final concept.

Next, the necessary electrical components for the system were chosen, CAD models of the prototype were created, and the parts were 3D printed. Each component was individually tested to evaluate its performance, and corresponding software was developed. The components were then assembled with the 3D printed parts, and the main software code was produced. Finally, tests were conducted on the prototype to asses its performance.

1.6 Contribution

This thesis aims to further the knowledge of stabilization systems for body-worn cameras and the challenges they face. One part is to chart the vibrations affecting the camera due to the wearer walking or running. The other part is to propose a design for a stabilization system capable of counteracting the charted vibrations. This design is also realized as a prototype and its effectiveness is evaluated.

1.7 Outline

Chapter 1 Provides an introduction to the subject and problem, as well as limitations and the course of work of the study.

Chapter 2 Gives theoretical information about key components and topics discussed in the project.

Chapter 3 Outlines the method used to gather data about the body-worn cameras movements. Also explains how the data is processed and turned into a set of requirements.

Chapter 4 Presents the design process for the prototype, starting with concept generation, explaining component choices, showcasing several design iterations and ending with the assembled prototype.

Chapter 5 Goes through the theory and kinematics guiding the controller design, and how the software implementation is done.

Chapter 6 Presents the final evaluation of the prototype, detailing the testing process and the results obtained.

Chapter 7 Concludes the thesis with a final reflection of the entire project, and a recommendation for areas that should be further explored.

2
Technical background

This chapter explains and expands on theories and equations used throughout the thesis in subsequent chapters. The information presented is based upon an extensive literature search and serve as the foundation for the work done in this thesis.

2.1 Image stabilization

There is no definitive way to categorize different image stabilization systems. For this thesis, the systems have been divided into three main categories: *Optical Image Stabilization* (OIS), *Electronic Image Stabilization* (EIS) and *Mechanical Image Stabilization* (MIS). They are categorised as follows: OIS corrects disturbances by physically moving components like lenses and image sensors inside the camera. EIS corrects disturbances using software that digitally corrects frame by frame to counteract the unwanted motion. MIS corrects disturbances by physically moving the whole camera. Furthermore, the information presented in this section is relevant for small scale cameras, such as those used in body-worn cameras or in mobile phones.

2.1.1 Optical image stabilization

OIS is a technique designed to reduce the effects of involuntary camera shakes. It utilizes sensors to gather data about undesired movements and then uses this information to make real-time physical adjustments to optical elements inside the imaging system. The technology aims to keep the trajectory for the optical path between the target and the image sensor aligned and constant when exposed to unintended vibrations [Rosa et al., n.d.]. Figure 2.1 shows a simplified demonstration of the basic principle of an OIS system. The difference between point X and Y in the left image leads to lower image quality. This degradation is then fixed in the right image by moving a lens to compensate for displacement and correct the optical patch.

There are several ways to implement an OIS system into a camera. The two most common methods are lens shift and camera tilt. Lens shift moves the internal lenses using translational motion while keeping the image sensor fixed (similar to the system demonstrated in figure 2.1). Camera tilt on the other hand moves both the internal lenses and the image sensor in unity with angular motion [Rosa et al., n.d.]. Advanced systems also exist where the internal lenses and image sensor move separately but in tandem to produce even better stabilization and sharper images [Nicholson and Summersby, 2024].

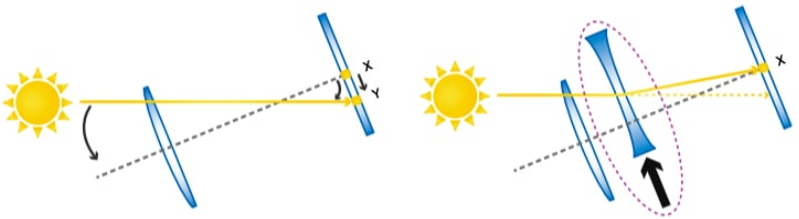


Figure 2.1 Demonstration of an OIS system [ROHM Semiconductor, 2013]

All methods usually require miniaturized actuators, drivers, high-precision sensors and microcontrollers. The sensors gather data on the disturbances affecting the system and send this data to the microcontroller. The microcontroller executes a control algorithm to determine the necessary corrections and transmits this information to the drivers. The drivers convert the information into commands for the actuators. With the correct commands, the actuators physically move the camera module the precise amount and in the required direction [Rosa et al., n.d.].

The majority of OIS systems, in today's market, deliver motion correction angles of approximately one degree. This is sufficient to correct and counter jitters caused by human hands in stationary settings. For dynamic conditions like walking and running, the amplitude of the generated shakes generally exceeds one degree. These conditions require greater compensation angles to achieve efficient image stabilization [Hansen, 2023].

2.1.2 Electronic image stabilization

EIS, also known as digital image stabilization, aims to reduce distortion introduced by involuntary camera shakes solely with the use of software. This form of stabilization uses digital processing techniques on captured images and videos. One of the techniques involves cropping portions of images that deviate from a predefined area, centering the image, and then enlarging it to meet the required dimensions [Golik, 2006].

Figure 2.2 demonstrates this process. The top row shows the unstable image sequence. The software then takes the center of focus and aligns it in each frame, which introduces cropping and/or areas without data, as seen in the middle row. This is then compensated for in the last row by introducing borders or enlarging the frames.

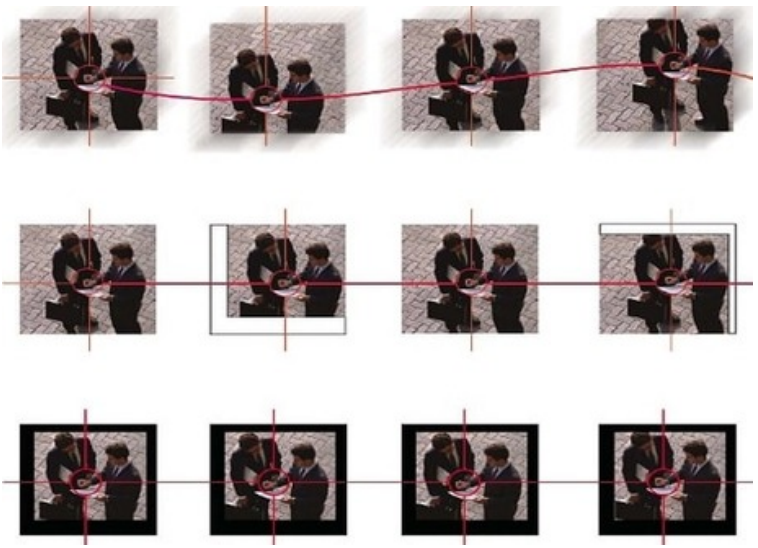


Figure 2.2 Demonstration of an EIS system [Golik, 2006]

The process requires the frames to move, necessitating the introduction of margins. This can be accomplished either by enlarging the image sensor or through digital magnification. Both methods negatively impact the system. A larger image sensor increases the size and cost of the build, while digital magnification reduces quality and increases the loss of visual information, due to cropping and enlargement of frames. The extent of the stabilization is therefore significantly limited by these factors [Golik, 2006].

2.1.3 Mechanical image stabilization

MIS works by using sensors to detect shifts in the camera platform and then counteract any unwanted motion by physically moving the platform [Souza and Pedrini, 2018]. One commonly used MIS system is the gimbal system. It detects camera shakes using sensors, and sends the sensor data to a microprocessor to compute the amount and direction of correction. This information is then transmitted to a control system that actuates motors to move the camera in order to counteract the detected motion. These systems efficiently minimize vibrations but require additional components, such as actuators, sensors and drivers, which increase weight and power consumption [Doe, 2024].

2.2 Gyroscope

Gyroscopes are devices that measure the angular rate of motion. They are mounted on objects where they can detect angular velocity when the object rotates. Various classes of gyroscopes exist, such as mechanical, optical, and electromechanical. Each utilizes different technologies to acquire the same result. The one used in this thesis is the *Micro-Electro-Mechanical System* (MEMS) gyroscope. MEMS gyroscopes, unlike traditional gyroscopes, do not include rotating parts that require bearings. They instead utilize vibrating mechanical components as their sensing elements to detect angular velocity. The technology functions by leveraging the Coriolis effect, where the vibrating component experiences deflection when exposed to rotation [Passaro et al., 2017]. The MEMS gyroscopes are ideal for applications with size and weight constraints and are commonly used in electronic devices such as wearable gadgets and smartphones. Their compact size and affordability make them well-suited for mass production and widespread use in different applications and industries [Ericco Inertial System, 2023].

2.3 Accelerometer

Accelerometers are sensor devices that measure the acceleration of motion. Mounted on objects, they provide accurate measurements of the physical acceleration experienced by that object. The most common type is the piezoelectric accelerometer. This technology works by using a sensing crystal with a seismic weight attached. When acceleration occurs, the crystal experiences a force exerted by the weight. The piezoelectric crystal then converts the force into electrical signals, which are measured and converted into acceleration readings [Omega Engineering, n.d.].

Accelerometers can be found in a lot of different technologies. Nearly all smartphones are equipped with an accelerometer, and they are also found in airbags, drones, and on space stations [Omega Engineering, n.d.].

2.4 Magnetometer

Magnetometers are sensor devices that measures magnetic fields. They can detect the strength and orientation of magnetism, such as the relative change of a magnetic field at a point or the magnetization of a material. There are different types of magnetometers. Scalar magnetometers, such as the Proton Precession Magnetometer, work by utilizing an external magnetic field to align hydrogen atoms within a fluid. When the field is removed, the protons revert back to their original orientation while emitting a signal. This signal can then be measured and converted into magnetic field strength. Vector magnetometers, like Fluxgate Magnetometers, work by wrapping a compact magnetic core with two coils of wire. An alternating current is transmitted through one coil, causing the core to magnetize and demagnetize rapidly. The second coil then registers changes in the magnetic field. Magnetometers are valuable sensors that are useful in a lot of different applications. They are commonly used to measure the earth's magnetic field, locate resources, and navigate our surroundings [Electricity & Magnetism, 2024].

2.5 Inertial measurement unit

An *Inertial measurement unit* (IMU) is a electronic device that combines different sensors to collect data. They are among the most common devices used in navigation systems, motion captures, and robotics. IMUs typically integrate multiple sensors such as 3-axis gyroscopes, 3-axis accelerometers, and occasionally 3-axis magnetometers. These units combine all the previous sensors to track and measure acceleration, force, angular rate, and magnetic fields. With sensor fusion, all measurements can be merged to offer heading and orientation in 3D space [Gunasekaran, 2019].

2.6 Sensor fusion

Sensor fusion is an algorithm used to processes data gathered from sensors. It considers the different sensors' advantages and disadvantages to enhance the overall system's accuracy. Since sensors alone only generate raw data, an algorithm needs to optimize and convert this raw data into useful information [Strout, 2023].

Each individual sensor also has major drawbacks that sensor fusion mitigates. For example, accelerometers have problems with noise interference, gyroscopes are prone to drift, and magnetometers suffer from electromagnetic interference. There are three commonly used algorithms to fuse the IMU sensor data: Kalman filters, complementary filters, and the Madgwick algorithm [Strout, 2023].

2.7 Brushless direct current motor

Brushless direct current (BLDC) motors are powerful and compact electric motors frequently found in both industrial and household electronics. They offer improved efficiency, lower operational noise, a higher torque-to-weight ratio, and rapid response compared to other types of motors. BLDC motors use electronic commutation instead of brushes, enhancing their performance and longevity. Like most electrical motors, the basic physical construction is separated into two parts, the stator and the rotor. This is shown in figure 2.3. The stator is the stationary part with windings that act as electromagnets when current is applied. The rotor is the rotating part with permanent magnets. The electromagnets generate magnetic fields that interact with the permanent magnets in the rotor, making the rotor rotate. This rotation occurs due to the attraction and repulsion forces between the magnetic poles. With a dedicated controller, the magnetic fields from the stator windings alternate in a desired manner by switching the current flow [Jenish, 2023]. The controller utilizes an algorithm to create the appropriate currents, which produce magnetic fields that ensure the desired motion of the motor. One of the most efficient control algorithms for electrically commutated motors is the *Field-Oriented Control* (FOC) algorithm [Skuric et al., 2022].

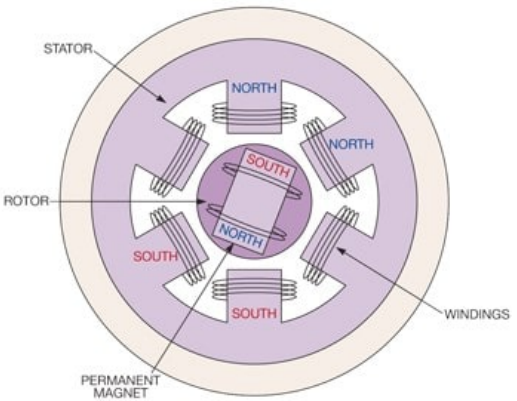


Figure 2.3 Simplified model of a BLDC motor[EDN, 2010]

2.8 Field-oriented control

FOC is a technique used for controlling various electrically driven motors. It is an advanced mathematical method for controlling the behavior of electrical motors, enhancing their efficiency and position accuracy [Goodwin, 2023]. The basic idea involves controlling motor torque by managing the three-phase stator currents. To maximize generated torque, the stator field vector most remain perpendicular to the rotor field vector. This is achieved by measuring the rotor position and then manipulating the three-phase stator currents (i_a, i_b, i_c) to align the stator field vector orthogonal to the measured rotor position. The stator field vector is separated into its direct axis (d-axis) and quadrature axis (q-axis). The d-axis aligns with the rotor field vector, while the q-axis is 90 degrees to the d-axis. By forcing the d-axis to zero and maximizing the q-axis, the stator field vector and rotor field vector remain perpendicular to each other [Ulusoy, 2020]. The Clark and Park transformations are used to convert the three-phase stator currents from the stationary reference frame (a, b ,c) to the rotating reference frame (d, q), and vice versa. This mathematically converts the three-phase time-variant quantities into two-phase time-invariant quantities, allowing easier control with PID controllers [Electrical4U, 2024]. Figure 2.4 shows an overview of a standard control loop utilizing FOC.

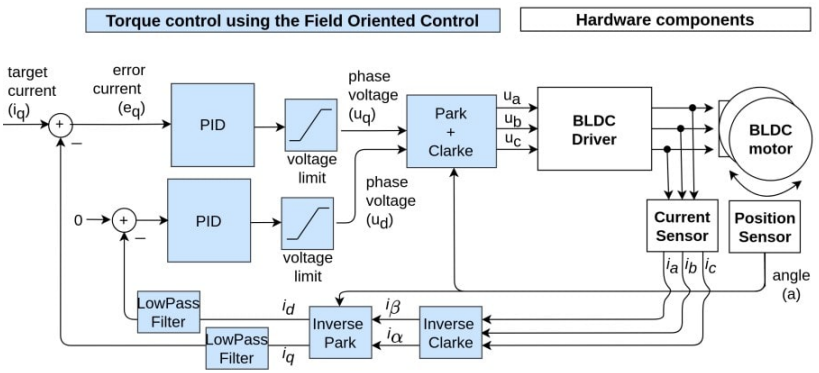


Figure 2.4 Overview of a torque control loop utilizing FOC [Skuric et al., 2022]

2.9 Coordinate systems

To fully describe the motions of a body, one uses both a global and a local coordinate system. The global coordinate system is detached from the body, meaning it does not translate or rotate with it. The local coordinate system is however attached to the body, usually located at the center of mass. In contrast to the global coordinate system, the local coordinate system does move and rotate with the body [Flores, 2015].

Several coordinate systems are referenced throughout this work, and are further explained in their respective sections: 3.1 and 5.1.1.

2.10 Nautical sequence

One way to describe rotations in 3D space, is the nautical sequence. It consists of three rotations: yaw θ, pitch ϕ, and roll ψ. To describe this sequence, a reference coordinate system is first defined with positive x-, y-, and z-axes, pointing in the forward, left, and up direction. Using this frame as a reference, the first rotation in the nautical sequence is around the z-axis, followed by a rotation around the y-axis, and finally a rotation around the x-axis. In the nautical system, the reference coordinate system rotates alongside these rotations [Haslwanter, 2018]. Refer to figure 2.5 for a visual representation of the three rotations.

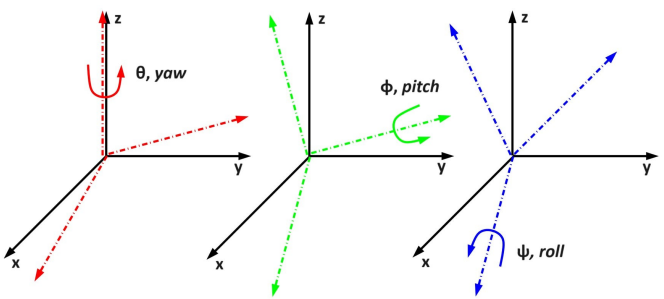


Figure 2.5 Image of the nautical sequence [Ardakani and Bridges, 2010]

Note that the nautical sequence not necessarily refers to a positive rotation around the z-axis, followed by the y-axis and ending with the x-axis. The order of rotations around the axis are dependent on how the coordinate system is defined. With the nautical sequence it is the order of yaw, pitch and roll, which is important to follow. How the rotations θ, ϕ, and ψ are defined for the camera in this thesis, can be seen in figure 2.6.

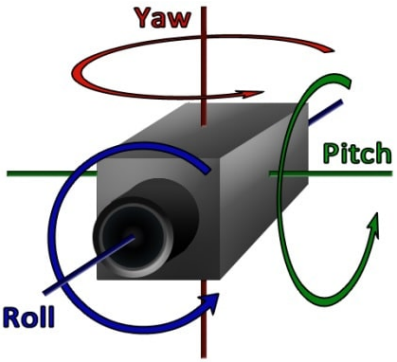


Figure 2.6 Example of yaw, pitch and roll

The three rotations are expressed mathematically using the three rotation matrices $R_z(\theta)$, $R_y(\phi)$, and $R_x(\psi)$, shown in equations 2.1, 2.2, and 2.3. They can all be combined into one rotation matrix, $R_{nautical}$, as seen in equation 2.4 [Haslwanter, 2018].

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.1}$$

$$R_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \tag{2.2}$$

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \tag{2.3}$$

$$R_{\text{nautical}} = R_z(\theta)R_y(\phi)R_x(\psi) =$$
$$= \begin{pmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi & \cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi \\ \sin\theta\cos\phi & \sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi \\ -\sin\phi & \cos\phi\sin\psi & \cos\phi\cos\psi \end{pmatrix} \tag{2.4}$$

The sequence of rotations presented in this section as the nautical sequence may be known by another name. The three angles yaw, pitch, and roll can be used in several other combinations to describe the same rotation in 3D space [Haslwanter, 2018]. In this thesis, the nautical sequence, along with the names and symbols for the three angels described in this section, are used throughout the text.

2.11 Gimbal lock

A problem known as gimbal lock can occur with the nautical sequence and similar three-angle sequences. For certain values of θ, ϕ, and ψ, singularities occur. This can be shown mathematically by examining $R_{nautical}$ from equation 2.4, when $\phi = \pm\frac{\pi}{2}$ (or $\cos\pm\frac{\pi}{2} = 0$ and $\sin\pm\frac{\pi}{2} = \pm 1$) [Challis, 2021]. Equation 2.5 shows the resulting matrix.

$$
\begin{aligned}
&R_{\text{nautical}} = \\
&= \begin{pmatrix} 0 & \pm\cos\theta\sin\psi - \sin\theta\cos\psi & \pm\cos\theta\cos\psi + \sin\theta\sin\psi \\ 0 & \pm\sin\theta\sin\psi + \cos\theta\cos\psi & \pm\sin\theta\cos\psi - \cos\theta\sin\psi \\ \mp 1 & 0 & 0 \end{pmatrix} = \\
&= \begin{pmatrix} 0 & -\sin(\theta \mp \psi) & \pm\cos(\theta \mp \psi) \\ 0 & \cos(\theta \mp \psi) & \pm\sin(\theta \mp \psi) \\ \mp 1 & 0 & 0 \end{pmatrix}
\end{aligned} \tag{2.5}
$$

The rotation matrix now depends only on the difference or sum of the angles θ and ψ. Changing either or both results in a rotation around the same axis, meaning the matrix has lost one degree of freedom. This phenomenon, called gimbal lock, can occur with any sequence of rotations similar to the nautical sequence [Challis, 2021]. The problem of gimbal lock is also shown in figure 2.7, which illustrates a three-axis gimbal system. The middle axis (blue to light green) rotates the inner axis (light green to green), preventing it from re-orienting in one of the three rotation directions. Consequently, the rotation axis of the outermost (light blue to blue) and innermost axis become the same [Haslwanter, 2018].

Figure 2.7 Illustration of gimbal lock, the inner axis (dark green) cannot be rotated in the direction of the dotted arrows [Haslwanter, 2018]

There are ways to avoid gimbal lock. One way is to ensure that the sequence of rotations made never reaches the singularity caused by gimbal lock [Challis, 2021]. Even approaching the gimbal lock orientation can cause problems, as rapid movements are then required to change direction [Haslwanter, 2018]. This means that the gimbal's allowable movement is restricted, and not all orientations are attainable. Gimbal lock can also be avoided by using an alternative method to represent the body's orientation, one that does not have the same singularity issue [Challis, 2021].

2.12 Quaternions

One of the ways to describe the orientation of an object in 3D space, instead of using the angles yaw, pitch, and roll, is quaternions. Quaternions are a way of representing the nine elements of a rotation matrix using four parameters. They also avoid the singularities of gimbal lock [Challis, 2021]. This thesis does not include a complete description of what quaternions are, or the mathematical proofs behind them. It only includes the parts that are relevant to describe a body's orientation.

The full quaternion $\tilde{q}$ has four components: q_0, q_1, q_2, and q_3. The first component q_0 represent the "scalar component", while q_{1-3} represent the "vector component". Equation 2.6 shows a full quaternion [Haslwanter, 2018].

$$\tilde{\boldsymbol{q}} = q_0 + (q_1 * \tilde{\boldsymbol{i}} + q_2 * \tilde{\boldsymbol{j}} + q_3 * \tilde{\boldsymbol{k}}) = q_0 + \boldsymbol{q} * \boldsymbol{I} \tag{2.6}$$

To describe a pure rotation in 3D space, one can use a unit quaternion. This is a quaternion with a norm of $|\tilde{\boldsymbol{q}}| = 1$. It describes a rotation around a unit vector $\mathbf{n}$ by the angle θ, as shown in equation 2.7. This unit quaternion has the mathematical properties shown in equations 2.8a to 2.8d [Haslwanter, 2018].

$$\tilde{\boldsymbol{q}} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}[n_i * \tilde{\boldsymbol{i}} + n_j * \tilde{\boldsymbol{j}} + n_k * \tilde{\boldsymbol{k}}] = q_0 + \boldsymbol{q} * \boldsymbol{I} \tag{2.7}$$

$$\tilde{\boldsymbol{q}}^{-1} = q_0 - \boldsymbol{q} * \boldsymbol{I} \tag{2.8a}$$

$$|\tilde{\boldsymbol{q}}| = \sqrt{\cos^2\frac{\theta}{2} + \sin^2\frac{\theta}{2}} \tag{2.8b}$$

$$|\boldsymbol{q}| = \sqrt{q_1^2 + q_2^2 + q_3^2} = \sin\frac{\theta}{2} \tag{2.8c}$$

$$\boldsymbol{q} \,||\, \boldsymbol{n} \tag{2.8d}$$

Using a unit quaternion, which describes a rotation, one can rotate a vector $\bar{\boldsymbol{v}}$ with equation 2.9. As seen in the equation, the vector being rotated is represented as a quaternion, but with a zero scalar component [Haslwanter, 2018]. The combination of rotations can be achieved by multiplying quaternions, as equation 2.10 illustrates. The order of multiplication reflects the sequence in which the rotations occur. The rightmost quaternion in the equation represents the first rotation, while the leftmost quaternion represents the final rotation. Consequently, quaternion multiplication is not commutative. Changing the order of multiplication yields a different combined rotation quaternion [Challis, 2021].

$$\tilde{\boldsymbol{v}}' = \begin{pmatrix} 0 \\ \bar{\boldsymbol{v}}' \end{pmatrix} = \tilde{\boldsymbol{q}} \circ \tilde{\boldsymbol{v}} \circ \tilde{\boldsymbol{q}}^{-1} \tag{2.9}$$

$$\tilde{\boldsymbol{s}} = \tilde{\boldsymbol{r}} \circ \tilde{\boldsymbol{q}} \tag{2.10}$$

In both equations 2.9 and 2.10, quaternion multiplication is performed. The process for calculating the result of quaternion multiplication is demonstrated in equation 2.11. It can also be expressed in matrix form, as shown in equation 2.12 [Challis, 2021].

$$\begin{aligned} \tilde{\boldsymbol{q}} \circ \tilde{\boldsymbol{r}} = &(q_0 r_0 - q_1 r_1 - q_2 r_2 - q_3 r_3) \\ &+ \tilde{\boldsymbol{i}}(q_0 r_1 + q_1 r_0 + q_2 r_3 - q_3 r_2) \\ &+ \tilde{\boldsymbol{j}}(q_0 r_3 + q_2 r_0 + q_3 r_1 - q_1 r_3) \\ &+ \tilde{\boldsymbol{k}}(q_0 r_3 + q_3 r_0 + q_1 r_2 - q_2 r_1) \end{aligned} \tag{2.11}$$

$$\tilde{\boldsymbol{q}} \circ \tilde{\boldsymbol{r}} = \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} \tag{2.12}$$

From a unit quaternion, describing a rotation, one can determine the angular velocity using equation 2.13. The result of the equation, $\tilde{\boldsymbol{\omega}}$, is a pure quaternion $(0, \boldsymbol{\omega})$, where $\boldsymbol{\omega}$ is a vector that describes the angular velocity with respect to 3D space. It provides the angular velocity around the x-, y-, and z-axis for the rotation in $\frac{rad}{s}$. Since the angular velocity is no longer represented as a quaternion, the angular acceleration is simply the derivative of the angular velocity, as shown in equation 2.14 [Haslwanter, 2018].

$$\tilde{\omega} = 2 * \frac{d\tilde{\boldsymbol{q}}}{dt} \circ \tilde{\boldsymbol{q}}^{-1} \tag{2.13}$$

$$\boldsymbol{\alpha} = \frac{d\boldsymbol{\omega}}{dt} \tag{2.14}$$

Quaternions can be interpolated similarly to vectors, but not in exactly the same manner. For quaternions, one must consider their four-dimensional nature, meaning each quaternion can be represented as a point on a hyper-sphere. Consequently, any interpolation technique for quaternions must accommodate the curvature of the hyper-sphere. One such method is *Spherical linear interpolation*, also referred to as Slerp. This method guarantees a consistent angular velocity between the two quaternions [Challis, 2021].

The equation for the Slerp formula is depicted in equation 2.15. Its components include $\tilde{\boldsymbol{q}}_{\mathbf{1,2}}$, which are the quaternions being interpolated between; θ, the angle between the quaternions; and f, the fraction of the interval between the quaternions. An example of the interpolation between two quaternions is illustrated in figure 2.8 [Challis, 2021].

$$\tilde{\boldsymbol{q}} = \frac{\sin\left((1-f)\theta\right)}{\sin\theta}\tilde{\boldsymbol{q}}_1 + \frac{\sin\left(f\theta\right)}{\sin\theta}\tilde{\boldsymbol{q}}_2 \tag{2.15}$$

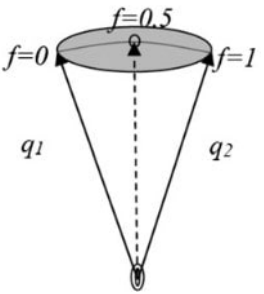


Figure 2.8 Interpolation between two quaternions using Slerp [Challis, 2021]

3
Image stabilization requirements

To evaluate the suitability of different stabilization systems, a set of criteria is created. The criteria are based on information obtained during the literature study and data gathered through experimentation. The aim is to establish and quantify the movements affecting the body-worn camera.

3.1 Data gathering

To collect data about the movements of the body-worn camera, a data gathering method is designed and set up. The goal of the method is to mimic real-life use of the camera while ensuring ease of repetition. The basic idea of the method is to attach a camera to a person who walks, jogs and runs on a treadmill, and the movements of the camera are recorded.

Figure 3.1 Body-worn camera with sensor box attached

To attach the body-worn camera to a person, a camera holster (see figure 3.1) is used. This holster secures the camera in the middle of the wearer's chest. The holster is chosen to try and minimize unwanted movements caused by an insufficiently secured camera.

Figure 3.1 also show a blue sensor box attached to the camera. This is the STEVAL-MKSBOX1V1 from STMicroelectronics, a wearable sensor platform. It contains the necessary sensors together with an accompanying app to record motion [STMicroelectronics, 2024b]. One of the pre-made application can record the orientation of the box as quaternions.

The orientation of the box describes how the IMU's coordinate system is rotated in reference to a global coordinate system. As described in section 2.9, the IMU's coordinate system is a body-fixed coordinate system. A sketch of the IMU's axes are shown in figure 3.2. Due to how the box is attached to the camera, the z-axis points in the same direction as the camera lens, and the x-axis points downwards. Referring back to the nautical sequence from section 2.10 and figure 2.6, the yaw θ, pitch ϕ, and roll ψ are defined as rotations around the x-, y- and z-axis, respectively.

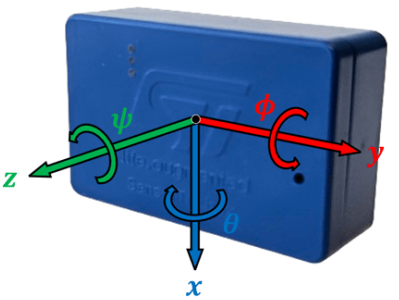


Figure 3.2 The IMU's coordinate axes

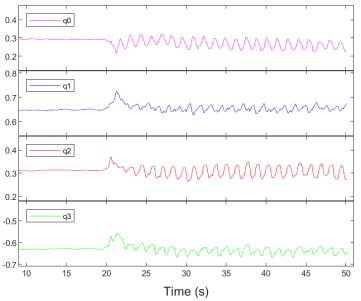
With the described setup, the wearer utilizes a treadmill, repeating the same procedure at several different speeds. The three speeds used on the treadmill are 3.0 mph, 8.0 mph, and 14.0 mph (mph is used since it is the original unit used by the treadmill). These correspond to a walking, jogging, and running pace. For each pace the procedure starts with the wearer standing still for 20 seconds to obtain an accurate reading of the camera's "home position". This is followed by 30 seconds of walking, jogging or running, after which the gathering method is completed.

3.2 Data processing

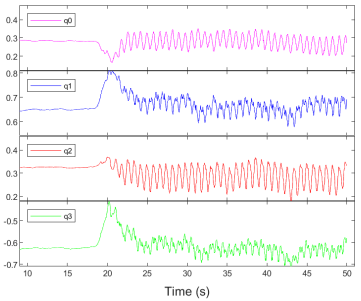
The gathered data is separated into the three different movement speeds. For each of them, the data consists of a number of timestamps and an associated quaternion. Each quaternion describes the rotation of the IMU's local coordinate system in relation to the earth's global coordinate system, at that current timestamp.

3.2.1 Quaternion data

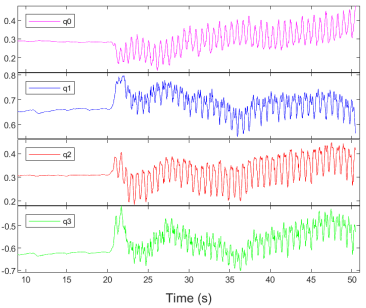
As described in section 2.12, a quaternion consists of four parts. The collected data can therefore be plotted as figures 3.3a, 3.3b, and 3.3c shows. All of them illustrate how the four parts of the quaternion change over time during the duration of the experiment. Any data from before 10 s is not included in the graph, since unintended movements or sensor misreadings could occur at the start of the experiment.



(a) Speed: 3.0 mph



(b) Speed: 8.0 mph



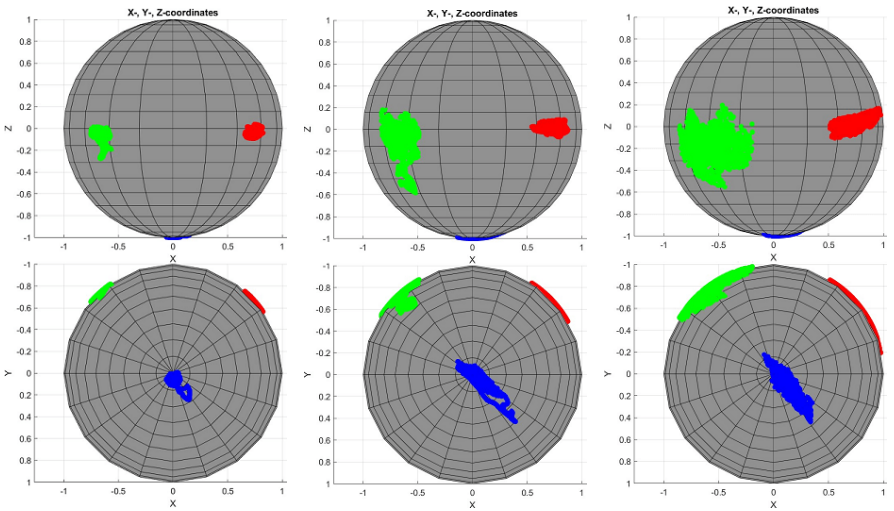
(c) Speed: 14.0 mph

Figure 3.3 Recorded quaternions for different speeds

In these graphs, a larger difference is seen between the quaternions recorded for 3.0 *mph* and 8.0 *mph*, compared to 8.0 *mph* and 14.0 *mph*. All three graphs show the quaternions oscillating over time. This oscillation has a fairly stable baseline for speed 3.0. As the speed increases to 8.0 and 14.0, the baseline around which the quaternions oscillate becomes increasingly unstable and start changing over time. The baseline of the quaternions indicate the average direction the camera is facing. A stable baseline suggests that the camera is pointing in the same average direction over the course of the experiment. In contrast, a changing baseline implies that the average direction changes over time. This variation could be due to the wearer leaning their upper body in different directions when jogging or running, whereas they maintain a steadier posture when walking.

3.2.2 Range of motion

Quaternions are not intuitive, so to better understand the data, the range of motion of the STEVAL-MKSBOX1V1 is first calculated. As described in section 2.12, unit quaternions are able to rotate any vector using equation 2.9. By representing each of the coordinate axes of the STEVAL-MKSBOX1V1 as a vector, they can be rotated using the recorded quaternion data. This process produces a set of coordinate points for each axis, for all recorded quaternions. These points represent the orientation of each coordinate axis over the duration of the data gathering process. Plotting these sets of coordinate points on a sphere results in the figures 3.4a, 3.4b, and 3.4c.



(a) Speed: 3.0 *mph* **(b)** Speed: 8.0 *mph* **(c)** Speed: 14.0 *mph*

Figure 3.4 Range of motion for the coordinate axes at different speeds

Each figure shows two spheres with three colored areas. Blue corresponds to the coordinate points of the x-axis, red represent the y-axis, and green is the z-axis. The size and shape of each colored area indicate the extent of motion for each axis during the experiment. The areas are directly related to the yaw, pitch and roll motions of the camera.

Recalling figures 3.1 and 3.2 from section 3.1, the z-axis points in the camera lens' direction, the y-axis points to the side of the camera, and the x-axis points downward. The size of the green area thus shows the range of motion for the yaw and pitch, the red area for the yaw and roll, and the blue area for the pitch and roll. A larger yaw movement causes the green and red areas to expand toward each other, a larger pitch movement expands the green and blue areas toward each other, and a larger roll movement expands the read and blue areas toward each other. It is clear, especially for 14.0 mph, that the green area is the largest area. This indicates that yaw and pitch rotations are the main rotations affecting the camera. The elongated shapes of both the red and blue areas further support this, since the smaller roll movement results in less growth of the areas.

The spheres in figures 3.4 also show the increase in maximum and minimum values for yaw, pitch and roll. The maximum and minimum values are represented by the overall extent of each colored area. The figures show no significant change in the maximum and minimum pitch between 8.0 mph and 14.0 mph, as indicated by the green areas in figures 3.4b and 3.4c, both stretching between a z-value of around $-0.6 \rightarrow +0.2$. Comparing this to figure 3.4a, where the green area only stretches between around $-0.3 \rightarrow +0.05$, which is clearly a smaller area. In contrast the maximum and minimum value of yaw increases for all three speeds. This result could further support the change in behaviour noticed between walking, jogging and running in the previous section (3.2.1). Since the maximum and minimum pitch does not seem to change between 8 mph and 14 mph, it indicates that the wearer leans the same amount of forward and backwards, when they jog and run.

3.2.3 Angular rotation, velocity, and acceleration calculations

To further clarify and quantify the movements of the body-worn camera, the following parameters are calculated:

1. The rotation (yaw, pitch, and roll) over time around each of the IMU's axes (figure 3.2).
2. The angular velocity around each axis.
3. The angular acceleration around each axis.

As section 2.12 show (specifically equations 2.13 and 2.14), the angular velocity can be directly calculated from the quaternions, and the angular acceleration is the derivative of the velocity. To simplify this process, the quaternions are initially interpolated to be evenly distributed along the time span. This is done by recalculating all timestamps to have uniform intervals. The quaternion data is then interpolated using Slerp (see section 2.12 and equation 2.15) to fit the new timestamps. After this, the angular velocity and acceleration are calculated using equations 2.13 and 2.14. Both the angular velocity and acceleration are assumed to reach their calculated values halfway between the two data points. If the angular velocity $\boldsymbol{\omega}$ is calculated between the quaternions $\tilde{\boldsymbol{q}}_{\mathbf{1}}$ and $\tilde{\boldsymbol{q}}_{\mathbf{2}}$, it is assumed to be reached at the time $\frac{t_2+t_1}{2}$, where $t_{1,2}$ are the time points for quaternions $\tilde{\boldsymbol{q}}_{\mathbf{1,2}}$. This same reasoning is repeated for the angular acceleration, using the time points for the velocity.

As noted earlier, there is no definitive way to transform a quaternion into rotational angles around the coordinate axes. One can use Euler angles or the nautical sequence, but since they depend on the order of rotations, they do not give a singular answer. Through testing it is found that quaternions describing pure rotations around the x-, y-, and z-axis are not always transformed into rotation angles around only the x-, y-, or z-axis. A quaternion sequence describing a rotation around the x-axis followed by a rotation around the z-axis could be transformed into a rotation around the z-axis followed by the y-axis. Instead, the solution used in this thesis is to look at the movement of each coordinate axis in the planes adjacent to it. The sequence of calculations used is:

1. Calculate the average quaternion recorded during the standstill (approximately between the time points 10 and 20 seconds) and save as the ”home position”.

2. Rotate the x-, y-, and z-axis by each quaternion in the data set, to get the IMU’s coordinate system’s new axes, in relation to the global coordinate system.

3. Rotate the frame of reference to fit the ”home position’s” frame of reference (a vector (1, 0, 0) in the global coordinate system, rotated by the home position quaternion will become (1, 0, 0) in the new frame of reference).

4. Mirror each vector into the two adjacent planes yielding two new vectors (for example, the x-axis after rotation is mirrored in the xy- and zx-plane).

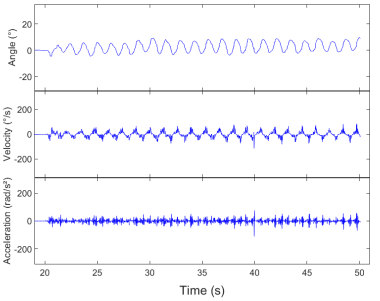
5. The angle of rotation around the plane's normal is calculated between the mirrored vector and the corresponding coordinate axis (if the original vector was the rotated x-axis, the angle is measured against the x-axis).

To summarize and give an example, one can look at what happens when the z-axis is rotated by a quaternion $\tilde{\boldsymbol{q}}$. Assuming that $\tilde{\boldsymbol{q}}$ is not equal to the "home position" quaternion, the z-axis vector gets the new direction (z_1, z_2, z_3) after the rotation. This direction is in relation to the global axes, and after rotating the frame of reference, the direction becomes $z' = (z'_1, z'_2, z'_3)$ in the "home position" frame. Now the $\boldsymbol{z'}$ vector is mirrored into its adjacent planes (yz and zx), yielding the two vectors $(0, z'_2, z'_3)$ and $(z'_1, 0, z'_3)$. For the first mirrored vector, the angle of rotation around x-axis is acquired, and from the second one, it is the angle around the y-axis. This process is repeated for the two other axes yielding six total equations describing the three rotations.

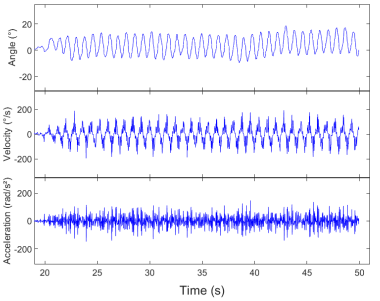
In theory, the six equations can be separated into groups of two, where each pair gives the same value for the rotation around the x-, y-, or z-axis. However when these six equations are used on the recorded quaternion data, they result in six unique values. This probably stems from rounding errors or faults in the recorded data. For both the x- and y-axis, the error between the two calculated values is less than one degree on average. For the z-axis, the error is closer to 3 degrees at some points. To negate this difference the average of the two calculated angles is used as a good enough approximation of the actual value.

The results of the described calculations for angular rotation, velocity and acceleration are shown in the following figures 3.5 to 3.7. Each graph is color coded, with blue representing motions around the x-axis (yaw), red representing motions around the y-axis (pitch), and green representing motions around the z-axis (roll). For each color there are three graphs, one for each speed used in the data gathering process. The graphs are presented to give a visual understanding of the calculated data in this chapter.

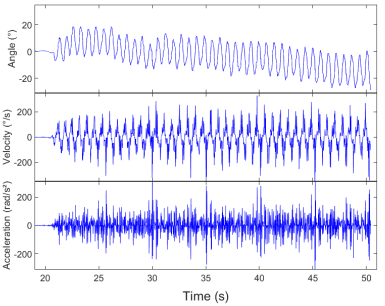
Figures 3.5a, 3.5b, and 3.5c show the relative stable oscillations for the yaw motions of the body-worn camera. The same behaviour of the baseline changing over time for 8.0 *mph* and 14.0 *mph* (as seen in figures 3.3a, 3.3b, and 3.3c) is visible in these graphs.



(a) Speed: 3.0 *mph*



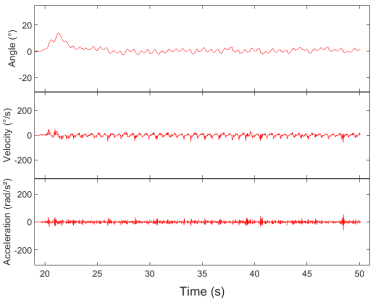
(b) Speed: 8.0 *mph*



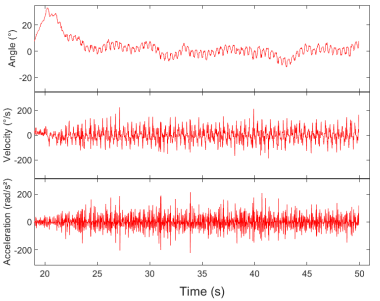
(c) Speed: 14.0 *mph*

Figure 3.5 Result of calculations for yaw θ at different speeds

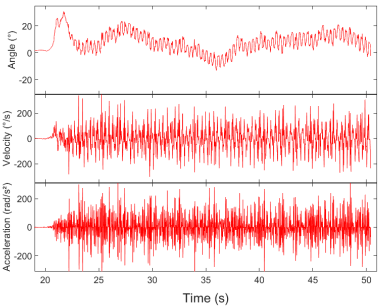
Following the result for yaw, figures 3.6a, 3.6b, and 3.6c display the change of pitch for the three different speeds. Compared to the results for yaw, the oscillations for pitch are more erratic and have a less stable baseline. A possible explanation for this difference is how the body-worn camera is affected when the wearer takes a step. With the impact of each step the body-worn camera will experience an added disturbance or small bounce, which primarily acts on the pitch rotation.



(a) Speed: 3.0 *mph*



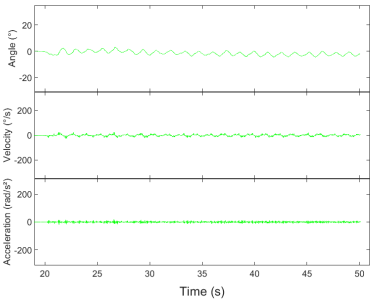
(b) Speed: 8.0 *mph*



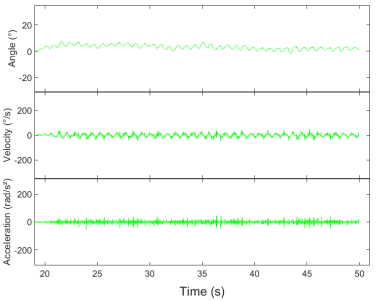
(c) Speed: 14.0 *mph*

Figure 3.6 Result of calculations for pitch ψ at different speeds

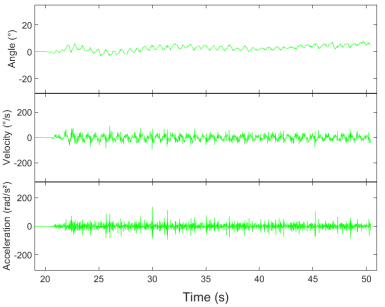
The final figures 3.7a, 3.7b, and 3.7c are the results of the roll calculations. It is quite clear how small the roll rotations are compared to yaw and pitch, even at 14.0 *mph*. This indicates further the necessity to focus on the yaw and pitch movements of the body-worn camera, since they most likely have the largest impact on the record footage.



(a) Speed: 3.0 *mph*



(b) Speed: 8.0 *mph*



(c) Speed: 14.0 *mph*

Figure 3.7 Result of calculations for roll ϕ at different speeds

In summation, the calculation of the data was done separately for walking, jogging and running. The recorded quaternions can be directly calculated into angular velocity, which can then be differentiated into angular acceleration. The rotation of the IMU's coordinate system is used to calculate the angular rotation around the x-, y-, and z-axis.

3.3 Data evaluation

Based upon the result of the angular rotation, velocity, and acceleration calculations, the maximum and minimum of each of them is estimated by looking at figures 3.5a to 3.7c. These estimations are shown in table 3.1.

	x-axis (Yaw θ)	y-axis (Pitch ϕ)	z-axis (Roll ψ)
Rotation γ (°)	$-29 \rightarrow +19$	$-13 \rightarrow +33$	$-4 \rightarrow +8$
Velocity $\dot{\gamma}$ ($°/s$)	$-440 \rightarrow +330$	$-300 \rightarrow +350$	$-94 \rightarrow +94$
Acceleration $\ddot{\gamma}$ (rad/s^2)	$-350 \rightarrow +380$	$-429 \rightarrow +345$	$-95 \rightarrow +134$

Table 3.1 Estimated maximum and minimum values

The values in the table indicate how much each axis is impacted by the movement of the body-worn camera. Connecting it back to the nautical sequence, θ spans almost 50°, followed by ϕ with a span of around 45° and finally ψ at 12°. The table shows a significant gap between how much roll movement there is compared to yaw and pitch. This gap between ψ, and θ or ϕ stays consistent for both the angular velocity and acceleration and it is also visible in the graphs in sections 3.2.2 and 3.2.3.

The results in table 3.1 also reinforces the findings of Samuel Bryngelsson and Jonathan Gustafsson's thesis, which suggested that the rotational movement of the camera, especially yaw and pitch rotations, were the leading cause of camera shake [Bryngelsson and Gustafsson, 2023].

θ	$\pm 30°$
$\dot{\theta}$	$\pm 440°/s$
$\ddot{\theta}$	$\pm 380 rad/s^2$
ϕ	$\pm 33°$
$\dot{\phi}$	$\pm 350°/s$
$\ddot{\phi}$	$\pm 429 rad/s^2$
ψ	$\pm 8°$
$\dot{\psi}$	$\pm 94°/s$
$\ddot{\psi}$	$\pm 134 rad/s^2$

Table 3.2 Calculated requirements

Using the worst case estimations of the data in table 3.1, the requirements each solution needs to fulfill is set. They are displayed in table 3.2, and represent the calculated requirements that a design has to fulfill in order to be considered viable. In addition to these calculated requirements, the design should also allow movement with three degrees of freedom. This allows the effect of the yaw, pitch, and roll movements on stabilization to be evaluated.

4
Prototype design

Based on the requirements and criteria established in the previous chapter, the next step is to conceptualize and design a prototype. This prototype must be a 3 DOF capable solution to ensure stabilization along each rotational axis. Each image stabilization system and prototype concept is evaluated on its own merits. No consideration is taken to how two systems or concepts could be combined in order to fulfill the requirements.

4.1 Choice of image stabilization system

With the set of requirements established in section 3.3, the different stabilization methods explained in section 2.1 are evaluated against them. Table 4.1 outlines some of the advantages and disadvantages of each image stabilization system. These are based not only upon the information gathered in section 2.1, but also from discussions with employees at the company.

	Advantages	Disadvantages
OIS	• Maintains image quality • Works well in low light • Performs consistently • Low impact on battery life	• Higher manufacturing cost • Bulkier camera modules • Hardware-dependent • Mechanical components • Low angle compensation
EIS	• Cost-effective • Compact design • No moving parts • Software based	• Lowered image quality • Poor low-light performance • Computational load • Low angle compensation
MIS	• High-quality stabilization • Consistent performance • Maintains image quality • Works well in low light • High angle compensation	• Requires calibration • Higher manufacturing cost • Bulkier system • Hardware-dependent • Mechanical components

Table 4.1 Comparison of Image Stabilization Techniques

The key disadvantage of an OIS system for this project is its limited angle compensation. As noted in section 2.1.1, current OIS systems on the market offer motion correction angles of approximately one degree for small scale cameras, such as those found in smartphones. Given that the proposed system requires compensation of up to 30 degrees, implementing an OIS system is inadequate.

The main drawbacks of an EIS system is also its limited angle compensation. As described in section 2.1.2, the introduction of margins for each frame negatively impact the final outcome. Enhancing the angle compensation necessitates increasing these margins, which further compounds the issue.

Another significant drawback is the computational load required for real-time stabilization, as demonstrated in Emil Manelius' thesis *Improving Image Stabilization in Modern Surveillance Cameras*. Manelius concludes that real-time stabilization in surveillance cameras is not feasible due to the excessive computational demands on current hardware, making it impractical within the necessary time frame [Manelius, 2023].

Based upon these reasons, the most viable option at present is to develop a MIS system. Although MIS systems can be bulky and require additional components, they effectively address the maximum and minimum values in table 3.2.

4.2 Concept Generation & Selection

Initially, various conceptual MIS systems are sketched, each capable of rotating an object around a single axis. Parallel to this, an external search is conducted, involving an investigation of leading technologies and companies in the field to gather inspiration. Figure 4.1 showcases some of the more reasonable sketches. All of them are viable 1 DOF solutions.

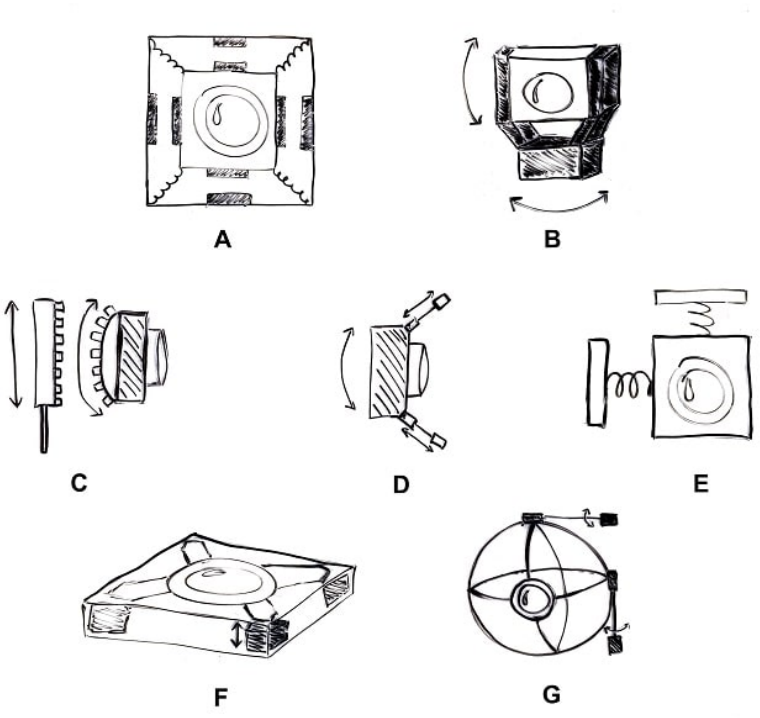


Figure 4.1 MIS concept sketches (camera lens is denoted by "water drop" shape)

The nine concepts showcased in the figure are evaluated and compared to each other, and the set requirements. Below is a summary of this evaluation:

A: This concept involves using variable magnetic fields around the camera module to rotate it. However, this approach poses a challenge since the magnetic fields could interfere with other sensors and components in the camera. To address this issue, magnetic shielding would need to be added, which would increase the size and complexity of the solution.

B: The idea for this concept is to use rotational motors to keep the camera module stable and level despite external forces and movement. With three motors mounted on the X, Y, and Z axes, the system enables 360-degree rotation along all axes, but it also increases the size and power consumption of the system.

C: With this concept a translating linear motion is converted into a circular motion using gears. The advantage is that less force has to be applied by the actuator to keep the camera module stable, which in turn means smaller actuators. However, the sketch is only of a 1 DOF design, and how to implement this concept as a 3 DOF design poses some difficulties. The linear actuators would have to rotate with the camera module or perhaps be connected to it with some form of ball joint. This could result in a overall larger system or be more complex than just using rotational actuators.

D: The reasoning behind this concept is to use linear actuators at each corner of the camera's front to angle it correctly. A drawback with this is the inability to compensate for rotational movements, making it viable only as a 2 DOF design.

E: This concept relies on inertia, with the camera either placed in a fluid or attached to springs. The idea is that the material absorbs and mitigates the force from shakes and movement, slowing the reaction and making the camera more stable. However, it quickly becomes apparent that this approach is not viable due to the wide range of the unwanted movements and the difficulty of finding the perfect inertia index for such a broad range.

F: With this concept linear actuators, such as piezoelectric actuators, is added in each corner. Each actuator moves up and down in tandem to reach desired angles to stabilise the camera. This design keeps module size within reasonable limits but only works as a 2 DOF system.

G: The idea behind this concept is to use rotational motors to rotate a sphere on which the camera is attached. This can be implement into a 3 DOF design. However, the sphere will most likely be subject to significant wear and tear damage, since high friction is required between the motors and the sphere to prevent slippage.

After evaluating the concepts, the two most promising are B and F. Since concept F only works as a 2 DOF system, concept B is selected as the most viable option.

4.3 General system architecture

Based upon the sketch of the chosen concept, the components required for building the prototype need to be identified. To do this a block diagram is made, which shows the major components needed to control a motor in response to external movements. Using both the chosen concept and the block diagram in figure 4.2, the following list of necessary components is established:

- Three motors to rotate the camera along each rotational axis.
- Three encoders to get position feedback from the motors.
- Three motor drivers to control each motor.
- A MCU to control the entire system.
- An IMU to measure the orientation of the system.
- A power supply to power the entire system.
- A camera to record the stabilized footage.

It should be noted that this is not a complete list of components required to actually build the prototype. These are simply the most important ones, which have a large impact on the design of the prototype.

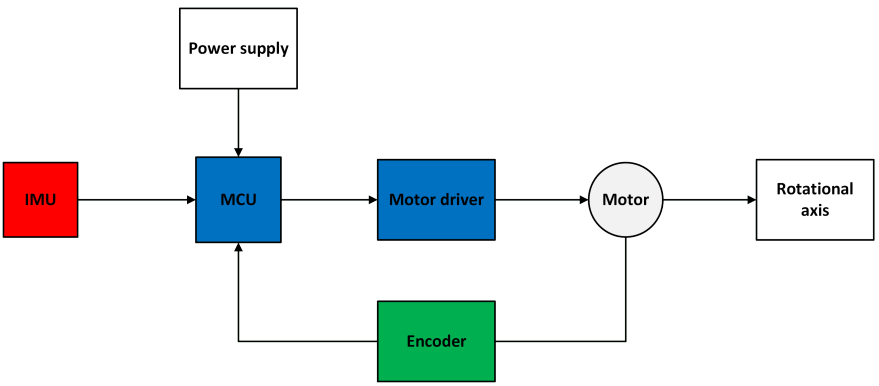


Figure 4.2 Block diagram of the system

4.4 Component selection

The main components used in the final prototype are listed in table 4.2. Other components have been tested throughout the work on this thesis, but are not necessary to discuss in this report. The choice of components have been based upon capacity, availability, compatibility with the MCU and ease of programming.

Component:	Description:
STEVAL-GMBL02V1	Gimbal evaluation card
STM32F303RE	MCU
STSPIN233	Motor driver
ICM-20948	Inertial measurement unit
ROB-20441	BLDC motor
AS5048A	Magnetic encoder
COM-15208	Buck-Boost Converter

Table 4.2 List of main components

The STEVAL-GMBL02V1 serves as the main board, featuring the STM32F303RE MCU and three STSPIN233 low voltage three-phase motor drivers with integrated sense capabilities [STMicroelectronics, 2024a]. The STEVAL-GMBL02V1 is refereed to as the MCU board throughout this text. It is chosen due to it containing an "integrated environment for three axis gimbal controller applications" as STMicroelectronics describes it [STMicroelectronics, 2024a]. The necessary connections, resistors, capacitors, etc. to program and control three gimbal motors are available on the board.

The ICM-20948 is a 9-axis IMU containing a gyroscope, accelerometer, and magnetometer. It includes a digital motion processor that offloads computation of motion processing algorithms from the host processor. This minimizes power consumption and ensures optimal performance of the data generated through sensor fusion [TDK InvenSense, 2017].

Three ROB-20411 BLDC motors are needed to rotate the camera around all three axes. These are three-phase brushless gimbal stabilizer motors, known for their high efficiency and torque. They operate smoothly and are compatible with magnetic encoders, ensuring precise position detection [sparkfun, 2024b]. The ROB-20411 also fulfills the angular velocity and acceleration requirements set in table 3.2.

Three AS5048A magnetic encoders determine the motor's position. These are 14-bit rotary position sensors based on contactless magnetic sensor technology [ams OSRAM, 2024].

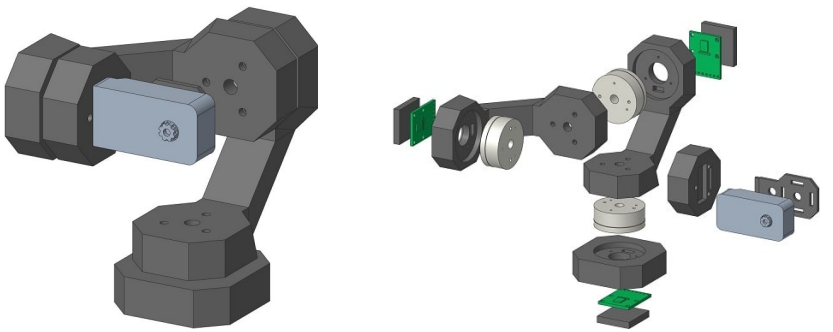
The COM-15208 is a buck-boost converter used to increase the voltage received from the portable power supply [sparkfun, 2024a].

4.5 CAD models

Over the course of this project, the design of the prototype is changed in three major iterations for development and improvement. The three iterations mark significant redesigns, but between each iteration smaller redesigns occur to make parts fit better together, or add missed features. For all CAD models, the motor closest to the camera module is referred to as the innermost motor, while the motor closest to the base of the prototype is referred to as the outermost motor.

4.5.1 First iteration

Based on the selected components and concept B in sketch 4.1, a CAD model is made in order to 3D print the prototype. Figure 4.3 displays the resulting CAD model, showing both the assembled model (4.3a), and an exploded view (4.3b). The dark grey objects in the figures represent the 3D printed parts of the prototype. The figure also highlights the magnetic encoders in green, the BLDC motors in white, and the camera module in light grey. All dark grey objects are designed specifically for this project, while any other colored object is based upon a manufactured component. Additionally, the camera in this CAD model is an external camera, not the actual module used in the body-worn camera.



(a) Assembled view of CAD model **(b)** Exploded view of CAD model

Figure 4.3 CAD model of the first iteration

The main parts of the gimbal CAD model is a base platform, two identical right angle arms, and a camera holder. The base platform and each arm houses a magnetic encoder, and all three are connected together by the motors. In this model the motors are numbered one to three starting with the outermost motor and ending with the innermost. The MCU card and the IMU is not visible in the CAD model, as they were kept separate from the gimbal prototype at this stage.

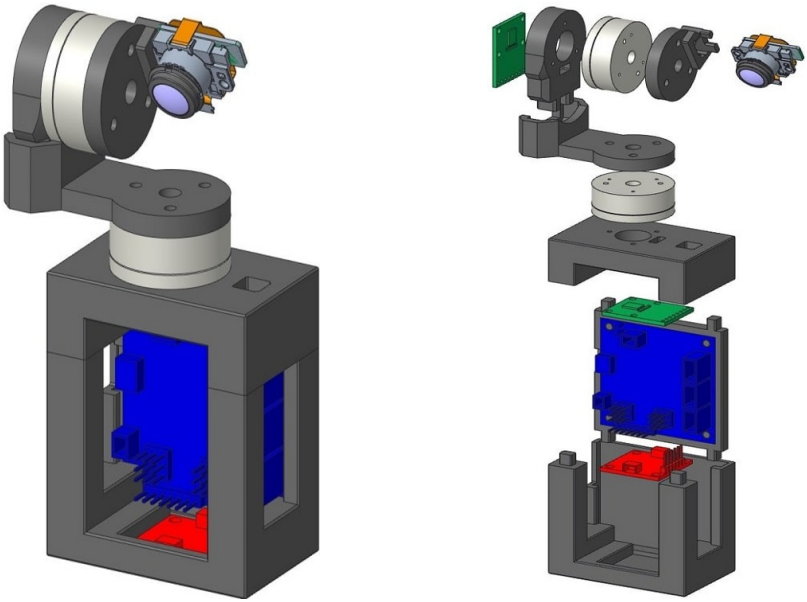
After 3D printing and assembling the prototype, a big issue becomes apparent. If the prototype is placed as shown in figure 4.3a, motor one and three operate normally, but motor two struggles to rotate properly. This happens because motor two has to work against the gravitational pull and the torque added by the second gimbal arm. The added torque is primarily due to the weight of motor three combined with the leverage arm's length between motor two and three.

Another potential issue discovered after assembly is the position of the camera. The axes of rotation for the three motors do not intersect at a common point, resulting in the camera lens being offset from this intersection. A consequence of this is that the camera lens experiences both rotation and translation, when the motors rotate. This is an issue when the kinematics of the system are calculated, which is further explained in section 5.1.

4.5.2 Second iteration

After taking into consideration the problems discovered in the first iteration, a second CAD model is made. It is shown in figure 4.4, and the same color code from the first iteration is used. The new blue object is the MCU board and the red object is the IMU. The camera module is also changed, to the one present in the actual body-worn camera. Note that this second iteration is only a 2 DOF system instead of a 3 DOF system like the first iteration. This change is made in order to first implement the less complex 2 DOF system and later add the third axis. With this change, the outermost motor is now motor one, and the innermost motor is motor two.

With the smaller gimbal arm, less torque is required to rotate it, which in turn makes it easier to control and actuate. This was tested by turning the prototype on its side and checking if the motor two could overcome the added torque due to gravity, which it could. It is now possible to effectively control the motors no matter the orientation of the prototype, in contrast to iteration one. The camera module is also placed so that the lens is positioned in the intersection of the motors' rotational axes. Another change from the first iteration is that all of the components of the prototype are modeled and fit inside of the prototype's body.



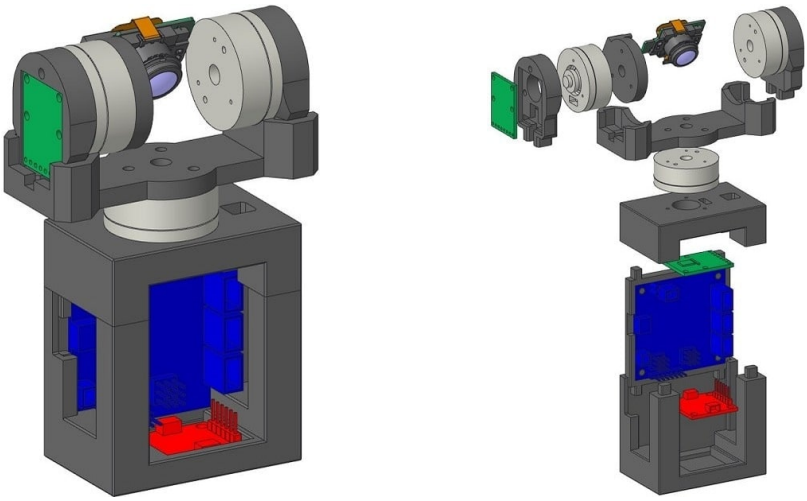
(a) Assembled view of CAD model **(b)** Exploded view of CAD model

Figure 4.4 CAD model of the second iteration

During testing of this model, it is discovered that it does not successfully stabilize the camera module. Instead it possibly makes it more unstable. The problem is once again connected to the weight of the motor and the resulting torque. Specifically, the problem stems from where the center of mass is placed. With the current design the center of mass is skewed towards motor two, meaning that motor one's rotational axis does not intersect with it. The consequence of this is that motor one is subject to more torque, and it is not powerful enough to hold against this added torque. As a result the gimbal arm becomes unstable during movement, leading to an overall worse stabilization than the original unstabilized body-worn camera.

4.5.3 Third iteration

The third and final iteration tries to solve the center of mass issue by mirroring the gimbal arm, as figure 4.5 shows. The same color code of the previous iteration is used for this model. The added motor is just there to provide a counter weight to the gimbal arm and is not connected to the MCU or controlled in any way. Therefore the motors are numbered the same way as in iteration two.



(a) Assembled view of CAD model **(b)** Exploded view of CAD model

Figure 4.5 CAD model of the third iteration

By adding the third motor, the gimbal arm is more balanced around motor one's rotational axis. The results of this are immediately apparent, when the prototype is tested. Instead of worsening the stabilization, as iteration two did, a clear improvement in stability is visible during testing. How the testing is done is presented in section 6 and the results from it are shown in section 6.2. As this is the final iteration, there is no proposed design for how the 3 DOF solution should look. It is however clear that a re-design is needed to make sure that each gimbal arm is balanced and that the center of mass intersects with each motor's rotational axis.

4.6 Final Prototype design

The assembled final prototype design is shown in figure 4.6. All components are connected and powered by a portable power supply. The main components, such as the MCU and IMU, are housed inside of the body. The camera module from the actual body-worn camera is fastened in the center of the motors. The red component at the front is the buck-boost converter, which increases the voltage received from the power supply.

Figure 4.6 Final prototype design

Reflecting on the concept choice made in section 4.2, a major reason for selecting the initial concept was its capability to support a 3 DOF solution. Since the proposed final design is a 2 DOF system, this initial decision should be reevaluated. The chosen concept, optimized for 3 DOF, may not be the most suitable for a 2 DOF system, and a different concept might be better suited.

5
System Implementation

With the prototype assembled, a controller needs to be designed and implemented, which responds correctly to outside disturbances. The sought behaviour of the controller is to cancel out the movement of the body-worn camera by rotating the camera module. To do this, the relation between the rotation of the motors and the orientation of the camera module is needed. It is also necessary to find how the camera module should be re-oriented in order to correctly negate the movements of the camera body.

5.1 Kinematics

In order to design a proper controller, the kinematics of the system are needed. The first step to find the kinematics is to define the coordinate systems of the prototype. A conversion between the quaternions recorded by the prototypes IMU and the corresponding motor angles are also needed to define the controller's objectives.

5.1.1 Coordinate systems

The three main parts of the prototype is the IMU sensor, the camera module, and the body. Each of them have their own coordinate system, but since the stabilization system only concerns rotation, the IMU's coordinate system can be merged with the body's. The reason for this is that the IMU is placed inside the body and moves with it, meaning any rotation impacts both identically. Both the IMU's and the camera module's coordinate systems are body-fixed (see section 2.9), and a third global coordinate system is required as a reference. The most suitable global coordinate system is the earth's, which the IMU uses as a reference to estimate its orientation. In summary, the three coordinate systems needed to describe the kinematics of the system are:

- The earth's coordinate system, referred to as the earth frame.
- The IMU's coordinate system, referred to as the IMU frame.
- The camera module's coordinate system, referred to as the CAM frame.

The orientation of the IMU frame is dependent on the placement of the IMU sensor in the prototype. This orientation is described in relation to the earth frame by a quaternion $\tilde{\boldsymbol{Q}}$. How the IMU frame is oriented in relation to the prototype is shown in figure 5.1. It also shows the orientation of the CAM frame, which has been chosen to simplify the kinematics later on.

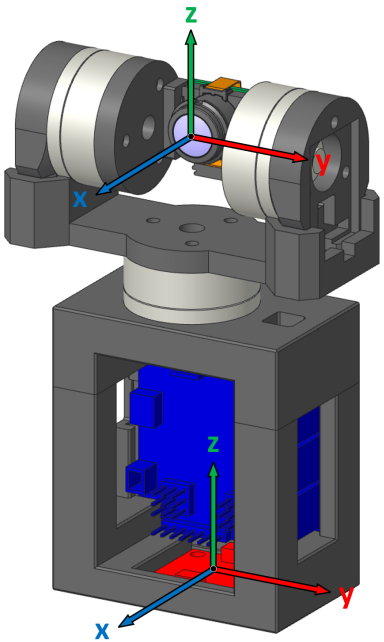


Figure 5.1 Coordinate systems for the prototype

As stated earlier, the goal of the stabilization system is to counteract the rotations acting on the prototype body by rotating the camera module. However, not all rotations should be counteracted. To understand why, the concept of the wearer's forward direction is needed. It is the direction of the entire person and is the same direction as the wearer's chest is facing when at standstill. Referring back to figure 5.1, the forward direction is equal to the IMU frame's x-axis when the wearer is not moving. It is important to note that the forward direction is not fixed in regards to the earth frame. If it were, the camera would continue to point south even if the wearer turned east. Another way to define the forward direction is to see it as the wearer's overall heading.

With the forward direction established, the goal of the stabilization system is to always point the camera module in this direction. As long as the forward direction is constant, all rotations acting on the prototype are negated. Then, when the forward direction changes, the stabilization system allows the camera module to be rotated the same way the forward direction is rotated.

To simplify the kinematics, the CAM frame is defined to be aligned with the IMU frame when no rotations are acting on the prototype. They are therefore aligned when both the camera module and the prototype body are pointing in the forward direction. As figure 5.1 shows, the x-axis of the CAM frame goes through the center of the camera lens, and the y-axis is aligned with the closest motors rotational axis. Connecting this back to the nautical sequence in section 2.10, yaw θ is the rotation around the z-axis, pitch ϕ is around the y-axis, and roll ψ is around the x-axis for the CAM frame. The same definition of θ, ϕ, and ψ is used for the IMU frame.

5.1.2 Conversion between quaternions and motor angles

For this section the gimbal prototype will be assumed to be a 3 DOF system and therefore three motors are available to control the orientation of the camera.

To figure out the relation between the motor angles and the orientation of the CAM frame, the standard position of the prototype is first defined. This position is when the CAM frame is aligned with the IMU frame, meaning that all axes are pointing in the same direction. Refer to figure 5.1 for a visual representation of the standard position. When the gimbal is in this position, the rotations of the motors are consider to be zero. Another way to explain it is that the standard position is the same position the camera would have if there was no gimbal, and the camera module was fixed to the prototype body.

In figure 5.2 the CAM and IMU frames are in the standard position, and the three motors M_1, M_2, and M_3 of the prototype are also visible. M_1 is the outermost motor, and M_3 is the innermost motor. Using the definitions for yaw, pitch and roll in section 5.1.1, the motor angle β_1 of motor M_1 corresponds to θ, β_2 corresponds to ϕ, and β_3 is ψ. Since $\beta_{1,2,3}$ are directly tied to θ, ϕ, and ψ, they can describe the orientation of the CAM frame using the nautical sequence (section 2.10). Instead of the sequence yaw, pitch and roll, the motor angle β_1 followed by β_2 and then β_3 are used to describe the orientation.

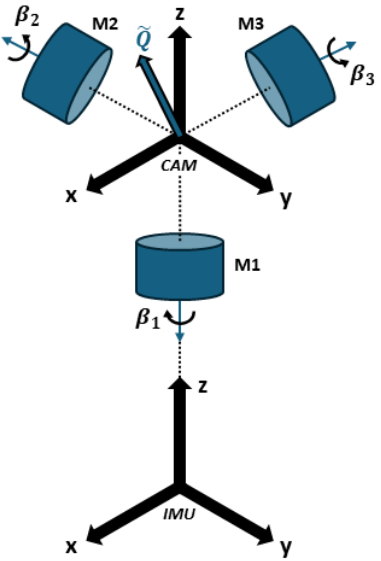


Figure 5.2 IMU and CAM frame plus motor angles

Figure 5.2 also illustrates a quaternion $\tilde{\boldsymbol{Q}}$, which represent a new orientation of the CAM frame. As equation 2.10 in section 2.12 shows, any rotation quaternion can be split into multiple consecutive quaternions to describe the same rotation. The quaternion $\tilde{\boldsymbol{Q}}$ can therefore be split into three quaternions, see equation 5.1.

$$\tilde{\boldsymbol{Q}} = \tilde{\boldsymbol{T}} \circ \tilde{\boldsymbol{S}} \circ \tilde{\boldsymbol{R}} \tag{5.1}$$

These three quaternions are then mapped to the nautical sequence and the motor angles. The first rotation $\tilde{\boldsymbol{R}}$ is equal to a rotation of motor M_1, $\tilde{\boldsymbol{S}}$ is a rotation of M_2 and $\tilde{\boldsymbol{T}}$ is a rotation of M_3. Recall that a rotation quaternion can be described as a vector and a rotation, as shown in equation 2.7. Applying this to the three quaternions $\tilde{\boldsymbol{R}}$, $\tilde{\boldsymbol{S}}$, and $\tilde{\boldsymbol{T}}$, each one is described by the rotation angle $\beta_{1,2,3}$ and the corresponding rotational axis of each motor. Equations 5.2a, 5.2b, and 5.2c show this relation, where $\bar{\boldsymbol{n}}_{\mathbf{1,2,3}}$ are the rotational axes of each motor $M_{1,2,3}$.

$$\tilde{\boldsymbol{R}} = \cos\frac{\beta_1}{2} + \sin\frac{\beta_1}{2}\bar{\boldsymbol{n}}_{\mathbf{1}} \tag{5.2a}$$

$$\tilde{\boldsymbol{S}} = \cos\frac{\beta_2}{2} + \sin\frac{\beta_2}{2}\bar{\boldsymbol{n}}_{\mathbf{2}} \tag{5.2b}$$

$$\tilde{\boldsymbol{T}} = \cos\frac{\beta_3}{2} + \sin\frac{\beta_3}{2}\bar{\boldsymbol{n}}_{\mathbf{3}} \tag{5.2c}$$

The next step is to define $\bar{\boldsymbol{n}}_{\mathbf{1,2,3}}$, which is done by referring back to figure 5.2. If the quaternion $\tilde{\boldsymbol{Q}}$ represents a desired orientation of the CAM frame, then the motors can be rotated according to the nautical sequence to reach this orientation. The IMU and CAM frame are assumed to be in the standard position at the start of the rotation $\tilde{\boldsymbol{Q}}$, to simplify the calculations. As figure 5.2 show, M_1 will rotate the CAM frame around the negative z-axis. M_2 will then rotate the frame around the once rotated negative y-axis, and finally M_3 will rotate around the twice rotated negative x-axis. The equations 5.2a, 5.2b, and 5.2c are then changed to:

$$\tilde{\boldsymbol{R}} = \cos\frac{\beta_1}{2} - \sin\frac{\beta_1}{2}\bar{\boldsymbol{z}}_{\mathbf{1}} \tag{5.3a}$$

$$\tilde{\boldsymbol{S}} = \cos\frac{\beta_2}{2} - \sin\frac{\beta_2}{2}\bar{\boldsymbol{y}}_{\mathbf{2}} \tag{5.3b}$$

$$\tilde{\boldsymbol{T}} = \cos\frac{\beta_3}{2} - \sin\frac{\beta_3}{2}\bar{\boldsymbol{x}}_{\mathbf{3}} \tag{5.3c}$$

Since the CAM and IMU frames start in the standard position, the vectors $\bar{\boldsymbol{z}}_{\mathbf{1}}$, $\bar{\boldsymbol{y}}_{\mathbf{2}}$ and $\bar{\boldsymbol{x}}_{\mathbf{3}}$ are calculated with the following equations (see equation 2.9):

$$\bar{\boldsymbol{x}}_{\mathbf{1}} = [1\,0\,0] \tag{5.4a}$$

$$\bar{\boldsymbol{y}}_{\mathbf{1}} = [0\,1\,0] \tag{5.4b}$$

$$\bar{\boldsymbol{z}}_{\mathbf{1}} = [0\,0\,1] \tag{5.4c}$$

$$\tilde{\boldsymbol{y}}_{\mathbf{2}} = \tilde{\boldsymbol{R}} \circ \tilde{\boldsymbol{y}}_{\mathbf{1}} \circ \tilde{\boldsymbol{R}}^{\mathbf{-1}} \tag{5.4d}$$

$$\tilde{\boldsymbol{x}}_{\mathbf{3}} = \tilde{\boldsymbol{S}} \circ \tilde{\boldsymbol{x}}_{\mathbf{2}} \circ \tilde{\boldsymbol{S}}^{\mathbf{-1}} = \tilde{\boldsymbol{S}} \circ \tilde{\boldsymbol{R}} \circ \tilde{\boldsymbol{x}}_{\mathbf{1}} \circ \tilde{\boldsymbol{R}}^{\mathbf{-1}} \circ \tilde{\boldsymbol{S}}^{\mathbf{-1}} \tag{5.4e}$$

Combining these equations with equations 5.3a, 5.3b, and 5.3c yields the following results:

$$\tilde{\boldsymbol{R}} = \begin{bmatrix} \cos\frac{\beta_1}{2} \\ 0 \\ 0 \\ -\sin\frac{\beta_1}{2} \end{bmatrix} \tag{5.5}$$

$$\tilde{\boldsymbol{S}} = \begin{bmatrix} \cos\frac{\beta_2}{2} \\ -2 * \cos\frac{\beta_1}{2} * \sin\frac{\beta_1}{2} * \sin\frac{\beta_2}{2} \\ -\cos\frac{\beta_1}{2}\sin\frac{\beta_2}{2} \\ 0 \end{bmatrix} \tag{5.6}$$

$$\tilde{\boldsymbol{T}} = \begin{bmatrix} \cos\frac{\beta_3}{2} \\ -\cos\beta_1 * \cos\beta_2 * \sin\frac{\beta_3}{2} \\ 2 * \cos\frac{\beta_1}{2}\sin\frac{\beta_1}{2} * \cos\beta_2 * \sin\frac{\beta_3}{2} \\ -2 * \cos\frac{\beta_2}{2}\sin\frac{\beta_2}{2} * \sin\frac{\beta_3}{2} \end{bmatrix} \tag{5.7}$$

Combining these results with equation 5.1 gives the relation between the motor angles $\beta_{1,2,3}$ and an orientation described by the quaternion $\tilde{\boldsymbol{Q}}$:

$$\tilde{\boldsymbol{Q}} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\beta_1}{2}\cos\frac{\beta_2}{2} * \cos\frac{\beta_3}{2} - \sin\frac{\beta_1}{2}\sin\frac{\beta_2}{2} * \sin\frac{\beta_3}{2} \\ -\cos\frac{\beta_1}{2}\cos\frac{\beta_2}{2} * \sin\frac{\beta_3}{2} - \sin\frac{\beta_1}{2}\sin\frac{\beta_2}{2} * \cos\frac{\beta_3}{2} \\ \sin\frac{\beta_1}{2}\cos\frac{\beta_2}{2} * \sin\frac{\beta_3}{2} - \cos\frac{\beta_1}{2}\sin\frac{\beta_2}{2} * \cos\frac{\beta_3}{2} \\ -\sin\frac{\beta_1}{2}\cos\frac{\beta_2}{2} * \cos\frac{\beta_3}{2} - \cos\frac{\beta_1}{2}\sin\frac{\beta_2}{2} * \sin\frac{\beta_3}{2} \end{bmatrix} \tag{5.8}$$

In summation, equation 5.8 is the relation between the three motor angles $\beta_{1,2,3}$ and the resulting orientation described by a quaternion $\tilde{\boldsymbol{Q}}$. It assumes that the start position of the IMU and CAM frame is the standard position, meaning that if $\beta_{1,2,3} = 0$ then no rotation occurs and the gimbal remains in the standard position.

5.1.3 Controller objective

The goal of the controller is to keep the camera module pointing in the forward direction at all times. It does this by rotating the gimbal motors, thus completely eliminating any unwanted rotations acting on the camera module. The controller therefore needs the current orientation of the prototype as an input and outputs the correct motor angles to keep the camera module pointing forward. The current orientation is measured by the IMU as a quaternion, but some assumptions and calculations are required to use this quaternion in the controller.

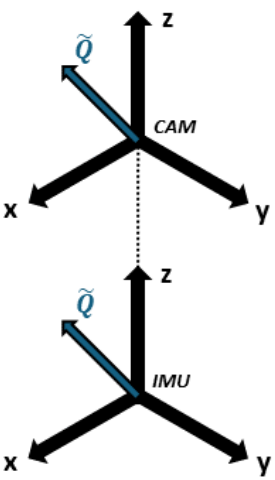


Figure 5.3 IMU and CAM frame

If the gimbal is assumed to be in the standard position, any rotation of the prototype can be described with the same quaternion $\tilde{\boldsymbol{Q}}$ for both the IMU and CAM frame. This is the case, because the frames are aligned as figure 5.3 shows. If $\tilde{\boldsymbol{Q}}$ is an unwanted rotation it means that the motors should rotate the camera module in the opposite direction of it (the direction $\tilde{\boldsymbol{Q}}^{-1}$). Since the gimbal is in the standard position, equation 5.8 is applicable. A simplification is however possible due to the prototype presented in section 4.5.3 is only a 2 DOF system, not a 3 DOF system as described by equation 5.8. The missing DOF is the result of the prototype not including motor M_3 in its design, which means that $\beta_3 = 0$ for all values of $\tilde{\boldsymbol{Q}}$. This results in the following equation:

$$\tilde{\boldsymbol{Q}} = \begin{bmatrix} \cos\frac{\beta_1}{2}\cos\frac{\beta_2}{2} \\ -\sin\frac{\beta_1}{2}\sin\frac{\beta_2}{2} \\ -\cos\frac{\beta_1}{2}\sin\frac{\beta_2}{2} \\ -\sin\frac{\beta_1}{2}\cos\frac{\beta_2}{2} \end{bmatrix} \tag{5.9}$$

This equation assumes that the motor angles are known and the quaternion is unknown, but for the controller the reverse is true. It knows the value of $\tilde{\boldsymbol{Q}}$ based upon the readings from the IMU and wants to calculate the values for $\beta_{1,2,3}$. Equation 5.9 is possible to reverse, but it is important to note that $\tilde{\boldsymbol{Q}}$ represents a 3 DOF rotation and the motor angle matrix represents a 2 DOF system, with no control over the roll axis. The calculation of $\beta_{1,2}$ is therefore only reliant on the $q_{2,3}$ parts of $\tilde{\boldsymbol{Q}}$, since these two are needed to represent either a pure yaw, or pitch rotation. Applying this to equation 5.9 yields:

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\beta_1}{2}\cos\frac{\beta_2}{2} \\ -\sin\frac{\beta_1}{2}\sin\frac{\beta_2}{2} \\ -\cos\frac{\beta_1}{2}\sin\frac{\beta_2}{2} \\ -\sin\frac{\beta_1}{2}\cos\frac{\beta_2}{2} \end{bmatrix} \rightarrow \begin{cases} \frac{q_3}{q_0} = -\tan\frac{\beta_1}{2} \\ \frac{q_2}{q_0} = -\tan\frac{\beta_2}{2} \end{cases} \rightarrow \begin{cases} \beta_1 = 2 * \arctan\frac{-q_3}{q_0} \\ \beta_2 = 2 * \arctan\frac{-q_2}{q_0} \end{cases} \tag{5.10}$$

This equation also shows that the reverse unit quaternion $\tilde{\boldsymbol{Q}}^{-1}$ will result in the same angles as $\tilde{\boldsymbol{Q}}$, but with the opposite sign ($\beta_{1,2} = -\beta_{1,2}$). Recall equation 2.8a from section 2.12 for how a unit quaternion is inverted. Furthermore, this equation allows the controller to use motor angles in its control loop instead of quaternions. Note that the calculated motor angles are always in relation to the standard position. Since the IMU bases its orientation on the earth frame, the standard position is assumed to also be aligned with the earth frame. Using equation 5.10 therefore yields the motor angles required to orient the CAM frame, according to some quaternion $\tilde{\boldsymbol{Q}}$, in relation to the standard position.

The goal of the controller is to point the camera in the forward direction. By using equation 5.10 both the current orientation of the prototype and the forward position are saved as motor angles, for example $\alpha_{1,2}$ and $\gamma_{1,2}$.

Since both $\alpha_{1,2}$ and $\gamma_{1,2}$ are in relation to the standard position, the difference between these angles ($\gamma_{1,2}$-$\alpha_{1,2}$) yields the motor angles that keep the camera pointing in the forward direction. The basic steps of the controller are then:

1. Read the IMU and converted the recorded quaternion into motor angles.
2. Calculate the difference between the forward position and the new orientation.
3. Update the motors with the new reference angles.
4. Update the forward position if necessary.

The software implementation of the controller is discussed further in the next section.

5.2 Software implementation

The complete control loop is implemented in the MCU using Arduino. The code is available in the appendix, but an overview of it is given in this section. The control loop is separated in two distinct parts: the motor angle calculation (section 5.2.1), and the motor control algorithm (section 5.2.2). The goal of the motor angle calculation is to find the reference values that the motor control algorithm then uses.

5.2.1 Motor angle calculation

To control the motor, a reference motor angle β_{ref} is needed, and how this angle is calculated by the software, is shown with a block diagram in figure 5.4.

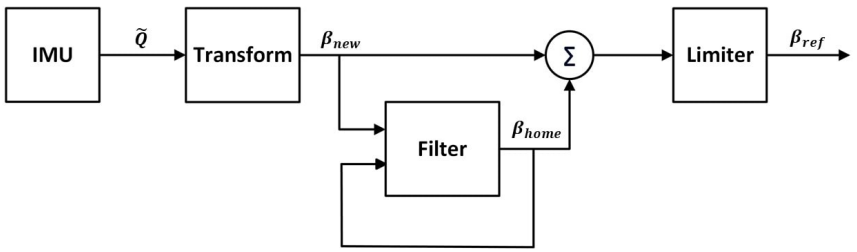


Figure 5.4 Block diagram representing the motor angle calculation

The variables shown in the block diagram are:

- $\tilde{\boldsymbol{Q}}$ - Unit quaternion measured by the IMU.
- β_{new} - Motor angles calculated from $\tilde{\boldsymbol{Q}}$ using equation 5.10.
- β_{home} - Motor angles that represent the current forward position.
- β_{ref} - Motor angles that result in the camera module pointing forward.

Each block in the diagram represent a method in the program code, which are:

- **IMU** - Method that communicates with the IMU and receives the current orientation described as a quaternion.
- **Transform** - Receives the quaternion from the IMU and converts it into motor angles using an implementation of equation 5.10.
- **Filter** - Uses the values of the current orientation β_{new} and the current forward position β_{home} to dynamically update the forward position over time.
- **Limiter** - Limits the calculated difference between β_{home} and β_{new} to ensure the result is within expected values.

In the filter method both β_{home} and β_{new} are used to dynamically update β_{home}. The current β_{home} are the motor angles that represent the quaternion, which describes the current forward position. To update the forward position, the filter method uses a moving average and a complementary filter. With the moving average the fifty latest β_{new} values are averaged, which show if a change in the forward position is occurring. In order to not update the forward position to often and to quickly, the complementary filter is used. It simply updates the forward position with the following equation:

$$\beta_{home} = 0.1 * \beta_{new} + 0.9 * \beta_{home} \tag{5.11}$$

The values of 0.1 and 0.9 are chosen by experimentation and represent how much weight β_{new} and β_{home} are given when the forward position is updated. Increasing the weight of β_{new} result in the forward position updating more frequently, but also being more sensitive to small changes of β_{new}.

As a protective measure the limiter block is added to the motor angle calculation. It limits β_{ref} to be between $\pm 45°$, which makes sure that no destructive movements are made by the gimbal motors. This limit is bigger than the estimated rotation requirement set in section 3.3, so it does not interfere with the stabilization of the gimbal system.

5.2.2 Motor control algorithm

To control the motors using the reference motor angle, a control algorithm is implemented in the MCU. Figure 5.5 provides an overview of the control loop used in this project, which is a software implementation of the FOC algorithm. The outermost loop (orange), called position motion control, employs a standard PID regulator with an added velocity limit. It receives the desired angle from the motor angle calculation in section 5.2.1 and subtracts the current motor position angle obtained from the magnetic encoder. The resulting error angle passes through the PID and velocity limit, converting it into the desired velocity. The desired velocity is then processed in the velocity motion control loop (yellow), undergoing a similar process to produce the target current.

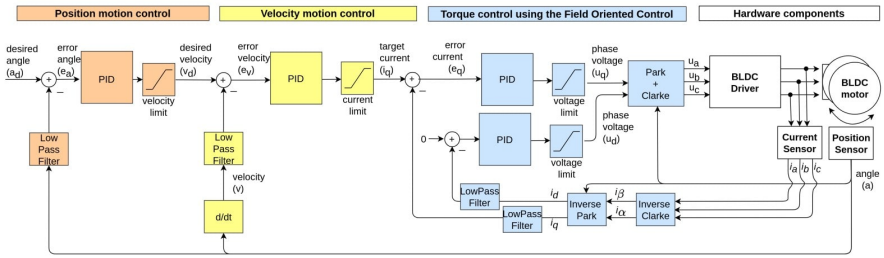


Figure 5.5 Overview of a standard control loop utilizing the Field-Oriented Control method [Skuric et al., 2022]

As described in section 2.8, to maximize generated torque and maintain it steadily, the stator magnetic field must remain 90 degrees ahead of the rotor's magnetic field. The innermost loop (blue), called the torque control, achieves this using FOC. In this loop, the phase currents (i_a, i_b, i_c) are converted into direct and quadrature currents (i_d, i_q) through inverse Clark and Park transformation. The direct current is forced to zero while the quadrature current is compared with the target current to determine the error current. This error current then passes through a PID regulator and a voltage limit to produce one of the two-phase voltages. The two-phase time-invariant voltages (u_d, u_q) are converted back into three-phase time-variant voltages (u_a, u_b, u_c) via Clark and Park transformation. Finally, the three-phase voltages are sent to the motor drivers.

The PIDs in the control loop are tuned manually, starting with the innermost loop. Each PID is tuned by looking at the step response of the motors and achieving a fast response time with little oscillation. First only the proportional part is tuned to a satisfactory degree, with the integral added after and the derivative part added last. When each part of the PID is added the previously added parts are tweaked again to achieve a better step response. After the innermost loop is tuned to satisfied degree, the next loop is added and the process is repeated.

6
Final evaluation

The final evaluation of the system involves conducting real-life tests with the prototype. The gimbal prototype attaches to a functioning body-worn camera, using its camera module to gather footage. This footage helps visualize the performance of the prototype. Additionally, the performance is compared with a commercially available gimbal product.

6.1 Evaluation setup

The entire prototype plus body-worn camera system is fastened to a person using the same holster as described in section 3.1. This is shown in figure 6.1 and the setup is used to try and mimic real-life use of the body-worn camera.

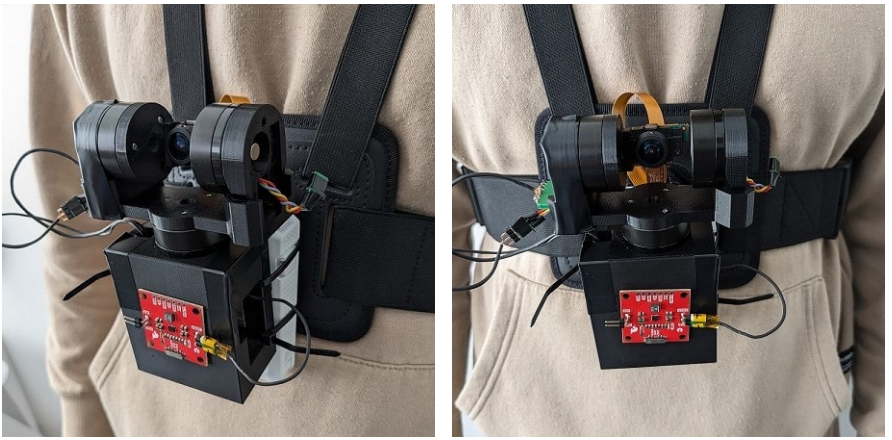


Figure 6.1 Prototype attached to wearer's chest

With the system fastened, videos are recorded as the wearer approaches a red object with the gimbal both activated and deactivated. The recordings take place outdoors in well-lit conditions, capturing footage of both walking and running towards the object. A snapshot of the recorded footage is displayed in figure 6.2, showing the beginning of a recording with the red object visible.

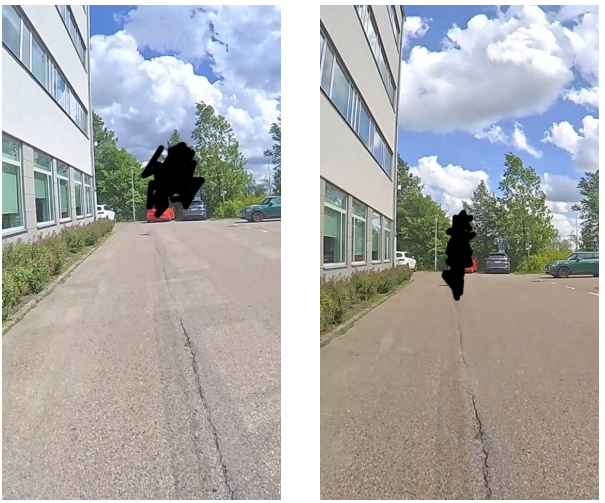
After recording, a video editing program is used to track the red object's deviation from its starting point in the first frame. By tracing the path of the object, the overall deviation in the footage is visualized, providing a clearer and better understanding of the recording's shakiness. The resulting traced paths are presented in the next section (6.2).

Figure 6.2 Snapshot from recorded footage

The setup for the final evaluation differs from the setup used during the data gathering process. This stems from difficulties with recordings done on the treadmill. A majority of the footage captured on the treadmill is obscured by a screen, which is a part of the treadmill. Any object placed in the center of the camera frame therefore has to be placed in front of this screen to be visible in the recording. This places the object at a distance of less than one meter away from the wearer. The short distance does not accurately reflect real-life use of the body-worn camera and could skew the final evaluation. Using the data gathering setup was therefore deemed to not give an accurate reflection of real-life use of the body-worn camera and the prototype is instead tested outside. A consequence of this decision is that the exact speeds used on the treadmill cannot be replicated.

6.2 Evaluation results

The results from the final evaluation of the prototype are presented in figures 6.3 and 6.4. They show the use of the final prototype in two predefined test cases (walking and running) with and without the image stabilization system activated. In the figures, the black line represents the movement patterns of the tracked object captured in the videos. This movement pattern is a result of vibrations and rotations that occur during walking and running. Figure 6.3 displays the results of walking with the stabilization system activated and deactivated, while figure 6.4 shows the results for running with the same setup. It is important to note that the motors are made rigid when the stabilization system is deactivated. The motors are therefore not allowed to rotate freely, which ensures a similar behaviour as wearing the normal non stabilized body-worn camera. This result in a more accurate comparison between the constructed prototype and the actual body-worn camera.



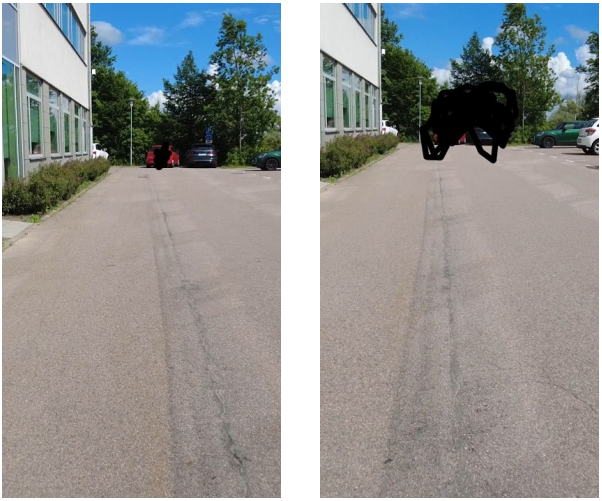
(a) Without stabilization **(b)** With stabilization

Figure 6.3 Tracking the movement while walking

In addition to testing the prototype, a commercially available gimbal camera is also evaluated using the previously described final evaluation method. Although this gimbal camera is designed as a handheld device, it features a camera module and lens of similar size to the prototype. By fastening the gimbal camera to the same holster used with the prototype, it simulates how the commercially available gimbal could act as a body-worn camera. The results of these tests are found in figure 6.5, with the black line once again representing the movement pattern of the red object.

(a) Without stabilization **(b)** With stabilization

Figure 6.4 Tracking the movement while running



(a) Walking **(b)** Running

Figure 6.5 Tracking the movement of a commercially available gimbal camera

6.3 Evaluation discussion

The results from the final evaluation tests show overall improved stabilization using either the prototype or the commercially available gimbal. Starting by only looking at the prototype's performance, the most improvement is seen while running (figure 6.4). Quite clearly the area of the black line is smaller in both the vertical and horizontal direction. This indicates an improvement in the reduction of yaw and pitch rotations acting on the camera module. In figure 6.3, where the test is conducted while walking, the horizontal stabilization is improved while the vertical stabilization does not show similar progress.

A significant reason for the vertical movement is the impact of each step the wearer takes. When the feet hit the ground, major vibrations occur due to the sudden change in speed and direction. This primarily affects the pitch rotation. In the stabilized system, the camera module is movable which makes it more receptive to these disturbances. This is because its inertia is lower than the inertia of the fixed camera module, making it easier to move. The worse stabilization in the vertical direction when walking is likely due to the low inertia of the camera module combined with an incorrectly tuned controller for the innermost motor. The controller needs to be more sensitive and faster to correct the smaller more rapid disturbances affecting the camera module due to its lower inertia. A more sensitive controller is however more likely to be unstable or react to noise in the system, which takes time to correctly tune.

The stabilized footage is also more centered during both walking and running. It is especially clear in figure 6.4. The black path is more compact and centered in the frame. It has also removed the upside-down u-shaped form seen in the unstabilized footage. This improvement demonstrates the effectiveness of the dynamic home position implemented in section 5.2.1 and the ability of the controller to maintain a forward-pointing orientation.

Comparing the result of the constructed prototype with the commercially available gimbal shows a vast improvement, with the gimbal providing even better stabilized footage. This further supports the achievability and importance of implementing a MIS system to eliminate disturbances and vibrations acting on the body-worn camera. It is primarily with the vertical stabilization that the gimbal outperforms the prototype, indicating again the need to improve the controller for the pitch motor. Another interesting observation is the presence of the upside-down u-shape in the footage from the commercially available gimbal.

7
Conclusions

7.1 General

In conclusion, the thesis has successfully designed, assembled and evaluated an image stabilization system for a body-worn camera. The final system presented makes noticeable improvements in the stability of the captured footage, proving that the chosen concept works for its intended use. The viability of using a MIS system is furthered supported by the evidence gathered with the commercially available gimbal.

A notable undeveloped feature of the prototype is the third axis of rotation, which meant that all three axes could not be stabilized simultaneously. It remains unclear how big of an impact stabilization of all three axes have compared to stabilizing only two axes. Although the commercially available gimbal is a 3 DOF system compared to the prototype's 2 DOF, it is hard to draw any conclusion of the impact this extra DOF has on the stabilization. There are too many unknown differences between the gimbal and the prototype to clearly determine if the gimbal outperforms the prototype due too its extra DOF.

Alongside the constructed prototype, the rotational motions affecting the body-worn camera during use have been studied and quantified. This was accomplished through both a literature study and experimentation, which can hopefully aid future developments in this area.

7.2 Further work

To further improve the stabilization system, some areas of improvement have been identified. One potential improvement is to add the third rotational axle as intended. This would add compensation for disturbances in the roll angle. While the roll angle, as described before, has the least impact on the stabilization it still influences the system, making this addition worth considering.

Another major improvement would be to thoroughly tune the PID settings for the entire system. Since there are four PID-regulators in the system, each with three parameters, a lot of interdependencies exist between them. A possible solution is to create a simulation of the system to more easily test all possible parameter combinations. The optimal values identified in the simulation can then be used as guidelines for real-world testing and tuning.

An important component of a gimbal system is its motors. For optimal stabilization, the motors need to be custom-made for the specific build. In this project, there is room for improvement since the motors used are off-the-shelf solutions and not specifically designed for this prototype. Another potential enhancement, which has not been tested, involves adding a second IMU on the camera module. This additional IMU would collect data on the camera module's specific movements and could be used to compensate for vibrations occurring specifically on the module.

Another improvement not explored in this work is to combine different stabilization systems for better performance. For example, a MIS system could take care of larger and slower disturbances, while either an OIS or EIS system could address the smaller and faster vibrations. This hybrid approach would optimize stabilization by leveraging the strengths of each system and potentially help counter each systems' weaknesses.

Bibliography

ams OSRAM (2024). *ams AS5048A High-Resolution Position Sensor*. Available at: https://ams-osram.com/products/sensors/position-sensors/ams-as5048a-high-resolution-position-sensor. (accessed on 2024-05-30).

Ardakani, H. A. and Bridges, T. J. (2010). *Review of the 3-2-1 Euler Angles: a yaw–pitch–roll sequence*. University of Surrey.

Bryngelsson, S. and Gustafsson, J. (2023). *Image Stabilization for Body-Worn Cameras*. Master's Thesis. Lund University.

Challis, J. H. (2021). *Experimental Methods in Biomechanics*. Springer, Cham.

Doe, J. (2024). *What Is a Gimbal? Everything You Want to Know*. Available at: https://www.hollyland.com/blog/tips/what-is-a-gimbal. (accessed on 2024-05-22).

EDN (2010). *Hardware-controlled brushless dc motors ease the burden on CPUs*. Available at: https://www.edn.com/hardware-controlled-brushless-dc-motors-ease-the-burden-on-cpus/. (accessed on 2024-04-29).

Electrical4U (2024). *What is Field Oriented Control?* Available at: https://www.electrical4u.com/field-oriented-control/. (accessed on 2024-05-10).

Electricity & Magnetism (2024). *Magnetometers*. Available at: https://www.electricity-magnetism.org/magnetometers/. (accessed on 2024-02-22).

Ericco Inertial System (2023). *What's the advantages and disadvantages of MEMS gyroscope?* Available at: https://www.ericcointernational.com/application/whats-the-advantages-and-disadvantages-of-mems-gyroscope.html. (accessed on 2024-02-14).

Flores, P. (2015). *Concepts and Formulations for Spatial Multibody Dynamics*. Springer, Cham.

Golik, B. (2006). *Development of a Test Method for Image Stabilizing Systems*. Diploma Thesis. University of Cologne.

Goodwin, D. (2023). *Field-oriented Control (Vector Control) for Brushless DC Motors*. Available at: https://control.com/technical-articles/field-oriented-control-vector-control-for-brushless-dc-motors/. (accessed on 2024-05-13).

Gunasekaran, B. (2019). *IMU Sensors: Everything You Need To Know!* Available at: https://embeddedinventor.com/what-is-an-imu-sensor-a-complete-guide-for-beginners/. (accessed on 2024-02-26).

Hansen, S. (2023). *Large angle OIS compensation with SMA camera products*. Available at: https://www.cambridgemechatronics.com/en/news/resources-blogs/large-angle-ois-compensation-sma-camera-products/. (accessed on 2024-03-07).

Haslwanter, T. (2018). *3D Kinematics*. Springer, Cham.

Hytera (2023). *VM780 Body Camera*. Available at: https://www.hytera.com/en/product-new/body-worn-camera/body-worn-camera/vm780.html. (accessed on 2024-06-05).

Jenish, C. (2023). *Performance Analysis of BLDC Motors and its Various Control Strategies*.

Manelius, E. (2023). *Improving Image Stabilization in Modern Surveillance Cameras*. Master's Thesis. Lund University.

Nicholson, A. and Summersby, A. (2024). *Image Stabilisation*. Available at: https://www.canon.se/pro/infobank/image-stabilisation-lenses/. (accessed on 2024-05-22).

Omega Engineering (n.d.). *How to Measure Acceleration?* Available at: https://www.omega.com/en-us/resources/accelerometers. (accessed on 2024-02-21).

Ordro (n.d.). *New ORDRO EP8 FPV Wearable Action 4K POV Camcorder Vlog Camera for Youtuber Cam*. Available at: https://ordro.online/products/camcamcorder-ep8?_pos=2&_sid=ed65c1f6c&_ss=r. (accessed on 2024-06-05).

Passaro, V. M. N., Cuccovillo, A., Vaiani, L., De Carlo, M., and Campanella, C. E. (2017). *Gyroscope Technology and Applications: A Review in the Industrial Perspective*. Available at: https://doi.org/10.3390/s17102284. (accessed on 2024-02-12).

ROHM Semiconductor (2013). *Optical Image Stabilization (OIS)*.

Rosa, F. L., Virzi, M. C., Bonaccorso, F., and Branciforte, M. (n.d.). *Optical Image Stabilization (OIS)*. STMicroelectronics.

Skuric, A., Bank, H. S., Unger, R., Williams, O., and González-Reyes, D. (2022). *SimpleFOC: A Field Oriented Control (FOC) Library for Controlling Brushless Direct Current (BLDC) and Stepper Motors.* Available at: https://doi.org/10.21105/joss.04232. (accessed on 2024-04-26).

Souza, M. R. and Pedrini, H. (2018). *Digital video stabilization based on adaptive camera trajectory smoothing.* Available at: https://doi.org/10.1186/s13640-018-0277-7. (accessed on 2024-02-01).

sparkfun (2024a). *SparkFun Buck-Boost Converter.* Available at: https://www.sparkfun.com/products/15208. (accessed on 2024-06-03).

sparkfun (2024b). *Three Phase Brushless Gimbal Stabilizer Motor.* Available at: https://www.sparkfun.com/products/20441. (accessed on 2024-05-30).

STMicroelectronics (2024a). *STEVAL-GMBL02V1.* Available at: https://www.st.com/en/evaluation-tools/steval-gmbl02v1.html. (accessed on 2024-05-29).

STMicroelectronics (2024b). *STEVAL-MKSBOX1V1.* Available at: https://www.st.com/en/evaluation-tools/steval-mksbox1v1.html. (accessed on 2024-05-22).

Strout, Z. (2023). *How Does IMU Sensor Fusion Work?* Available at: https://www.sagemotion.com/blog/how-does-imu-sensor-fusion-work. (accessed on 2024-04-23).

TDK InvenSense (2017). *ICM-20948.* Available at: https://cdn.sparkfun.com/assets/7/f/e/c/d/DS-000189-ICM-20948-v1.3.pdf. (accessed on 2024-05-30).

Ulusoy, M. (2020). *Understanding Field-Oriented Control — Motor Control, Part 4.* Available at: https://se.mathworks.com/videos/motor-control-part-4-understanding-field-oriented-control-1587967749983.html. (accessed on 2024-05-13).

A
MATLAB code

Three main scripts are used throughout this work to process the data gathered, and help with calculations and plotting of graphs. Each script is presented in a separate subsection.

A.1 QuaternionAngles

This code reads the collected quaternion data in each excel file and plots the graphs for:

- Recorded quaternions over time
- Angular rotation around each axis over time
- Angular velocity around each axis over time
- Angular acceleration around each axis over time

The code is used to produce the graphs 3.3a to 3.3c, and 3.5a to 3.7c in section 3.2.

```
%Start of Code
clc
clear
close all

%Names of each Excel file containing quaternion data
FileNames = ["Speed3","Speed8","Speed14"];
for k = 1:length(FileNames)
    % Define the file paths
    FilePath = "C:\Users\jakob\Documents\MATLAB\Thesis\" + FileNames(k) +
        ".xlsx"; % Add more file paths as needed

    % Import data from the Excel file
    Data = readtable(FilePath);

    % Get unique time values
    [temp,Indices] = unique(Data.Time);
    UniqueData = Data(Indices,:);
    Time = temp/1000;

    % Calculate quaternion
    Quaternions = calculateQuaternion(UniqueData, ["qs","qi","qj","qk"]);
    % Create a new time vector with regular intervals
    NewTime = linspace(min(Time), max(Time), length(Time))';
```

```
% Initialize array for interpolated quaternions
NewQuaternions = quaternion.zeros(size(NewTime));

% Loop over each new time point
idx = 1;
for i = 1:length(NewTime)
    %Find the next original time point that is larger than NewTime(i)
    while Time(idx+1) < NewTime(i)
        idx = idx + 1;
    end
    % If the new time point is exactly equal to an original time point,
        use the original quaternion
    if NewTime(i) == Time(idx)
        NewQuaternions(i) = Quaternions(idx);
    else
        % Otherwise, interpolate between the two original quaternions
        t = (NewTime(i) - Time(idx)) / (Time(idx+1) - Time(idx));
        NewQuaternions(i) = slerp(Quaternions(idx), Quaternions(idx+1),
            t);
    end
end
Quaternions = NewQuaternions;
Time = NewTime;

%Find the mean rotation during the start to get the home position
IndiceStartingPosition = find(Time > 5 & Time < 15);
QuaternionsStart = Quaternions(min(IndiceStartingPosition):max(
    IndiceStartingPosition));
QuaternionHome = meanrot(QuaternionsStart);

x = [1 0 0];
y = [0 1 0];
z = [0 0 1];

%Find the coordinates for the three axis of the home position in the
    global
%coordinate system.
PointHomeX = rotatepoint(QuaternionHome,x);
PointHomeY = rotatepoint(QuaternionHome,y);
PointHomeZ = rotatepoint(QuaternionHome,z);

%Rotate the point by each recorded quaternion, and get the new
    coordinates
%in the global system
PointsRotatedX = rotatepoint(Quaternions,x);
PointsRotatedY = rotatepoint(Quaternions,y);
PointsRotatedZ = rotatepoint(Quaternions,z);

%Change the frame of reference to be the home position and get the
%coordinates for each rotated point.
PointsHomeX = rotateframe(QuaternionHome,PointsRotatedX);
PointsHomeY = rotateframe(QuaternionHome,PointsRotatedY);
PointsHomeZ = rotateframe(QuaternionHome,PointsRotatedZ);

%Calculate the yaw, pitch, and roll angles. Yaw rotation around x,
    pitch
%rotation around y, roll rotation around z.
Yaw = zeros(length(PointsHomeX),2);
Pitch = zeros(length(PointsHomeY),2);
Roll = zeros(length(PointsHomeZ),2);
for n = 1 : length(PointsHomeX)
    %Calculate the angles between the x-axis and the points in the xz-
        and
    %xy-plane
    Pitch(n,1) = vectorAngle(x,[PointsHomeX(n,1),0,PointsHomeX(n,3)],y)
        ;
    Roll(n,1) = vectorAngle(x,[PointsHomeX(n,1),PointsHomeX(n,2),0],z);

    %Calculate the angles between the y-axis and the points in the xy-
        and
    %yz-plane
    Yaw(n,1) = vectorAngle(y,[0,PointsHomeY(n,2),PointsHomeY(n,3)],x);
    Roll(n,2) = vectorAngle(y,[PointsHomeY(n,1),PointsHomeY(n,2),0],z);
```

```
        %Calculate the angles between the z-axis and the points in the yz-
            and
        %zx-plane
        Yaw(n,2) = vectorAngle(z,[0,PointsHomeZ(n,2),PointsHomeZ(n,3)],x);
        Pitch(n,2) = vectorAngle(z,[PointsHomeZ(n,1),0,PointsHomeZ(n,3)],y)
            ;
    end
    Angles = [(Yaw(:,1)+Yaw(:,2))/2 (Pitch(:,1)+Pitch(:,2))/2 (Roll(:,1)+
        Roll(:,2))/2];

    [AngVel, VelMidPoints] = angleVelocity(Quaternions, Time);
    [AngAcc, AccMidPoints] = angleAcceleration(AngVel, VelMidPoints);
    TimeTot = [Time,VelMidPoints,AccMidPoints];
    AngAcc = AngAcc*pi/180;

    plotRotVelAcc(Angles, AngVel, AngAcc, TimeTot, FileNames(k));
    plotQuaternion(Quaternions, Time, FileNames(k));

end

% Function to calculate quaternion
function Q = calculateQuaternion(data, inputVariables)
temp = rowfun(@quaternion, data, "InputVariables", inputVariables, "
    OutputVariableNames", "q");
Q = temp{:,"q"};
Q = normalize(Q);
end

%Function to caclulate the angle between two vectors v1 and v2 around the
%vector n (n must be vector that is not in the plane of v1 and v2).
function a = vectorAngle(v1,v2,n)
x = cross(v1,v2);
%The dot product of x and n tells us if the angle around n should be
%calculated clockwise or counter clockwise
c = sign(dot(x,n))*norm(x);
a = atan2d(c,dot(v1,v2));
end

% Function to calculate angular velocity
function [AngVel, VelMidPoints] = angleVelocity(Q, T)
dt = mean(diff(T));
VelMidPoints =T-dt/2;
AngVel=angvel(Q,dt,'point')*180/pi;
AngVel = [0,0,0;AngVel(2:end,:)];
end

% Function to calculate angular acceleration
function [AngAcc, AccMidPoints] = angleAcceleration(AngVel, VelMidPoints)
dt = mean(diff(VelMidPoints));
AccMidPoints = VelMidPoints - dt/2;
AngAcc = [0,0,0;diff(AngVel)]./dt;
end

% Function to plot quaternions
function plotQuaternion(Q, T, FileName)
FigQuat = figure("Name","Quaternions");
[qs,qi,qj,qk] = parts(Q);
t = tiledlayout(4,1,TileSpacing="none");
xlabel(t,"Time (s)")

nexttile
plot(T, qs,Color="m")
legend("q0",Location="northwest");
xlim([9 51])
ylim([0.12 0.48])

nexttile
plot(T, qi,Color="b")
legend("q1",Location="northwest");
xlim([9 51])
ylim([0.54 0.81])
```

```
nexttile
plot(T, qj,Color="r")
legend("q2",Location="northwest");
xlim([9 51])
ylim([0.18 0.46])

nexttile
plot(T, qk,Color="g")
legend("q3",Location="northwest");
xlim([9 51])
ylim([-0.71 -0.41])

saveas(FigQuat, FileName + "\Quaternions.png")
end

%Function to plot rotation, angular velocity and angular acceleration
function plotRotVelAcc(Rot,Vel,Acc,T, FileName)
FigXAxis = figure('Name', 'X-axis');
helpPlotData(T, Rot(:,1), Vel(:,1), Acc(:,1), "b");
saveas(FigXAxis, FileName + "\XAxis.png")

FigYAxis = figure('Name', 'Y-axis');
helpPlotData(T, Rot(:,2), Vel(:,2), Acc(:,2), "r");
saveas(FigYAxis, FileName + "\YAxis.png")

FigZAxis = figure('Name', 'Z-axis');
helpPlotData(T, Rot(:,3), Vel(:,3), Acc(:,3), "g");
saveas(FigZAxis, FileName + "\ZAxis.png")
end

function helpPlotData(T, RotVec, VelVec, AccVec,color)
TRot=T(:,1);
TVel=T(:,2);
TAcc=T(:,1);
t = tiledlayout(3,1,TileSpacing="none");
xlabel(t,"Time (s)")

nexttile;
plot(TRot, RotVec,Color=color);
xlim([19 51]);
ylim([-31 35]);
ylabel("Angle (deg)")

nexttile;
plot(TVel, VelVec,Color=color);
xlim([19 51]);
ylim([-350 350]);
ylabel("Velocity (deg/s)")

nexttile;
plot(TAcc, AccVec,Color=color);
xlim([19 51]);
ylim([-310 310]);
ylabel("Acceleration (rad/s^2)")
end
```

A.2 QuaternionSphere

This code reads the collected quaternion data in each excel file and plots the change in coordinates for each coordinate axis as points on a sphere. The code is used to produce the graphs 3.4a, 3.4b, and 3.4c in section 3.2.2.

```
%Start of Code
clc
clear
close all
```

```
%Names of each Excel file containing quaternion data
FileNames = ["Speed3","Speed8","Speed14"];

for k = 1:length(FileNames)
    % Define the file paths
    FilePath = "C:\Users\jakob\Documents\MATLAB\Thesis\" + FileNames(k) +
        ".xlsx"; % Add more file paths as needed

    % Import data from the Excel file
    Data = readtable(FilePath);

    % Get unique time values
    [temp,Indices] = unique(Data.Time);
    UniqueData = Data(Indices,:);
    Time = temp/1000;

    % Calculate quaternion
    Quaternions = calculateQuaternion(UniqueData, ["qs","qi","qj","qk"]);

    %Find the mean rotation during the start to get the home position
    IndiceStartingPosition = find(Time > 5 & Time < 15);
    QuaternionsStart = Quaternions(min(IndiceStartingPosition):max(
        IndiceStartingPosition));
    QuaternionHome = meanrot(QuaternionsStart);

    x = [1 0 0];
    y = [0 1 0];
    z = [0 0 1];

    %Find the coordinates for the three axis of the home position in the
        global
    %coordinate system.
    PointHomeX = rotatepoint(QuaternionHome,x);
    PointHomeY = rotatepoint(QuaternionHome,y);
    PointHomeZ = rotatepoint(QuaternionHome,z);

    %Rotate the point by each recorded quaternion, and get the new
        coordinates
    %in the global system
    PointsRotatedX = rotatepoint(Quaternions,x);
    PointsRotatedY = rotatepoint(Quaternions,y);
    PointsRotatedZ = rotatepoint(Quaternions,z);

    %Change the frame of reference to be the home position and get the
    %coordinates for each rotated point.
    PointsHomeX = rotateframe(QuaternionHome,PointsRotatedX);
    PointsHomeY = rotateframe(QuaternionHome,PointsRotatedY);
    PointsHomeZ = rotateframe(QuaternionHome,PointsRotatedZ);

    plotSphere(PointsRotatedX, PointsRotatedY, PointsRotatedZ, FileNames(k)
        );
end

% Function to calculate quaternion
function Q = calculateQuaternion(data, inputVariables)
temp = rowfun(@quaternion, data, "InputVariables", inputVariables, "
    OutputVariableNames", "q");
Q = temp{:,"q"};
end

%Function to plot coordinates of all coordinate axes for each quaternion.
function plotSphere(PointsX,PointsY,PointsZ,FileName)
FigSphere = figure("Name","Sphere");
[X,Y,Z] = sphere;
surf(X,Y,Z,FaceColor=[0.57 0.57 0.57])
hold on
scatter3(PointsX(:,1),PointsX(:,2),PointsX(:,3),"filled")
scatter3(PointsY(:,1),PointsY(:,2),PointsY(:,3),"filled")
scatter3(PointsZ(:,1),PointsZ(:,2),PointsZ(:,3),"filled")
colororder(["b" "r" "g"]);
view([0 -90])
axis equal
hold off
```

```
title("X-, Y-, Z-coordinates");
xlabel("X");
ylabel("Y");
zlabel("Z");
saveas(FigSphere,FileName + "\Sphere1.png")
view([0 0])
saveas(FigSphere,FileName + "\Sphere2.png")
end
```

A.3 QuaternionToMotorAngle

This code is based upon the equations in section 5.1.2, and shows how the result of equation 5.8 is calculated.

```
%Start of Code
clear
clc

%Variables for the three motor angles.
syms beta1 beta beta3

% Coordinates of the three original coordinate axes.
X1 = [0 1 0 0];
Y1 = [0 0 1 0];
Z1 = [0 0 0 1];

%First quaternion rotation.
R = [cos(beta1/2), 0, 0,0] - sin(beta1/2)*Z1;

%Calculating new coordinates of y-axis
Y2 = calcy2(R,Y1);

%Second quaternion rotation
S = [cos(beta/2) 0 0 0] - sin(beta/2)*Y2;

%Calculating new coordinates of y-axis
X3 = calcx3(S,R,X1);

%Third quaternion rotation
T = [cos(beta3/2) 0 0 0] - sin(beta3/2)*X3;

%Combining all three rotations
Q(beta1,beta,beta3) = simplify(quatmulti(T,quatmulti(S,R)));

%Simplification of Q to make it more readable
QSimple(beta1,beta,beta3) = [cos(beta1/2)*cos(beta/2)*cos(beta3/2) - sin(
    beta1/2)*sin(beta/2)*sin(beta3/2);
- cos(beta1/2)*cos(beta/2)*sin(beta3/2) - cos(beta3/2)*sin(beta1/2)*sin(
    beta/2);
  cos(beta/2)*sin(beta1/2)*sin(beta3/2) - cos(beta1/2)*cos(beta3/2)*sin(
    beta/2);
- cos(beta/2)*cos(beta3/2)*sin(beta1/2) - cos(beta1/2)*sin(beta/2)*sin(
    beta3/2)];

%Calculates the multiplication of two quaternions using matrices S = Q * R
function S = quatmulti(q,r)
Q = [q(1), -q(2), -q(3), -q(4); q(2), q(1), -q(4), q(3);
    q(3), q(4), q(1), -q(2); q(4), -q(3), q(2), q(1)];
QR = Q * r.';
S = QR';
end

%Inverses a quaternion
function Qinv = quatinv(Q)
Qinv = [Q(1), -Q(2), -Q(3), -Q(4)];
end

%Calculates the new value of the y-axis based upon Y2 = R*Y1*R^-1
```

```
function Y2 = calcy2(R,Y1)
QY1 = quatmulti(R,Y1);
Qinv = quatinv(R);
Y2 = quatmulti(QY1,Qinv);
end

%Calculates the new value of the x-axis based upon X3 = S*X2*S^-1
function X3 = calcx3(S,R,X1)
QX1 = quatmulti(R,X1);
Qinv = quatinv(R);
X2 = quatmulti(QX1,Qinv);
RX2 = quatmulti(S,X2);
Rinv = quatinv(S);
X3 = quatmulti(RX2,Rinv);
end
```

B
Arduino code

The code used to program the MCU is written using Arduino and is an implementation of the controller design described in section 5.2. The following Arduino packages, are used by the code:

- **Arduino Simple Field Oriented Control (FOC)**
- **SPI**
- **SparkFun 9DoF IMU Breakout - ICM 20948**
- **RunningAverage**

```
#include "SimpleFOC.h"
#include "SPI.h"
#include "SimpleFOCDrivers.h"
#include "encoders/as5048a/MagneticSensorAS5048A.h"
#include "ICM_20948.h"
#include "RunningAverage.h"

#define LEDPIN PA7
#define CS1 PA15
#define CS2 PA8
#define AD0_VAL 1

int count = 0;
bool mode = 0;
double roll = 0, pitch = 0, yaw = 0;
double qw, qx, qy, qz;
double quat[4] = { 1, 0, 0, 0 };
double motor_angles_old[2] = { 0.0, 0.0 };
double motor_angles_new[2] = { 0.0, 0.0 };
double motor_angles_res[2] = { 0.0, 0.0 };
double angle_offset1 = 62 * PI / 90;
double angle_offset2 = 58 * PI / 90;

//create running average array with size 50
RunningAverage avgYaw(50);
RunningAverage avgPitch(50);

//create IMU instance - I2C
ICM_20948_I2C myICM;
icm_20948_DMP_data_t data;

//create encoder instances - SPI
MagneticSensorAS5048A sensor1(CS1);
MagneticSensorAS5048A sensor2(CS2);

//create BLDC motor and driver instances
BLDCMotor motor1 = BLDCMotor(7);
BLDCDriver3PWM driver1 = BLDCDriver3PWM(PC0, PC1, PC2, PC13);
BLDCMotor motor2 = BLDCMotor(7);
BLDCDriver3PWM driver2 = BLDCDriver3PWM(PA0, PA1, PA2, PC14);
```

```
41
42
43 void setup() {
44
45   //initialise led pin
46   pinMode(LEDPIN, OUTPUT);
47
48   //initialise IMU
49   initI2C();
50
51   calibrate();
52
53   SPI.setMOSI(PC12);
54   SPI.setMISO(PC11);
55   SPI.setSCLK(PC10);
56   SPI.begin();
57
58   //initialise position sensor (encoder)
59   sensor1.init();
60   sensor2.init();
61
62   //link motor to position sensor
63   motor1.linkSensor(&sensor1);
64   motor2.linkSensor(&sensor2);
65
66   //initialise driver
67   driver1.init();
68   driver2.init();
69
70   //link motor to driver
71   motor1.linkDriver(&driver1);
72   motor2.linkDriver(&driver2);
73
74   //power supply voltage [V]
75   driver1.voltage_power_supply = 8;
76   driver2.voltage_power_supply = 8;
77
78   //Set motion control type (torque, velocity or angle)
79   motor1.controller = MotionControlType::angle;
80   motor2.controller = MotionControlType::angle;
81
82   //Set modulation algorithm (SinePWM or SpaceVectorPWM)
83   motor1.foc_modulation = FOCModulationType::SpaceVectorPWM;
84   motor2.foc_modulation = FOCModulationType::SpaceVectorPWM;
85
86   setMotorControlParameters();
87
88   //Initialise motor
89   motor1.init();
90   motor2.init();
91
92   //Align encoder and start FOC
93   motor1.initFOC();
94   motor2.initFOC();
95
96   _delay(1000);
97 }
98
99 void loop() {
100   //call the FOC algorithm
101   motor1.loopFOC();
102   motor2.loopFOC();
103
104   //move the motors to the target angle according to the FOC algorithm
105   motor1.move();
106   motor2.move();
107
108   //get the current orientation of the device
109   read_imu(motor_angles_new);
110
111   //calculate the new target angles
112   motor_angles_res[0] = motor_angles_old[0] - motor_angles_new[0];
113   motor_angles_res[1] = motor_angles_old[1] - motor_angles_new[1];
```

```
114   motor_angles_res[0] = constrain(motor_angles_res[0],-PI/4, PI/4);
115   motor_angles_res[1] = constrain(motor_angles_res[1],-PI/4, PI/4);
116
117   motor1.target = motor_angles_res[0] + angle_offset1;
118   motor2.target = motor_angles_res[1] + angle_offset2;
119 }
120
121 //initialize I2C and set the correct SDA and SCL pins for the board
122 void initI2C() {
123   Wire.setSDA(PB9);
124   Wire.setSCL(PB8);
125   Wire.begin();
126   Wire.setClock(400000);
127
128   bool initialized = false;
129
130   while (!initialized) {
131     myICM.begin(Wire, AD0_VAL);
132     if (myICM.status != ICM_20948_Stat_Ok) {
133       delay(500);
134     } else {
135       initialized = true;
136     }
137   }
138
139   bool success = true;
140   success &= (myICM.initializeDMP() == ICM_20948_Stat_Ok);
141   success &= (myICM.enableDMPSensor(INV_ICM20948_SENSOR_ROTATION_VECTOR) ==
          ICM_20948_Stat_Ok);
142   success &= (myICM.setDMPODRrate(DMP_ODR_Reg_Quat9, 0) ==
          ICM_20948_Stat_Ok);
143   success &= (myICM.enableFIFO() == ICM_20948_Stat_Ok);
144   success &= (myICM.enableDMP() == ICM_20948_Stat_Ok);
145   success &= (myICM.resetDMP() == ICM_20948_Stat_Ok);
146   success &= (myICM.resetFIFO() == ICM_20948_Stat_Ok);
147 }
148
149 void setMotorControlParameters() {
150   //Velocity controller parameters for motor 1
151   motor1.PID_velocity.P = 0.5;
152   motor1.PID_velocity.I = 5.0;
153   motor1.PID_velocity.D = 0.001;
154   motor1.PID_velocity.output_ramp = 1000;
155   motor1.LPF_velocity.Tf = 0.01;  //10ms
156   motor1.voltage_limit = 7.4;     //volt
157   motor1.current_limit = 2; //amps
158
159   //Angle controller parameters for motor 1
160   motor1.P_angle.P = 30;
161   motor1.P_angle.I = 10;
162   // motor1.P_angle.D = 0;
163   // motor1.P_angle.output_ramp = 10000; //rad/s^2
164   // motor1.LPF_angle.Tf = 0;
165   motor1.velocity_limit = 40;  //rad/s
166
167   //Velocity controller parameters for motor 2
168   motor2.PID_velocity.P = 0.1;
169   motor2.PID_velocity.I = 1.0;
170   motor2.PID_velocity.D = 0.0001;
171   motor2.PID_velocity.output_ramp = 1000;
172   motor2.LPF_velocity.Tf = 0.01;  //10ms
173   motor2.voltage_limit = 7.4;     //volt
174   motor2.current_limit = 2; //amps
175
176   //Angle controller parameters for motor 2
177   motor2.P_angle.P = 20;
178   motor2.P_angle.I = 5;
179   // motor2.P_angle.D = 0;
180   // motor2.P_angle.output_ramp = 10000; //rad/s^2
181   // motor2.LPF_angle.Tf = 0;
182   motor2.velocity_limit = 40;  //rad/s
183 }
184
```

```
185 //gives the IMU time to calibrate itself, and reads the starting angles
186 void calibrate() {
187   digitalWrite(LEDPIN, HIGH);
188   int time = millis();
189   while (millis() - time < 2000)
190     ;
191   digitalWrite(LEDPIN, LOW);
192   read_imu(motor_angles_old);
193 }
194
195 //reads the IMU and update the current and previous orientation of the
      device
196 void read_imu(double* angle) {
197   myICM.readDMPdataFromFIFO(&data);
198   if ((myICM.status == ICM_20948_Stat_Ok)) {
199     if ((data.header & DMP_header_bitmap_Quat9) > 0) {
200       qx = ((double)data.Quat9.Data.Q1) / 1073741824.0;  // Convert to
            double. Divide by 2^30
201       qy = ((double)data.Quat9.Data.Q2) / 1073741824.0;  // Convert to
            double. Divide by 2^30
202       qz = ((double)data.Quat9.Data.Q3) / 1073741824.0;  // Convert to
            double. Divide by 2^30
203       qw = sqrt(1.0 - ((qx * qx) + (qy * qy) + (qz * qz)));
204
205       quat[0] = qw;
206       quat[1] = qx;
207       quat[2] = qy;
208       quat[3] = qz;
209       quat_to_angle(quat, angle);
210
211       avgYaw.addValue(angle[0]);
212       avgPitch.addValue(angle[1]);
213
214       motor_angles_old[0] = 0.1*avgYaw.getAverage() + 0.9*motor_angles_old
            [0];
215       motor_angles_old[1] = 0.1*avgPitch.getAverage() + 0.9*
            motor_angles_old[1];
216     }
217   }
218 }
219
220 //converts a quaternion into motor angles
221 void quat_to_angle(double* quat, double* angle) {
222   angle[0] = 2 * atan(-quat[3] / quat[0]);
223   angle[1] = 2 * atan(quat[2] / quat[0]);
224 }
```