Eötvös Loránd University

Faculty of Informatics

Dept. of Software Technology and Methodology

Estimation of gravity vector from IMU data

Supervisor:
Dr. Levente Hajder
Associate Professor

Author:
Botond A. Nás
Computer Science for Autonomous Systems MSc

Budapest, 2022

Contents

Abstract

Inertial measurement units (IMUs) enable velocity, orientation and position estimation with the help of gyroscopes, accelerometers and sometimes magnetometers. Errors in the sensor measurements have to be mitigated by taking advantage of the complementary properties of gyroscopes, accelerometers and magnetometers with the help of sensor filtering and fusion algorithms. In these algorithms, the accelerometer and magnetometer help mitigate the low-frequency gyroscope bias errors, while the signals from the accelerometer and magnetometer, which are prone to high-frequency errors, are smoothed using the gyroscope data.

The goal of this thesis work is to develop a software application that calculates and visualizes the estimated gravity vectors from sensor orientation given IMU sensory data. The task entails understanding the orientation filter and fusion algorithms, implementing said algorithms in software and developing a 3D visualization tool to observe the results. First, I investigate the background and significance of orientation estimation with IMUs and give a technical overview and comparison of different filtering algorithms. Then, I present the design and development steps taken, and decisions made, during the implementation of the software application, as well as a presentation of, and commentary on, the estimation data generated by the application. Lastly, I compare the estimation results of the different filtering algorithms I implemented on a public dataset and discuss the possible use cases of each method.

Chapter 1

Introduction

Autonomous systems, including self-driving cars, have garnered a lot of attention in recent years. Their improvement and development are accelerated by advances in not only the sensor hardware, but the data processing software that is required to make these systems reliable. The cooperation between multiple sensors (i.e., cameras, LiDARs, inertial sensors) is paramount, as information from one aids in the operation of another. If the direction of the gravity vector of the sensor setup is known with respect to the world coordinate system, the processing of the data from all sensors in the local frame becomes easier. The gravity vector can be derived from the sensor's orientation. A variety of technologies enable the accurate measurement of orientation, but inertial based sensor systems have the advantage of being self-contained, meaning the measurement entity is not constrained in motion or to any specific environment [1].

An inertial measurement unit (IMU) with nine degrees of freedom (9DoF) measures specific force (also called proper acceleration), angular rate and magnetic field strength. Alternatively, a 6DoF IMU only measures the specific force and angular rate with a gyroscope and accelerometer respectively. The measurements are three dimensional vectors in a local coordinate system that rotates with the body to which the IMU is attached [2]. A 9DoF IMU is required to provide a complete measurement of orientation relative to the direction of magnetic field and gravity, while a 6DoF IMU only provides a partial orientation as all accelerometers are insensitive to rotations about the constant gravitational field vector, pointing straight down on Earth. An orientation estimating algorithm that can fuse the sensor information of an IMU into an optimal orientation can be used as a functional component of any

autonomous system.

I will implement a naive complementary filter and the Madgwick filter compatible with both 9DoF and 6DoF IMU data to estimate the orientation of the IMU. Using the estimated orientations, I will visualize the movement of the gravity vector of the IMU over time in a custom built application with a graphical user interface (GUI). The GUI will also provide the option to visualize the motion of the true orientation (if the data exists) to provide context to the estimations. The accuracy of the estimation of a filter depends on filter gain parameters. There will be an option to change the filter parameters for each filter, allowing the user to view the motion of the new estimations, as well as the accuracy metrics for that filter and parameter combination.

Chapter 2

Theoretical Background

The main concepts and techniques necessary to understand orientation and rotations in three dimensions are presented here. This project utilizes both Euler angles and quaternions in the representation of orientation.

2.1 Frames of reference

Orientation and rotation in three dimensions requires the understanding of three coordinate frames of reference.

The Earth-centered, Earth-fixed frame (ECEF)

This is a global frame attached to the Earth itself, with a set of Cartesian axes originating at Earth's center, represented by the black axes in Figure 2.1. The x-axis points to the 0° latitude, 0° longitude point. The y-axis points to 0° latitude and 90° longitude while the z-axis points to 90° latitude along Earth's axis of rotation [3]. The axes of the ECEF are right handed and rotate with the center of the Earth.

The local geographic frame (Earth frame)

This is a frame attached to Earth but based at where the sensor is. At any point on Earth's surface, the local geographic frame is defined by the ground (horizontal direction) and the vertical direction [3]. It is the basic reference to which the sensor orients, defining straight and level orientation and the direction of down. Two Cartesian coordinate systems are used in the local geographic frame with their origins at the point in question. In the North–East–Down (NED), the local directions

of north, east, and down define a right-handed set of three axes where the x-axis corresponds to north, the y-axis to east and the z-axis to down. [3]. An alternative is the local East–North–Up (ENU). In both axes sets, the north and east directions define a plane tangent to Earth's surface. At any point, north points along the local meridian, while east points along the local small circle of constant latitude [3]. The vertical direction is perpendicular to this plane and generally does not intersect with the center of the Earth, as seen in Figure 2.1.

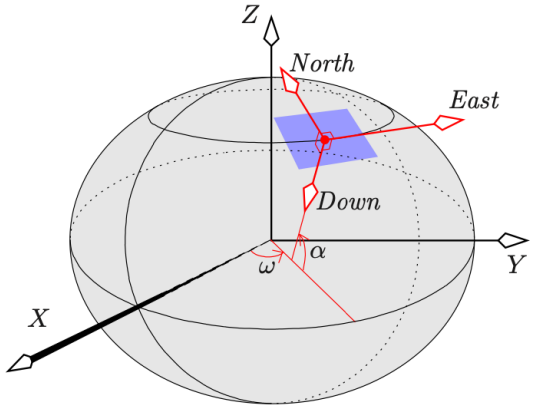


Figure 2.1: NED frame diagram from [3]

The sensor frame

This frame is fixed to the sensor and uses three Cartesian axes relative to it. Conventionally the x-axis points forward, the y-axis points out along the starboard side, and the z-axis points down.

2.2 Euler angle orientation

The orientation of a rigid body can be described by its yaw, pitch, and roll rotations from an initial position, called the Euler angle notation. The yaw, pitch, and roll rotation matrices, given by Equations 2.1, 2.2, and 2.3, transform a vector under rotation of the coordinate system by angles ψ, θ, and ϕ around the z, y, and x axes respectively.

$$\boldsymbol{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.2}$$

$$\boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.3}$$

The rotation matrices do not commute, meaning that the composite rotation matrix $\boldsymbol{R}$ depends on the order in which the yaw, pitch and roll rotations are applied. We use the conventional aerospace sequence of yaw, then pitch and finally a roll rotation. The composite rotation matrix to determine the effect on the Earth's gravitational field of 1g initially aligned downwards along the z-axis is defined by Equation 2.4.

$$\boldsymbol{R}_{xyz} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \sin(\theta) \\ -\cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix} \tag{2.4}$$

Notice that the composite rotation matrix is only a function of the roll and pitch angles. The accelerometer output has three components but, since the vector magnitude must always equal 1g in the absence of linear acceleration, it has just two degrees of freedom. Thus, it is not possible to solve for the three unique values of the roll, pitch and yaw. All accelerometers are completely insensitive to rotations about the gravitational field vector and cannot be used to determine a rotation about the z-axis. To get the pitch and roll angles, Equation 2.4 can be rewritten as

$$\frac{\boldsymbol{a}}{||\boldsymbol{a}||} = \frac{1}{\sqrt{a_x^2 + a_y^2 + a_z^2}} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \sin(\theta) \\ -\cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix} \tag{2.5}$$

where $\boldsymbol{a}$ is the accelerometer reading, with x, y and z components. Solving for the roll (ϕ) and pitch (θ) angles yields

$$\phi = \text{atan2}(-a_y, a_z) \tag{2.6}$$

$$\theta = \text{atan2}(a_x, \sqrt{a_y^2 + a_z^2}) \tag{2.7}$$

The yaw angle ψ can be determined if a magnetometer sample is available in the same reference frame. The magnetometer measurement first has to be compensated by the tilt, $\boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)$,

$$\boldsymbol{b} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ \sin(\phi)\sin(\theta) & \cos(\phi) & -\sin(\phi)\cos(\theta) \\ -\cos(\phi)\sin(\theta) & \sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \tag{2.8}$$

$$\begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} m_x\cos(\theta) + m_z\sin(\theta) \\ m_x\sin(\phi)\sin(\theta) + m_y\cos(\phi) - m_z\sin(\phi)\cos(\theta) \\ -m_x\cos(\phi)\sin(\theta) + m_y\sin(\phi) + m_z\cos(\phi)\cos(\theta) \end{bmatrix} \tag{2.9}$$

The yaw angle ψ is then obtained as

$$\psi = \text{atan2}(-b_y, b_x) \tag{2.10}$$

The rotation matrix representing a rotation from the Earth frame to the sensor frame can be directly calculated from the accelerometer and magnetometer measurements, assuming that there is no linear acceleration. When there is no linear acceleration, the accelerometer measurements are a function of orientation in the Earth's gravitational field [4]. The three columns of the identity matrix in Equation 2.11 are the three unit vectors in the x, y and z directions aligned with the initial orientation in the Earth frame.

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

The accelerometer measures the gravity vector and thus provides the last column of the rotation matrix. The magnetometer measures the geomagnetic vector in the global frame pointing northwards and downwards in the northern hemisphere and northwards and downwards in the southern hemisphere [4]. Therefore, the cross product between the gravity and geomagnetic vector is the easterly pointing vector from the global frame which is the y direction for NED or x direction for ENU. Since the matrix is orthonormal, a final cross product between the two vectors determined so far gives the last column vector [4]. The rotation matrix for NED can be determined by Equation 2.12.

$$\boldsymbol{R} = \begin{bmatrix} (\boldsymbol{a} \times \boldsymbol{m}) \times \boldsymbol{a} & \boldsymbol{a} \times \boldsymbol{m} & \boldsymbol{a} \end{bmatrix} \tag{2.12}$$

2.3 Quaternion representation

Orientation is commonly parameterized using quaternions, first introduced by Sir William Rowan Hamilton in 1844 [5]. Quaternions are generally represented in the form in Equation 2.13, where q_0 is referred to as the real part and q_1, q_2 and q_3 the imaginary parts.

$$q = q_0 + \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3 \tag{2.13}$$

Quaternions can be regarded as a four-dimensional vector space formed by combining a real number with a three-dimensional vector, with a basis given by the unit vectors 1, i, j and k such that $i^2 = j^2 = k^2 = ijk = -1$. They can be used to represent the orientation of a rigid body or coordinate frame in three-dimensional space. Although less intuitive than Euler angles, quaternions do not suffer from the "gimbal lock" phenomenon, where, in some rotation sequences, two of the three Euler angles cause rotation around the same axis. We will adopt the notation used by Madgwick [1], where a system of leading super-scripts and sub-scripts is used to denote the relative frames of orientations and vectors. A leading sub-script denotes the frame being described and a leading super-script denotes the frame this is with reference to.

The complex conjugate of a quaternion swaps the relative frames described by it, as shown in Equation 2.14.

$${}^{B}_{A}\boldsymbol{q} = {}^{A}_{B}\boldsymbol{q}^{*} = \begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \end{bmatrix}^T \tag{2.14}$$

The inverse of a quaternion is defined as Equation 2.15. In the case of a unit quaternion, the inverse is the conjugate.

$${}^{A}_{B}\boldsymbol{q}^{-1} = \frac{{}^{A}_{B}\boldsymbol{q}^{*}}{|{}^{A}_{B}\boldsymbol{q}|^2} \tag{2.15}$$

A unit quaternion, ${}^{A}_{B}\boldsymbol{q}$, describing the orientation of frame B relative to frame A is defined by Equation 2.16.

$${}^{A}_{B}\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T, \quad {}^{A}_{B}\boldsymbol{q} \in \mathbb{R}^4, \quad ||{}^{A}_{B}\boldsymbol{q}||_2 = 1 \tag{2.16}$$

The quaternion product, or Hamilton product, denoted by $\otimes$, is used to define compound orientations [5]. The Hamilton product of two quaternions, $\boldsymbol{a}$ and $\boldsymbol{b}$, is

defined by Equation 2.17. It is important to note that the quaternion product is not commutative.

$$\boldsymbol{a} \otimes \boldsymbol{b} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \otimes \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix} = \begin{bmatrix} a_1 b_1 - a_2 b_2 - a_3 b_3 - a_4 b_4 \\ a_1 b_2 + a_2 b_1 + a_3 b_4 - a_4 b_3 \\ a_1 b_3 - a_2 b_4 + a_3 b_1 + a_4 b_2 \\ a_1 b_4 + a_2 b_3 - a_3 b_2 + a_4 b_1 \end{bmatrix}^T \tag{2.17}$$

The orientation described by ${}^A_B\boldsymbol{q}$ can be represented as the rotation matrix ${}^A_B\boldsymbol{R}$ defined by Equation 2.18 [6].

$${}^A_B\boldsymbol{R} = \begin{bmatrix} 2q_1^2 + 2q_2^2 - 1 & 2(q_2 q_3 + q_1 q_4) & 2(q_2 q_4 - q_1 q_3) \\ 2(q_2 q_3 - q_1 q_4) & 2q_1^2 + 2q_3^2 - 1 & 2(q_3 q_4 + q_1 q_2) \\ 2(q_2 q_4 + q_1 q_3) & 2(q_3 q_4 - q_1 q_2) & 2q_1^2 + 2q_4^2 - 1 \end{bmatrix} \tag{2.18}$$

Euler angles yaw (ψ), pitch (θ), and roll (ϕ), defined by Equations 2.19, 2.20 and 2.21, describe an orientation of a frame B, from frame A, achieved by sequential rotations of ψ around the z-axis of frame B, θ around the y-axis of frame B and ϕ around the x-axis of frame B.

$$\psi = \text{atan2}(2q_2 q_3 - 2q_1 q_4, 2q_1^2 + 2q_2^2 - 1) \tag{2.19}$$

$$\theta = -\sin^{-1}(2q_2 q_4 + 2q_1 q_3) \tag{2.20}$$

$$\phi = \text{atan2}(2q_3 q_4 - 2q_1 q_2, 2q_1^2 + 2q_4^2 - 1) \tag{2.21}$$

2.3.1 Quaternion from Euler angles

Given the Euler angles representing a rotation, the quaternion can be determined by combining the quaternion representations of the individual Euler rotations, as described by Equation 2.22 [7]. First, the rotation around z is applied, then y and lastly x.

$$\boldsymbol{q} = \begin{bmatrix} \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) - \cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) - \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) \end{bmatrix} \tag{2.22}$$

2.3.2 Quaternion from rotation matrix

Converting a quaternion to the corresponding rotation matrix is given by Euler-Rodrigues formula, but the reverse can be performed in many different ways with different numerical behavior. In 1978, Shepperd [8] proposed a method for computing the quaternion corresponding to a rotation matrix with a voting scheme between four possible solutions, allowing his method to always work far from formulation singularities [9]. Currently, Sarabandi's method, developed in 2019, is considered the most reliable method, outperforming Shepperd's.

An arbitrary rotation matrix, in this case a direction cosine matrix, can be expressed in the general form in Equation 2.23.

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.23}$$

By equating the matrices in Equations 2.18 and 2.23, we can express the values of the quaternions with the values of the arbitrary rotation matrix, such as the example in Equation 2.24.

$$q_1 = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}} \tag{2.24}$$

Unfortunately, numerical problems arise when the term inside the square root gets close to zero. Since $\text{Trace}(\boldsymbol{R}) = r_{11} + r_{22} + r_{33} = 2\cos(\theta) + 1$, the term then coincides with $2 + 2\cos(\theta)$, where the equation becomes unstable as $\theta \to \pi$ [9]. In practice, it can become negative due to rounding errors [9]. To mitigate this, instead of only taking into account the diagonal entries, an alternative formula can be constructed involving all the elements of the rotation matrix, shown in Equation 2.25 [9].

$$\frac{1 + r_{11} + r_{22} + r_{33}}{4} + \left(\frac{r_{32} - r_{23}}{4q_1}\right)^2 + \left(\frac{r_{13} - r_{31}}{4q_1}\right)^2 + \left(\frac{r_{21} - r_{12}}{4q_1}\right)^2 = 1 \tag{2.25}$$

Solving this equation from q_1 results in Equation 2.26.

$$q_1 = \frac{1}{2}\sqrt{\frac{(r_{32} - r_{23})^2 + (r_{13} - r_{31})^2 + (r_{21} - r_{12})^2}{3 - r_{11} - r_{22} - r_{33}}} \tag{2.26}$$

Here, the denominator coincides with $2 - 2\cos(\theta)$, making it unstable as $\theta \to 0$. Equations 2.24 and 2.26 are complementary based on a threshold for the trace of $\boldsymbol{R}$ [9]. The entire quaternion can be determined from Equations 2.27, 2.28, 2.29 and 2.30.

$$q_1 = \begin{cases} \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}}, & \text{if } r_{11} + r_{22} + r_{33} > \eta \\ \frac{1}{2}\sqrt{\frac{(r_{32}-r_{23})^2+(r_{13}-r_{31})^2+(r_{21}-r_{12})^2}{3-r_{11}-r_{22}-r_{33}}}, & \text{otherwise} \end{cases} \tag{2.27}$$

$$q_2 = \begin{cases} \frac{1}{2}\sqrt{1 + r_{11} - r_{22} - r_{33}}, & \text{if } r_{11} - r_{22} - r_{33} > \eta \\ \frac{1}{2}\sqrt{\frac{(r_{32}-r_{23})^2+(r_{12}+r_{21})^2+(r_{31}+r_{13})^2}{3-r_{11}+r_{22}+r_{33}}}, & \text{otherwise} \end{cases} \tag{2.28}$$

$$q_3 = \begin{cases} \frac{1}{2}\sqrt{1 - r_{11} + r_{22} - r_{33}}, & \text{if } -r_{11} + r_{22} - r_{33} > \eta \\ \frac{1}{2}\sqrt{\frac{(r_{13}-r_{31})^2+(r_{12}+r_{21})^2+(r_{23}+r_{32})^2}{3+r_{11}-r_{22}+r_{33}}}, & \text{otherwise} \end{cases} \tag{2.29}$$

$$q_4 = \begin{cases} \frac{1}{2}\sqrt{1 - r_{11} - r_{22} + r_{33}}, & \text{if } -r_{11} - r_{22} + r_{33} > \eta \\ \frac{1}{2}\sqrt{\frac{(r_{21}-r_{12})^2+(r_{31}+r_{13})^2+(r_{32}+r_{23})^2}{3+r_{11}+r_{22}-r_{33}}}, & \text{otherwise} \end{cases} \tag{2.30}$$

The signs of $r_{32} - r_{23}$, $r_{13} - r_{31}$, and $r_{21} - r_{12}$ have to be assigned to q_2, q_3 and q_4 respectively because the square roots make the signs of each undefined (q_1 is assumed to be positive). According to the tests done by Sarabandi, a threshold value of $\eta = 0$ results in the least error and outperforms Shepperd's method.

2.3.3 Quaternion rotation

A vector $\boldsymbol{v} \in \mathbb{R}^3$ is a pure quaternion. A pure quaternion is defined as a quaternion whose scalar part is zero [6]. From the one-to-one relationship between all vectors in $\mathbb{R}^3$ and their corresponding pure quaternion, the meaning of the product of a vector and a quaternion becomes the quaternion product of two quaternions, the former being a pure quaternion [6].

Any unit quaternion can be written as Equation 2.31 where $\boldsymbol{u} = \frac{\boldsymbol{q}}{|\boldsymbol{q}|}$ and $\tan(\theta) = \frac{|\boldsymbol{q}|}{q_0}$ [6].

$$q = q_0 + \boldsymbol{q} = \cos(\theta) + \boldsymbol{u}\sin(\theta) \tag{2.31}$$

Using this notation, the following two theorems describe the rotation of a vector by a quaternion [6].

Theorem 1. *For any unit quaternion*

$$q = q_0 + \boldsymbol{q} = \cos(\theta) + \boldsymbol{u}\sin(\theta)$$

and for any vector $\boldsymbol{v} \in \mathbb{R}^3$ *the action of the operator*

$$L_q(\boldsymbol{v}) = q\boldsymbol{v}q^*$$

on $\boldsymbol{v}$ *is equivalent to a rotation of the vector through an angle* θ *about* $\boldsymbol{u}$ *as the axis of rotation.*

Theorem 2. *For any unit quaternion*

$$q = q_0 + \boldsymbol{q} = \cos(\theta) + \boldsymbol{u}\sin(\theta)$$

and for any vector $\boldsymbol{v} \in \mathbb{R}^3$ *the action of the operator*

$$L_q(\boldsymbol{v}) = q^*\boldsymbol{v}q$$

is a rotation of the coordinate frame about the axis $\boldsymbol{u}$ *through an angle* 2θ *while* $\boldsymbol{v}$ *is not rotated.*

Chapter 3

Filters

Accurate estimates of the orientation, and thus the gravity vector, of a body by inertial sensing require that the complementary properties of gyroscopes, accelerometers and magnetometers are exploited. In the sensor fusion algorithms, the accelerometer and magnetometer help mitigate the low-frequency gyroscope bias errors, while, in turn, the signals from the accelerometer and magnetometer, which are prone to high-frequency errors, are smoothed using gyroscope data [10]. Different design approaches exist for such sensor fusion algorithms.

3.1 Kalman filter approach

3.1.1 Linear models

Optimal filtering refers to the method used for estimating the state of a time varying system, from which we observe noisy measurements. State refers to the physical state of the system which can be described by dynamic variables, such as the kinematics of a moving body. The noise represents the uncertainty present in the measurements [11]. A dynamic system evolves as a function of time, with process noise present, meaning the dynamic system cannot be modelled entirely deterministically [11]. In this context, filtering basically means the process of filtering out the noise in the measurements and providing an optimal estimate for the state, given the observed measurements and the assumptions made about the dynamic system.

The instantaneous state of the system is represented with a vector updated through discrete time increments to generate the next state. The simplest of the

state space models are linear models, expressed with equations of the following form [11]:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{F}_{t-1}\mathbf{x}_{t-1} + \mathbf{w}_x \\ \mathbf{y}_t &= \mathbf{H}_t\mathbf{x}_t + \mathbf{w}_z \end{aligned} \tag{3.1}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is the state of the system at time t, $\mathbf{y}_t \in \mathbb{R}^m$ is the measurement at time t, $\mathbf{w}_x \sim \mathcal{N}(0, \mathbf{Q}_{t-1})$ is the process noise at time t-1, $\mathbf{w}_z \sim \mathcal{N}(0, \mathbf{R}_t)$ is the measurement noise at time t, $\mathbf{F}_{t-1}$ is the transition matrix (fundamental matrix) of the model, and $\mathbf{H}_t$ is the measurement model matrix.

3.1.2 Kalman filter

The classical Kalman filter was introduced by Rudolf E. Kálmán in his 1960 paper [12]. His method models a system with a set of n-th order differential equations, converts them into an equivalent set of first-order differential equations, puts them into the matrix form, and converts these linear differential equations into a closed-form recursive solution for the estimation of linear discrete-time dynamic systems. The filter has two steps: the prediction step, where the next state of the system is predicted given the previous measurements, and the update step, where the current state of the system is estimated given the measurement at that time step [11].

- Prediction:

$$\begin{aligned} \hat{\mathbf{x}}_t &= \mathbf{F}_{t-1}\mathbf{x}_{t-1} \\ \hat{\mathbf{P}}_t &= \mathbf{F}_{t-1}\mathbf{P}_{t-1}\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1} \end{aligned} \tag{3.2}$$

- Update:

$$\begin{aligned} \mathbf{v}_t &= \mathbf{y}_t - \mathbf{H}_t\hat{\mathbf{x}}_t \\ \mathbf{S}_t &= \mathbf{H}_t\hat{\mathbf{P}}_t\mathbf{H}_t^T + \mathbf{R} \\ \mathbf{K}_t &= \hat{\mathbf{P}}_t\mathbf{H}_t^T\mathbf{S}_t^{-1} \\ \mathbf{x}_t &= \hat{\mathbf{x}}_t + \mathbf{K}_t\mathbf{v}_t \\ \mathbf{P}_t &= \hat{\mathbf{P}}_t - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T \end{aligned} \tag{3.3}$$

where

- $\hat{\mathbf{P}}_t$ is the predicted covariance of the state before seeing the measurement.
- $\mathbf{P}_t$ is the estimated covariance of the state after seeing the measurement.
- $\mathbf{v}_t$ is the innovation or measurement residual.

- $\mathbf{S}_t$ is the measurement prediction covariance.
- $\mathbf{K}_t$ is the filter gain which tells how much the predictions should be corrected at time t.

The predicted state $\hat{\mathbf{x}}_t$ is estimated in the first step based on the previously computed state $\mathbf{x}_{t-1}$, and then corrected during the second step to obtain the final estimation $\mathbf{x}_t$.

3.1.3 Extended Kalman filter

Many dynamic systems, such as rigid body motion, are not linear by nature, so the traditional Kalman filter cannot be applied in estimating the state of such a system. In these kind of systems, either or both of the dynamics and the measurement processes can be nonlinear. The Extended Kalman filter (EKF), which is based on the Taylor series approximation of the joint distribution, handles nonlinear dynamical systems by forming Gaussian approximations to the joint distribution of the state $\mathbf{x}$ and measurement $\mathbf{y}$ [11].

The filtering model used in the EKF is [11]

$$\begin{aligned} \mathbf{x}_t &= \mathbf{f}(\mathbf{x}_{t-1}, t-1) + \mathbf{w}_x \\ \mathbf{y}_t &= \mathbf{h}(\mathbf{x}_t, t) + \mathbf{w}_z \end{aligned} \tag{3.4}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is the state of the system at time t, $\mathbf{y}_t \in \mathbb{R}^m$ is the measurement at time t, $\mathbf{w}_x \sim \mathcal{N}(0, \mathbf{Q}_{t-1})$ is the process noise at time $t-1$, $\mathbf{w}_z \sim \mathcal{N}(0, \mathbf{R}_t)$ is the measurement noise at time t, $\mathbf{f}$ is the dynamic model function, and $\mathbf{h}$ is the measurement model function. Both the dynamic and measurement model function can be nonlinear. Like the classic Kalman filter, the EKF is separated into two steps:

- Prediction:

$$\begin{aligned} \hat{\mathbf{x}}_t &= \mathbf{f}(\mathbf{x}_{t-1}, t-1) \\ \hat{\mathbf{P}}_t &= \mathbf{F}_{\mathbf{x}}(\mathbf{x}_{t-1}, t-1)\mathbf{P}_{t-1}\mathbf{F}_{\mathbf{x}}^T(\mathbf{x}_{t-1}, t-1) + \mathbf{Q}_{t-1} \end{aligned} \tag{3.5}$$

- Update:

$$\begin{aligned} \mathbf{v}_t &= \mathbf{y}_t - \mathbf{h}(\hat{\mathbf{x}}_t, t) \\ \mathbf{S}_t &= \mathbf{H}_{\mathbf{x}}(\hat{\mathbf{x}}_t, t)\hat{\mathbf{P}}_t\mathbf{H}_{\mathbf{x}}^T(\hat{\mathbf{x}}_t, t) + \mathbf{R}_t \\ \mathbf{K}_t &= \hat{\mathbf{P}}_t\mathbf{H}_{\mathbf{x}}^T(\hat{\mathbf{x}}_t, t)\mathbf{S}_t^{-1} \\ \mathbf{x}_t &= \hat{\mathbf{x}}_t + \mathbf{K}_t\mathbf{v}_t \\ \mathbf{P}_t &= \hat{\mathbf{P}}_t - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T \end{aligned} \tag{3.6}$$

where the matrices $\mathbf{F_x}$ and $\mathbf{H_x}$ are the Jacobians of $\mathbf{f}$ and $\mathbf{h}$.

3.2 Complementary filters

Complementary filters are an alternative to Extended Kalman filters for orientation estimation [13]. These algorithms are based on the understanding that both the gyroscope and the combination of the accelerometer and magnetometer measurements provide information of the orientation of the sensor. They leverage the fact that accelerometer and magnetometer measurements are noisy in the short term but accurate over long periods of time and gyroscope measurements are accurate on a short time scale but drift over time. In the frequency domain, the orientation estimates using the gyroscope have desirable properties at high frequencies and are filtered using a high-pass filter. Conversely, the orientation estimates obtained from the accelerometer and magnetometer measurements have desirable properties at low frequencies and are filtered using a low pass filter. The basic structure of a complementary filter is shown in Figure 3.1, which consists of two inputs, x_1 and x_2, which are low- and high-frequency noise-corrupted versions of the signal x. The output $\hat{x}$ is given in Equation 3.7, where $G(s)$ represents the transfer function for the low-pass filter and $\overline{G}(s)$ is the transfer function of the high-pass filter, such that $G(s) + \overline{G}(s) = 1$.

$$\hat{x} = x_1 G(s) + x_2 \overline{G}(s) \tag{3.7}$$

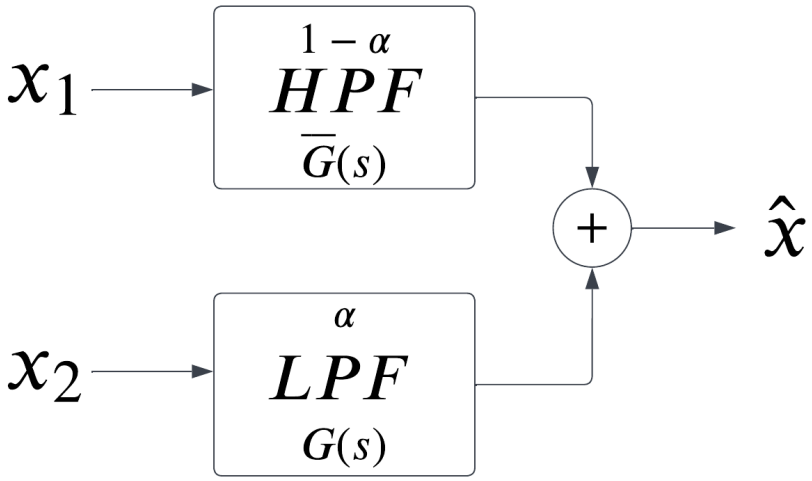


Figure 3.1: Basic complementary filter structure

3.2.1 Naive complementary filter

The most basic naive complementary filter, described by Equation 3.8, utilizes a filter gain, α, as a floating value within the range $[0.0, 1.0]$. When $\alpha = 1$, the orientation is estimated entirely with the accelerometer and the magnetometer measurements. When $\alpha = 0$, it is estimated solely with the gyroscope measurements. The gain decides how much of each estimation is blended into the final estimation.

$$\boldsymbol{q}_{est} = (1 - \alpha)\boldsymbol{q}_{\omega} + \alpha\boldsymbol{q}_{g,b} \tag{3.8}$$

The orientation from the gyroscope measurements, $\boldsymbol{q}_{\omega}$, is estimated by a process called attitude propagation where the current angular rate is numerically integrated and added to the previous orientation estimate. If only accelerometer measurements are available, the partial orientation, $\boldsymbol{q}_g$, is determined by calculating the Euler angles roll and pitch using Equations 2.6 and 2.7. The yaw angle is 0. The quaternion representation is obtained using Equation 2.22. If magnetometer measurements are also available, the complete orientation is determined as a rotation matrix with Equation 2.12. Using the process described in Section 2.3.2, the quaternion, $\boldsymbol{q}_{g,b}$, is attained.

3.2.2 Madgwick filter

The Madgwick orientation filter, proposed by Sebastian Madgwick [1], is a type of complementary filter which uses a gradient-descent algorithm to compute the gyroscope measurement error. The filter is applicable to IMUs consisting of tri-axis gyroscopes and accelerometers, with the option of including tri-axis magnetometer data as well. Madgwick's original implementation incorporates magnetic distortion and gyroscope bias drift compensation [1] and is computationally less expensive than the traditional Kalman filter [1].

Orientation from angular rate

The rate of change of orientation of the Earth frame relative to the sensor frame, ${}^{S}_{E}\dot{\boldsymbol{q}}$, can be calculated as

$$ {}^{S}_{E}\dot{\boldsymbol{q}} = \frac{1}{2} {}^{S}_{E}\hat{\boldsymbol{q}} \otimes {}^{S}\boldsymbol{\omega} \tag{3.9} $$

where ${}^{S}\boldsymbol{\omega} = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix}$ is the quaternion representation of the gyroscope measurements. The orientation of the Earth frame relative to the sensor frame at time t, ${}^{S}_{E}\boldsymbol{q}_{\omega,t}$, can be computed by numerically integrating the quaternion derivative from Equation 3.9 given that the initial conditions are known [1].

$$ {}^{S}_{E}\dot{\boldsymbol{q}}_{\omega,t} = \frac{1}{2} {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} \otimes {}^{S}\boldsymbol{\omega} \tag{3.10} $$

$$ {}^{S}_{E}\boldsymbol{q}_{\omega,t} = {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} + {}^{S}_{E}\dot{\boldsymbol{q}}_{\omega,t}\Delta t \tag{3.11} $$

In Equations 3.10 and 3.11, ${}^{S}\boldsymbol{\omega}_t$ is the angular rate measured at time t, Δt is the sampling period and ${}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}$ is the previous estimation at time $t-1$.

Orientation from accelerometer and magnetometer

An accelerometer measures the magnitude and direction of the field of gravity in the sensor frame, combined with linear accelerations due to motion. A magnetometer measures the magnitude and direction of the Earth's magnetic field in the sensor frame combined with local magnetic flux and noise due to distortions. If the direction of the Earth's field is known in the Earth frame, a measurement of the field's direction within the sensor frame will allow an orientation of the sensor frame relative to the Earth frame to be calculated [1]. However, for any given measurement there will

not be a unique sensor orientation, instead there will infinite solutions represented by all those orientations achieved by the rotation of the true orientation around an axis parallel with the field (the vertical, or z, axis in the Earth frame). If Euler angle representation is used, the incomplete solution is two known Euler angles (pitch and roll) and one unknown (yaw or heading). A quaternion representation, however, requires a complete solution to be found. The Madgwick filter achieves this through the formulation of an optimization problem where an orientation of the sensor, ${}^{S}_{E}\hat{\boldsymbol{q}} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}$, is that which aligns a predefined reference direction of the field of gravity in the Earth frame, ${}^{E}\hat{\boldsymbol{d}} = \begin{bmatrix} 0 & d_x & d_y & d_z \end{bmatrix}$, with the measured direction of the field of gravity in the sensor frame, ${}^{S}\hat{\boldsymbol{s}} = \begin{bmatrix} 0 & s_x & s_y & s_z \end{bmatrix}$ [1].

$$\min_{{}^{S}_{E}\hat{\boldsymbol{q}} \in \mathbb{R}^4} \boldsymbol{f}({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}) \tag{3.12}$$

$$\boldsymbol{f}({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}) = {}^{S}_{E}\hat{\boldsymbol{q}}^{*} \otimes {}^{E}\hat{\boldsymbol{d}} \otimes {}^{S}_{E}\hat{\boldsymbol{q}} - {}^{S}\hat{\boldsymbol{s}} \tag{3.13}$$

where the conjugate ${}^{S}_{E}\hat{\boldsymbol{q}}^{*} = {}^{E}_{S}\hat{\boldsymbol{q}} = \begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \end{bmatrix}$.

To solve the optimization problem, the gradient descent algorithm is implemented for n iterations resulting in an orientation estimation based on the initial orientation and a step size μ, described by Equation 3.14. The gradient of the solution surface, defined by the objective function and its Jacobian, can be computed according to Equation 3.15.

$${}^{S}_{E}\boldsymbol{q}_{k+1} = {}^{S}_{E}\hat{\boldsymbol{q}} - \mu \frac{\nabla \boldsymbol{f}({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}})}{||\nabla \boldsymbol{f}({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}})||}, \ k = 0, 1, 2...n \tag{3.14}$$

$$\nabla \boldsymbol{f}({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}) = \boldsymbol{J}^{T}({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}) \boldsymbol{f}({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}) \tag{3.15}$$

This is the general form of the gradient descent algorithm, applicable to a field in any direction $\boldsymbol{d}$. For orientation estimation using linear acceleration measurements, the equation can be simplified by substituting ${}^{E}\hat{\boldsymbol{g}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ for ${}^{E}\hat{\boldsymbol{d}}$ and ${}^{S}\hat{\boldsymbol{a}} = \begin{bmatrix} 0 & a_x & a_y & a_z \end{bmatrix}$ for ${}^{S}\hat{\boldsymbol{s}}$. The direction of Earth's field of gravity can be assumed to be the vertical, thus the z-axis and the normalized accelerometer measurements define the direction of the field of gravity in the sensor frame. This results in Equations 3.16 and 3.17 for the objective function and the Jacobian.

$$\boldsymbol{f}_g({}^S_E\hat{\boldsymbol{q}}, {}^S\hat{\boldsymbol{a}}) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - a_x \\ 2(q_1q_2 + q_3q_4) - a_y \\ 2(\frac{1}{2} - q_2^2 - q_3^2) - a_z \end{bmatrix} \tag{3.16}$$

$$\boldsymbol{J}_g({}^S_E\hat{\boldsymbol{q}}) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{3.17}$$

The linear acceleration and the angular rate are good parameters for an orientation estimation over a short period of time, but they do not hold for longer periods of time. This is especially true when estimating the heading orientation, as the gyroscope measurements, prone to drift, are instantaneous and local, while the accelerometer computes the roll and pitch orientations only. Therefore, it is always convenient to add a reference that provides constant information about the heading (yaw) angle, such as Earth's magnetic field, provided the sensor array includes a magnetometer.

A similar simplification can be made to the general form of the gradient descent algorithm for the orientation estimation from the magnetic field given the magnetometer measurements. The Earth's magnetic field has components in one horizontal axis and the vertical axis, represented by ${}^E\hat{\boldsymbol{b}} = \begin{bmatrix} 0 & b_x & 0 & b_z \end{bmatrix}$. The objective function and the Jacobian can be simplified by substituting ${}^E\hat{\boldsymbol{b}}$ for ${}^E\hat{\boldsymbol{d}}$ and the normalized magnetometer measurements ${}^S\hat{\boldsymbol{m}} = \begin{bmatrix} 0 & m_x & m_y & m_z \end{bmatrix}$ for ${}^S\hat{\boldsymbol{s}}$.

$$\boldsymbol{f}_b({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{b}}, {}^S\hat{\boldsymbol{m}}) = \begin{bmatrix} 2b_x(\frac{1}{2} - q_3^2 - q_4^2) + 2b_z(q_2q_4 - q_1q_3) - m_x \\ 2b_x(q_2q_3 - q_1q_4) + 2b_z(q_1q_2 + q_3q_4) - m_y \\ 2b_x(q_1q_3 + q_2q_4) + 2b_z(\frac{1}{2} - q_2^2 - q_3^2) - m_z \end{bmatrix} \tag{3.18}$$

$$\boldsymbol{J}_b({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{b}}) = \begin{bmatrix} -2b_zq_3 & 2b_zq_4 & -4b_xq_3 - 2b_zq_1 & -4b_xq_4 + 2b_zq_2 \\ -2b_xq_4 + 2b_zq_2 & 2b_xq_3 + 2b_zq_1 & 2b_xq_2 + 2b_zq_4 & -2b_xq_1 + 2b_zq_3 \\ 2b_xq_3 & 2b - xq_4 - 4b_zq_2 & 2b_xq_1 - 4b_zq_3 & 2b_xq_2 \end{bmatrix} \tag{3.19}$$

To get a unique orientation of the sensor, the measurements and reference directions of both the accelerometer and magnetometer can be combined [1].

$$\boldsymbol{f}_{g,b}({}^S_E\hat{\boldsymbol{q}}, {}^S\hat{\boldsymbol{a}}, {}^E\hat{\boldsymbol{b}}, {}^S\hat{\boldsymbol{m}}) = \begin{bmatrix} \boldsymbol{f}_g({}^S_E\hat{\boldsymbol{q}}, {}^S\hat{\boldsymbol{a}}) \\ \boldsymbol{f}_b({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{b}}, {}^S\hat{\boldsymbol{m}}) \end{bmatrix} \tag{3.20}$$

$$\boldsymbol{J}_{g,b}({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{b}}) = \begin{bmatrix} \boldsymbol{J}_{g}^{T}({}^{S}_{E}\hat{\boldsymbol{q}}) \\ \boldsymbol{J}_{b}^{T}({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{b}}) \end{bmatrix} \tag{3.21}$$

The gradient quaternion, ${}^{S}_{E}\boldsymbol{q}_{\nabla,t}$, computed at time t from the previous estimate of orientation, ${}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}$, and the function gradient $\nabla \boldsymbol{f}$ defined by sensor measurements ${}^{S}\hat{\boldsymbol{a}}_t$ and, optionally, ${}^{S}\hat{\boldsymbol{m}}_t$.

$${}^{S}_{E}\boldsymbol{q}_{\nabla,t} = {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} - \mu_t \frac{\nabla \boldsymbol{f}}{||\nabla \boldsymbol{f}||} \tag{3.22}$$

The function gradient, $\nabla \boldsymbol{f}$, with just accelerometer measurements is defined by Equation 3.23. With both accelerometer and magnetometer measurements, it is defined by Equation 3.24.

$$\nabla \boldsymbol{f}_g = \boldsymbol{J}_{g}^{T}({}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1})\boldsymbol{f}_g({}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}, {}^{S}\hat{\boldsymbol{a}}_t) \tag{3.23}$$

$$\nabla \boldsymbol{f}_{g,b} = \boldsymbol{J}_{g,b}^{T}({}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}, {}^{E}\hat{\boldsymbol{b}})\boldsymbol{f}_{g,b}({}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}, {}^{S}\hat{\boldsymbol{a}}_t, {}^{E}\hat{\boldsymbol{b}}, {}^{S}\hat{\boldsymbol{m}}_t) \tag{3.24}$$

An optimal μ_t ensures that the convergence rate of the gradient quaternion, ${}^{S}_{E}\boldsymbol{q}_{\nabla,t}$, is limited to the physical orientation rate [1]. It is defined by Equation 3.25, where Δt is the sampling period of the sensor, α is an augmentation of μ to account for noise in the accelerometer and magnetometer measurements, and ${}^{S}_{E}\dot{\boldsymbol{q}}_{\omega,t}$ (defined by Equation 3.10) is the rate of change of orientation of the Earth frame relative to the sensor frame measured by the gyroscope [1].

$$\mu_t = \alpha||{}^{S}_{E}\dot{\boldsymbol{q}}_{\omega,t}||\Delta t, \ \alpha > 1 \tag{3.25}$$

Filter fusion

The estimated orientation of the sensor frame relative to the Earth frame, ${}^{S}_{E}\boldsymbol{q}_{est,t}$, is calculated through the weighted fusion of the orientation calculations, ${}^{S}_{E}\boldsymbol{q}_{\nabla,t}$ and ${}^{S}_{E}\boldsymbol{q}_{\omega,t}$, defined by Equations 3.11 and 3.22 respectively, with a simple complementary filter.

$${}^{S}_{E}\boldsymbol{q}_{est,t} = \gamma_t {}^{S}_{E}\boldsymbol{q}_{\nabla,t} + (1-\gamma_t){}^{S}_{E}\boldsymbol{q}_{\omega,t}, \ 0 \leq \gamma_t \leq 1 \tag{3.26}$$

The fusion step is simplified by Madgwick as Equation 3.27 where ${}^{S}_{E}\dot{\boldsymbol{q}}_{est,t}$ is the estimated rate of change of orientation and ${}^{S}_{E}\dot{\hat{\boldsymbol{q}}}_{\epsilon,t}$ is the direction of error of ${}^{S}_{E}\dot{\boldsymbol{q}}_{est,t}$ [1].

$$ {}^{S}_{E}\boldsymbol{q}_{est,t} = {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} + {}^{S}_{E}\dot{\boldsymbol{q}}_{est,t}\Delta t \tag{3.27} $$

$$ {}^{S}_{E}\dot{\boldsymbol{q}}_{est,t} = {}^{S}_{E}\dot{\boldsymbol{q}}_{\omega,t} - \beta {}^{S}_{E}\dot{\hat{\boldsymbol{q}}}_{\epsilon,t} \tag{3.28} $$

$$ {}^{S}_{E}\dot{\hat{\boldsymbol{q}}}_{\epsilon,t} = \frac{\nabla \boldsymbol{f}}{||\nabla \boldsymbol{f}||} \tag{3.29} $$

The filter gain β represents all mean zero gyroscope measurement errors, expressed as the magnitude of a quaternion derivative [1].

Magnetic distortion compensation

Measurements made by the magnetometer are distorted by the presence of ferromagnetic materials in the environment of the sensor, causing substantial errors [1]. These materials include electrical appliances, metal furniture and metal beams in building structures. Errors in the horizontal plane relative to the Earth's surface cannot be corrected without an additional reference of heading but errors in the vertical plane can be compensated for because the accelerometer provides an additional measurement of the sensor's inclination. The measured direction of the Earth's magnetic field in the Earth frame at time t can be computed by rotating the normalized magnetometer measurement by the estimated orientation provided by the filter, as shown in Equation 3.30. If the inclination of the filter's reference direction of Earth's magnetic field coincides with the inclination of the measured direction of Earth's magnetic field in the Earth frame, the distortion can be corrected. In Equation 3.31, the filter's reference direction of Earth's magnetic field, ${}^{E}\hat{\boldsymbol{b}}_t$, is computed as ${}^{E}\hat{\boldsymbol{h}}_t$ normalized to only have components in the x and z axes [1]. This compensation guarantees that magnetic disturbances only affect the heading.

$$ {}^{E}\hat{\boldsymbol{h}}_t = \begin{bmatrix} 0 & h_x & h_y & h_z \end{bmatrix} = {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} \otimes {}^{S}\hat{\boldsymbol{m}}_t \otimes {}^{S}_{E}\hat{\boldsymbol{q}}^{*}_{est,t-1} \tag{3.30} $$

$$ {}^{E}\hat{\boldsymbol{b}}_t = \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & 0 & h_z \end{bmatrix} \tag{3.31} $$

Gyroscope bias drift compensation

The gyroscope bias tends to drift over time, as a function of motion and temperature. The Kalman-based approaches are able to estimate the gyroscope bias as

an additional state within the system model [1]. In Madgwick's solution, a simpler orientation filter through the integral feedback of the error in the rate of change of orientation is implemented.

$$^{S}\boldsymbol{\omega}_{\epsilon,t} = 2\,{}^{S}_{E}\hat{\boldsymbol{q}}^{*}_{est,t-1} \otimes {}^{S}_{E}\dot{\hat{\boldsymbol{q}}}_{\epsilon,t} \tag{3.32}$$

$$^{S}\boldsymbol{\omega}_{b,t} = \zeta \sum_{t} {}^{S}\boldsymbol{\omega}_{\epsilon,t}\Delta t \tag{3.33}$$

$$^{S}\boldsymbol{\omega}_{c,t} = {}^{S}\boldsymbol{\omega}_{t} - {}^{S}\boldsymbol{\omega}_{b,t} \tag{3.34}$$

Chapter 4

Implementation

To create a user-friendly but comprehensive software solution for this task, a couple of requirements had to be met. Four different filter fusion algorithms had to be executed in a timely manner on different datasets. Accelerometer and gyroscope data were required, but magnetometer data was optional as the filters could handle estimation without it. The user had to be able to specify the input data as well as the input parameters for the fusion algorithms. After the orientation quaternions were estimated, the resulting gravity vectors had to be visualized for each time step and the quality of the results had to be quantified by determining the root mean squared error (RMSE) if a ground truth measurement and motion-indicating data was available.

4.1 Tools and technologies

With the requirements in mind, I chose to develop the application in C++ with help from external libraries for the graphical user interface (GUI) and 3D visualization. C++ supports dynamic memory allocation (DMA), which helps free up and allocate memory, allowing the developer to write highly efficient and performant code. This feature is beneficial for working with 3D rendering and visualization, but also for sensor fusion algorithms, which ideally run real-time on embedded computers. The filter algorithms, as well as the accuracy metrics, were written using the C++17 standard library and the graphical aspect of the application was written in C++ and C using external libraries and technologies.

For the graphics pipeline, I used OpenGL [14] with GLFW [15] and `gl3w` [16].

Open Graphics Library (OpenGL), is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics managed by the non-profit technology consortium Khronos Group. It eases the task of writing real-time 2D or 3D graphics applications by providing a mature, well-documented graphics processing pipeline that supports the abstraction of current and future hardware accelerators. GLFW is an open source, multi-platform library for OpenGL, OpenGL ES and Vulkan application development [15]. It provides a simple, platform-independent API for creating and managing windows, contexts and surfaces, reading input, and handling events. I chose GLFW over other libraries because I was familiar with it and it fit my needs in this case. Since OpenGL support is implementation specific and constantly updating, loaders are used to query which OpenGL specifications and extensions you want to use, and provide a way to access that OpenGL functionality. I used a loader called `gl3w` [16] to get the functionality offered by the OpenGL core profile specification. It consists of a Python script that downloads the Khronos supported `glcorearb.h` header and generates `gl3w.h` and `gl3w.c` from it, which can then be added and linked into the project. I chose this loader over other options because I only need the core functionalities and this loader is as simple as it gets.

I designed and constructed the GUI with the help of Dear ImGui [17], a lightweight graphical user interface library for C++. It is fast, portable, works with any renderer, and is easy to customize. The core of ImGui is self-contained within a few platform-agnostic source and header files which are built with the project [17]. ImGui provides backends for a variety of graphics APIs and platforms, including GLFW with OpenGL, allowing it to seamlessly integrate with my project.

Using this pipeline, I was able to visualize the orientation vectors in a 3D environment using the Visualization Toolkit (VTK) [18]. VTK is an open source software for image processing, 3D graphics, volume rendering and visualization of scientific data.

I built the project using CMake [19], an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in any compiler environment [19]. It provides the ability to automatically search for programs, libraries, and header files that may be required by the software being built

based on simple commands in the configuration file. It also allows the selection of optional components at configuration time, a feature I used with VTK component modules. This way I only built components I needed.

4.2 Metrics for orientation estimation accuracy

The orientation estimated by a filtering algorithm can be compared with the corresponding ground truth measurement to get an understanding of its accuracy. The disparity between two unit quaternions representing orientations is described by the shortest angular distance, e_t, between both orientations [2].

$$ {}^{S}\boldsymbol{q}_{e,t} = {}^{S}_{E}\boldsymbol{q}^{*}_{GT} \otimes {}^{S}_{E}\boldsymbol{q}_{t} =: \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^T \tag{4.1} $$

$$ e_t = 2\cos^{-1}|q_w| \tag{4.2} $$

This angular performance parameter describes the overall accuracy of the estimated orientation, and RMSE values can be used to quantify the performance of a motion interval of interest. It is important to note that the total error e_t yields only limited understanding into the potential cause of estimation errors. It is therefore important to distinguish between the error resulting from inaccurate heading estimation and error resulting from inaccurate inclination estimation. The accuracy of the heading estimation depends primarily on the sensor fusion between gyroscopes and magnetometers and the accuracy of the inclination error primarily depends on sensor fusion between gyroscopes and accelerometers [2]. When magnetic disturbance is present, the heading component of the error could be much larger than the inclination component of the orientation error [2].

I used the technique highlighted in the BROAD paper [2] to determine the heading and inclination components of the total error. They define the heading as a rotation around the vertical axis and the inclination as the rotation around a horizontal axis. Note, that the inclination decomposition in this case is not the same as the decomposition based on Euler angles, where inclination is the concatenation of two rotations around a generally not horizontal axis. First, the total orientation error is expressed in the Earth frame, as Equation 4.3.

$$ {}^{E}\boldsymbol{q}_{e,t} = {}^{S}_{E}\boldsymbol{q}_{t} \otimes {}^{S}_{E}\boldsymbol{q}^{*}_{GT} =: \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^T \tag{4.3} $$

Then, this error is decomposed into a rotation around the vertical z-axis and the shortest possible residual rotation [2], defined by Equations 4.4 and 4.5.

$$e_{h,t} = 2\text{atan2}(|q_z|, |q_w|) \tag{4.4}$$

$$e_{i,t} = 2\cos^{-1}\sqrt{q_w^2 + q_z^2} \tag{4.5}$$

Large inclination errors, $e_{i,t}$, indicate non-ideal fusion of accelerometer with gyroscope measurements while large heading errors, $e_{h,t}$, are mostly caused by magnetic disturbances in the sensor environment. The sum of both error portions is always larger or equal to the overall orientation error while each part by itself is smaller than that overall error [2].

4.3 Orientation estimation algorithm

The core of the application is the process of estimating the orientations from the IMU data and recording the orientation quaternions and error quaternions in comma-separated values (CSV) files. First, the accelerometer, gyroscope, and magnetometer data are read from CSV files. Each of these measurements are, at each time step, a three dimensional vector. The initial state quaternion is determined using Equation 2.12 and the method outlined in Section 2.3.2. For the filters that do not consider magnetometer data, the initial state is estimated using Equations 2.6, 2.7 and 2.22. Then, from the first time step to the last, the orientation estimation quaternion is updated with the IMU measurements at that time step and recorded. An sample list of orientation estimations can be found in Appendix B. The four implemented orientation estimation algorithms, detailed in Appendix A, are: the Madgwick filters with and without magnetometer data (Algorithms 1 and 2) and the naive complementary filters with and without magnetometer data (Algorithms 3 and 4). At each time step, if there is ground truth orientation data, the error quaternions are calculated and recorded for each filter's estimated orientation. Sample error quaternions can be found in Appendix B. Using the error quaternions, the total, inclination and heading RMSE are determined and recorded. The gravity vectors, for the visualization, are obtained for each time step by first determining the Euler angle representation of the orientation quaternion (Equations 2.19, 2.20 and 2.21) and calculating the z component of the rotation matrix with Equation 2.5.

4.4 Software operation

When the application is opened, the user is greeted with the GUI seen in Figure 4.1.

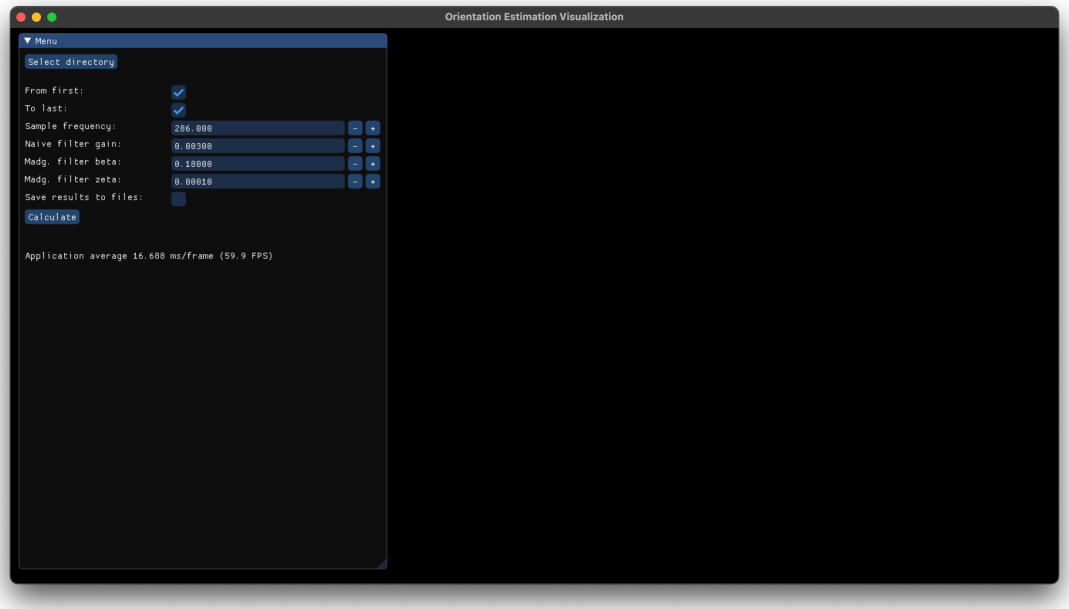


Figure 4.1: GUI initial state

The "Select directory" button opens a file explorer window through which the user can select the directory which contains the data files. Then, the user is given the option to specify which range of measurements they want to consider for the calculations by deselecting the "From first" and "To last" options. Otherwise, the whole range of measurements are considered. The following input boxes allow the user to modify the input parameters of the algorithms, i.e., sample frequency and filter gains. If the "Save results to files" option is selected, the estimation quaternions and error quaternions (if applicable) for each filter for each time step are saved to CSV files, in addition to the RMSE values for this configuration. The calculation is initiated by the "Calculate" button, which is accompanied by a simple loading graphic while the calculation is executing. Upon completion, the loading graphic disappears and the user is presented the GUI shown in Figure 4.2.

A couple changes can be noticed. The visualization options appear, the selected directory path is displayed under the "Select directory" button, the RMSE values are displayed and the 3D visualization windows appear on the right. All windows are re-sizeable and move-able.

The first set of visualization options, beginning with the “Black background” option, allow the user to configure the visualizations on the right. The “Black background” option changes the background of every 3D visualization window to black while the “Background alpha” slider changes the transparency of the background. The “Vector index” section can be used to navigate to a specific index (orientation estimation at a specific time) with a slider or navigation buttons. The “Play” button begins to animate the visualization windows from the current index to the last index. If the “Loop” option is selected, the animation continues from the first index when the end is reached, looping the animation. The animation can be stopped anytime by clicking the “Stop” button, which replaces the “Play” button when it is clicked. The RMSE values for this configuration are displayed in a table at the bottom of the menu. The table includes the inclination, heading, and total RMSE for each filter.

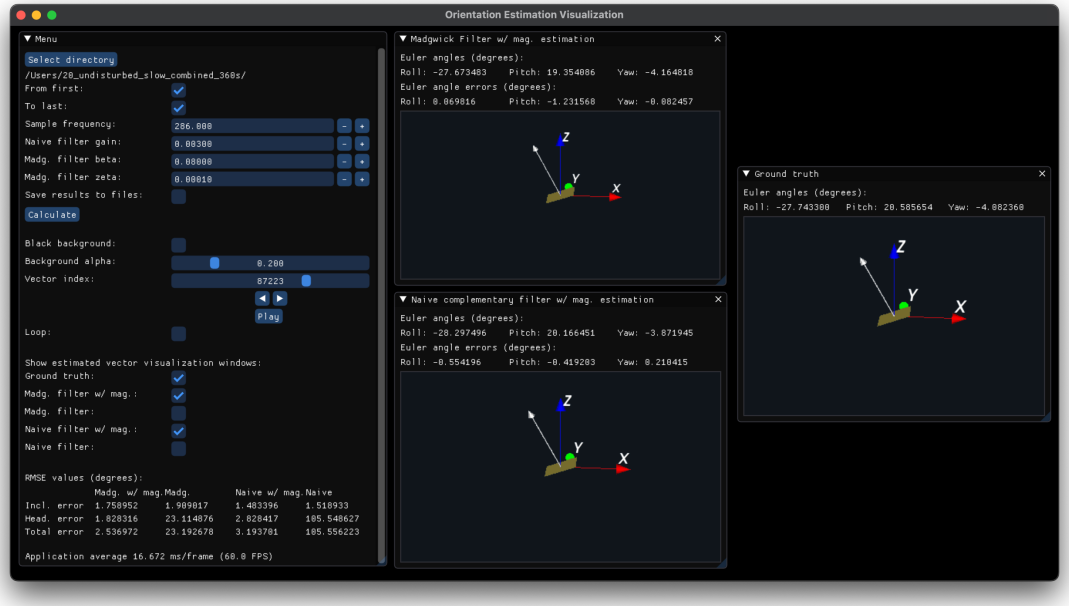


Figure 4.2: GUI state after calculation

The visualization windows can be toggled in the next section. In Figure 4.2, three filter visualizations are displayed. At the top of each visualization window, the Euler angles of the current orientation and the Euler angle errors (when ground truth data is available) are displayed. Each 3D window consists of coordinate axes, the gravity vector at that index in the ENU frame and a rectangular plane representing the IMU. The 3D windows are interactive, meaning the user can change the position of the virtual camera by clicking and dragging.

The data selection and filter parameter options remain accessible throughout the execution, allowing the user to initiate a new calculation anytime. In that case, the visualization configuration options reset and the visualization windows and RMSE values are re-rendered with the new data.

Chapter 5

Experimentation

Objective assessment of the accuracy of an orientation estimation algorithm requires a reliable and accurate ground truth measurement of the moving object's orientation. I decided to use the Berlin Robust Orientation Estimation Assessment Dataset (BROAD) [2]. This dataset consists of a diverse collection of trials, covering different movement types, speeds, as well as motions performed in the presence of magnetic disturbances. Compared to the other available datasets, BROAD covers more use case scenarios with the corresponding ground truth orientation and movement phase markers. The ground truth orientation is determined from the position measurements of active or reflective optical markers that are tracked by a set of cameras, a technique that is known as stereophotogrammetry or optical motion capture (OMC) [2].

I used all 39 trial sets offered by BROAD for the evaluation of the filters. The trials are summarized in Table 5.1. The 39 trials have a total duration of 8478 seconds when considering rest and motion phases and 5274 seconds of only motion phases [2]. The duration of a single movement phase ranges from 15 to 358 seconds. Each trial consist of five CSV files:

imu_gyr.csv contains the gyroscope measurements in rad/s

imu_acc.csv contains the linear acceleration measurements in m/s^2

imu_mag.csv contains magnetometer measurements in μT

opt_quat.csv OMC ground truth orientation as a unit quaternions in ENU frame

movement.csv booleans indicating movement phases

undisturbed	slow	fast
rotation	1, 2, 3, 4, 5	6, 7, 8, 9
translation	10, 11, 12, 13, 14	15, 16, 17, 18
combined	19, 20	21, 22, 23
disturbed (medium speed)		
tapping	24, 25	
vibrating smartphone	26, 27	
stationary magnet	25, 29, 30, 31	
attached magnet (1-5 cm)	32, 33, 34, 35, 36	
office environment	37, 38	
mixed (disturbed and undisturbed)	39	

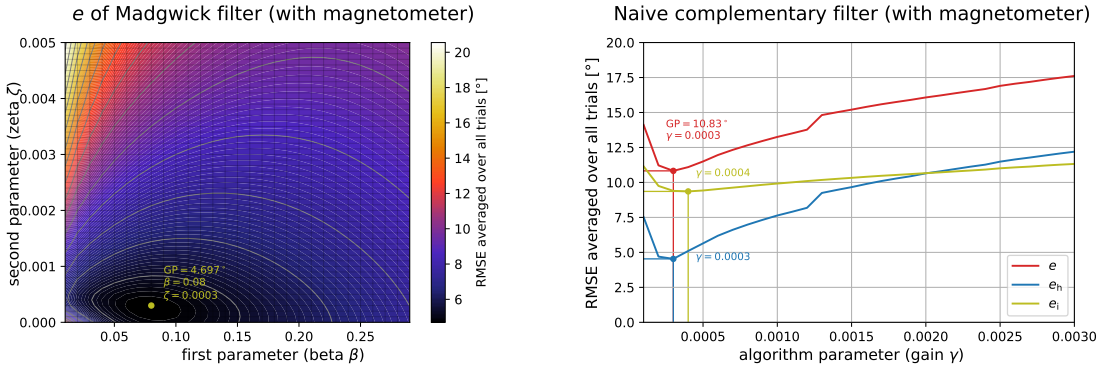
Table 5.1: Overview of the 39 trials from BROAD [2]

In order to allow for a simple and well-defined performance comparison between the different algorithms, I define the general performance (GP) metric as the smallest achievable average RMSE over all 39 trials that can be obtained with a common parameter setting for all trials. The average RMSE is determined by first running each algorithm on all 39 trial datasets with a given parameter setting. Then, for each trial, calculate the orientation RMSE (i.e., the RMS of e_t) with only motion phases (from data in movement.csv). Finally, the 39 RMSE values are averaged to get the GP of that algorithm with given parameter setting.

5.1 Results

The goal is to find the filter and parameter setting that minimizes the average orientation error over all possible scenarios. In order to determine this value, I calculated the average RMSE for many different parameter values for each filter algorithm. For the Madgwick filter with magnetometer measurements (Algorithm 1), there are two input parameters, beta β and zeta ζ. Thus, I searched a linearly spaced grid of parameter values (β: 0.01 to 0.29 in steps of 0.01, ζ: 0.0 to 0.005 in steps of 0.0001). For the naive complementary filter with magnetometer measurements (Algorithm 3), I used linearly spaced values of the single gain tuning parameter γ (0.0001 to 0.01 in steps of 0.0001). Figure 5.1 shows that for the Madgwick filter,

a $\beta = 0.08$ and a $\zeta = 0.0003$ provide the optimal results and a $\gamma = 0.0003$ provides the optimal results for the naive complementary filter. For the Madgwick filter without magnetometer measurements (Algorithm 2), I used linearly spaced values of the single gain tuning parameter β (0.01 to 0.29 in steps of 0.01). For the naive complementary filter without magnetometer measurements (Algorithm 4), I used linearly spaced values of the single gain tuning parameter γ (0.0001 to 0.01 in steps of 0.0001). Here, a $\beta = 0.03$ provides the best result for the Madwick filter and a $\gamma = 0.0009$ for the naive complementary filter.

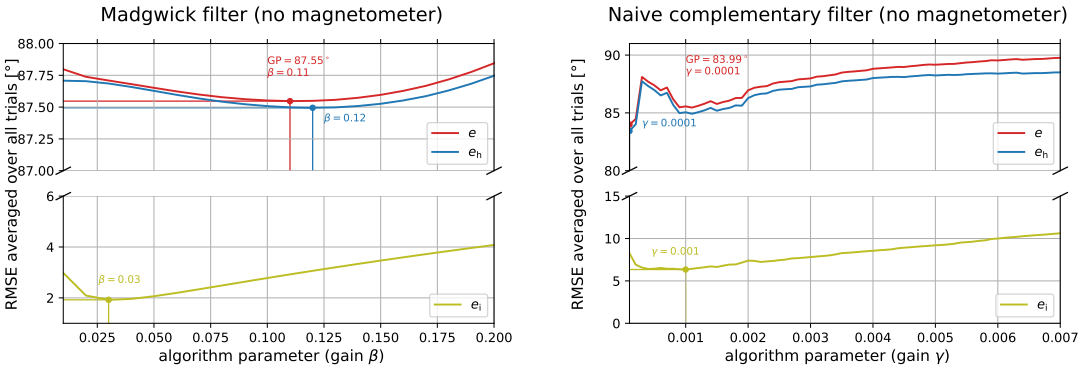


(a) Algorithm 1 for different gain values

(b) Algorithm 3 for different gain values

Figure 5.1: Orientation estimation RMSE (averaged over all trials) with magnetometer data

The Madgwick filter with magnetometer data reaches a GP score of 4.697° total error and the naive complementary filter reaches a score of 10.83°, i.e., the Madgwick filter yields a much better overall performance when there is magnetometer data. Both, however, outperformed their no magnetometer counterparts in total error. As expected, without magnetometer measurements, both filters produced much larger total error. Figure 5.2 shows that the Madgwick filter without magnetometer measurements yielded a GP of 87.55° and the naive complementary a GP of 83.99°.



(a) Algorithm 2 for different beta values (b) Algorithm 4 for different gain values

Figure 5.2: Orientation estimation RMSE (averaged over all trials) without magnetometer data

The decomposition into heading and inclination error allows for insight into potential sources of errors. For the Madgwick filter with magnetometer data, the heading error accounts for more of the total error than the inclination error, but only by a relatively small margin. Figure 5.3 shows that at the optimal gain parameters, the heading error is almost twice the inclination error, amounting to a difference of about 2°. When looking at just the inclination error, the two Madgwick filters perform nearly identically, while the heading error is much larger when no magnetometer measurements are used. The naive filters, however, diverge. The naive complementary filter without magnetometer measurements outperforms its magnetometer counterpart when looking at just the inclination error, with a difference of about 3° at optimal parameters. This phenomenon shows that the naive complementary filter sacrifices inclination accuracy for total orientation accuracy.

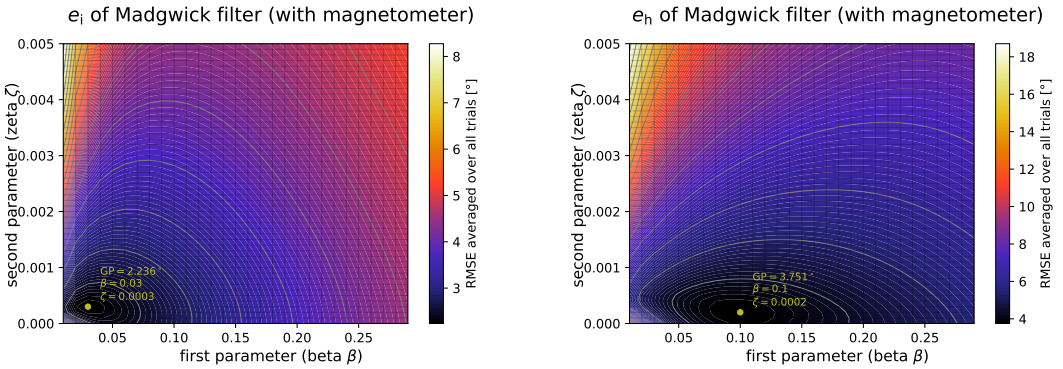


Figure 5.3: Inclination and heading estimation RMSE (averaged over all trials) for different beta and zeta values

These results support the understanding that without magnetometer measure-

ments, the heading estimation is unreliable. Accelerometers only compute the roll and pitch angles and another reference is needed, such as measurements of Earth's magnetic field, to reliably estimate the yaw. Therefore, if complete orientation estimation is the objective, magnetometer measurements are paramount and between these two filters, the Madgwick filter is preferred. Estimating the gravity vector, however, only requires the inclination, that is, roll and pitch. In that case, magnetometer measurements do not play a significant role and again, the Madgwick filter is preferred over the naive complementary filter.

To measure the execution speed of the filter algorithms, I ran each filter separately on five distinct trial datasets and measured the time it took for the algorithm to estimate an orientation for each time step. This was done by measuring the total execution time of the filter on the whole series and dividing by the number of samples in the trial dataset. Refer to Table 5.1 for a description of the trials. Table 5.2 shows that the Madgwick filter executes faster than the naive complementary filter with and without magnetometer measurements, and the magnetometer counterparts of each filter algorithm execute slower than the no magnetometer versions.

Trial	Execution times per update (nanoseconds)			
	Madwick	Madgwick w/ mag.	Naive	Naive w/ mag.
20	80.0212	147.238	126.185	156.772
23	80.8102	147.739	127.2	160.044
28	82.2305	144.767	126.811	160.266
39	82.0896	148.886	125.21	161.304
37	79.9898	144.635	126.006	156.718
Mean	81.02826	146.653	126.2824	159.0208

Table 5.2: Execution times per update of the filtering algorithms on different datasets

When considering orientation estimation accuracy and time of execution, the Madgwick filter is superior than the naive complementary filter irrespective of the presence of magnetometer data.

Chapter 6

Future Works

This project was focused on only two of the many filters available for orientation estimation. An immediate continuation of this work would be to implement an Extended Kalman filter [12] and the Mahony [20] filter and extend the application to support the configuration of these filters and the visualization of their estimations. It would be beneficial to compare the performance of these filters, in accuracy and speed, to the Madgwick filter.

Regarding the GUI application, the first avenue of improvement would be further multi-threading the functions to provide a better user experience and allow for faster execution times. This would include adding a window in which the user can run the full average RMSE metric evaluation on multiple datasets and visualize the results. This would allow the user to find the optimal algorithm and parameters for a set of data from a specific application, i.e, a test car or robot. Other minor customization options can also be added, e.g, more background colors, font size, etc.

The motivation for this project was rooted in the fact that orientation is valuable in the processing of visual data from cameras and LiDARs. To test the real-world viability of these algorithms, they have to be tested and used with data coming from the field. An exciting avenue for further development would be using the ELTECar to record IMU measurements in time synchronization with LiDAR point clouds and camera images. With data from a whole suite of sensors, the true value of these orientation estimation algorithms can be assessed as their outputs are utilized in the image and point cloud processing process.

Chapter 7

Conclusion

During this project, I researched and reviewed the mathematical definitions behind 3D orientation and the numerous IMU filtering algorithms, as well as the software technologies I used to develop my solution. I reviewed the work of researchers and mathematicians to develop an intuitive understanding of 3D rotation represented with Euler angles and quaternions, which aided my understanding of the various filtering algorithms, their benefits, and their drawbacks. Based on my research, I decided to implement two complementary filters and test their accuracy and execution time: the naive complementary filter and the Madgwick filter.

With an understanding of the theoretical concepts, I was able to develop an application that: (i) read accelerometer, gyroscope and, optionally, magnetometer data, (ii) estimated the orientation quaternion at each time step with each filter, (iii) calculated, for each orientation quaternion, the Euler angle representation, (iv) determined the gravity vector from the Euler angles, (v) visualized the gravity vectors in an interactive 3D coordinate system.

I established performance metrics for the filter algorithm orientation estimations and ran a benchmark on 39 sets of IMU data from the public BROAD dataset. The datasets covered a wide range of scenarios, from slow and fast translations and rotations to magnetically disturbed environments. My results concluded that the Madgwick filter is preferred for orientation estimation, and thus gravity vector estimation, as it outperforms the naive complementary filter with and without magnetometer data with optimal filter parameters. It also has a faster execution time, per filter update, as it is currently implemented.

Appendix A

Filter pseudocodes

Algorithm 1 Madgwick filter update (with magnetometer)

global variables

β, gain
$\boldsymbol{q}$, orientation estimate
$\boldsymbol{b}$, filter's reference direction of Earth's magnetic field

procedure update($\boldsymbol{a}$, $\boldsymbol{\omega}$, $\boldsymbol{m}$)

1: **if** $\boldsymbol{m}$ is *not* a valid input **then**
2: update($\boldsymbol{a}$, $\boldsymbol{\omega}$) ▷ Use Algorithm 2
3: **end if**
4: **if** $\boldsymbol{a}$ is a valid input **then**
5: normalize $\boldsymbol{a}$
6: normalize $\boldsymbol{m}$
7: $\boldsymbol{f}_{g,b} = \begin{bmatrix} \boldsymbol{f}_g(\boldsymbol{q}, \boldsymbol{a}) \\ \boldsymbol{f}_b(\boldsymbol{q}, \boldsymbol{b}, \boldsymbol{m}) \end{bmatrix}$ ▷ Equations 3.16 and 3.18
8: $\boldsymbol{J}_{g,b} = \begin{bmatrix} \boldsymbol{J}_g^T(\boldsymbol{q}) \\ \boldsymbol{J}_b^T(\boldsymbol{q}, \boldsymbol{b}) \end{bmatrix}$ ▷ Equations 3.17 and 3.19
9: $\nabla \boldsymbol{f}_{g,b} = \boldsymbol{J}_{g,b}^T \boldsymbol{f}_{g,b}$
10: normalize $\nabla \boldsymbol{f}_{g,b}$
11: **end if**
12: $\dot{\boldsymbol{q}}_\omega = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}$ ▷ Equation 3.10
13: $\boldsymbol{q} \leftarrow \boldsymbol{q} + (\dot{\boldsymbol{q}}_\omega - (\beta * \nabla \boldsymbol{f}_{g,b})) * \Delta t$ ▷ Equation 3.27
14: normalize $\boldsymbol{q}$
15: $\boldsymbol{h} = \boldsymbol{q} \otimes \boldsymbol{m} \otimes \boldsymbol{q}^*$ ▷ Equation 3.30
16: $\boldsymbol{b} \leftarrow \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & 0 & h_z \end{bmatrix}$ ▷ Equation 3.31

Algorithm 2 Madgwick filter update

global variables

β, gain
$\boldsymbol{q}$, orientation estimate

procedure update($\boldsymbol{a}$, $\boldsymbol{\omega}$)

if $\boldsymbol{a}$ is a valid input **then**
    normalize $\boldsymbol{a}$
    $\boldsymbol{f}_g = \boldsymbol{f}_g(\boldsymbol{q}, \boldsymbol{a})$ ▷ Equations 3.16
    $\boldsymbol{J}_g = \boldsymbol{J}_g^T(\boldsymbol{q})$ ▷ Equations 3.17
    $\nabla \boldsymbol{f}_{g,b} = \boldsymbol{J}_{g,b}^T \boldsymbol{f}_g$
    normalize $\nabla \boldsymbol{f}_g$
end if
$\dot{\boldsymbol{q}}_\omega = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}$ ▷ Equation 3.10
$\boldsymbol{q} \leftarrow \boldsymbol{q} + (\dot{\boldsymbol{q}}_\omega - (\beta * \nabla \boldsymbol{f}_g)) * \Delta t$ ▷ Equation 3.27
normalize $\boldsymbol{q}$

Algorithm 3 Naive complementary filter update (with magnetometer)

global variables

γ, gain
$\boldsymbol{q}$, orientation estimate

procedure update($\boldsymbol{a}$, $\boldsymbol{\omega}$, $\boldsymbol{m}$)

if $\boldsymbol{m}$ is *not* a valid input **then**
    update($\boldsymbol{a}$, $\boldsymbol{\omega}$) ▷ Use Algorithm 4
end if
$\mathbf{R}$ = rotation matrix from $\boldsymbol{a}$ and $\boldsymbol{m}$ ▷ Equation 2.12
$\boldsymbol{q}_{am}$ = quaternion from $\mathbf{R}$ ▷ Sarabandi's Method (Section 2.3.2)
normalize $\boldsymbol{q}_{am}$
$\dot{\boldsymbol{q}}_\omega = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}$ ▷ Equation 3.10
$\boldsymbol{q}_\omega = \boldsymbol{q} + \dot{\boldsymbol{q}}_\omega \Delta t$ ▷ Equation 3.11
normalize $\dot{\boldsymbol{q}}_\omega$
$\boldsymbol{q} \leftarrow (1 - \gamma)\boldsymbol{q}_\omega - \gamma \boldsymbol{q}_{am}$
normalize $\boldsymbol{q}$

Algorithm 4 Naive complementary filter update

global variables

γ, gain
$\boldsymbol{q}$, orientation estimate

procedure update($\boldsymbol{a}$, $\boldsymbol{\omega}$)

1: normalize $\boldsymbol{a}$
2: $\theta = \text{atan2}(a_y, a_z)$ ▷ Roll (Equation 2.6)
3: $\phi = \text{atan2}(-a_x, \sqrt{a_y^2 + a_z^2}$ ▷ Pitch (Equation 2.7)
4: $\psi = 0$
5: $\boldsymbol{q}_{am}$ = quaternion from Euler angles ▷ Equation 2.22
6: normalize $\boldsymbol{q}_{am}$
7: $\dot{\boldsymbol{q}}_\omega = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}$ ▷ Equation 3.10
8: $\boldsymbol{q}_\omega = \boldsymbol{q} + \dot{\boldsymbol{q}}_\omega \Delta t$ ▷ Equation 3.11
9: normalize $\dot{\boldsymbol{q}}_\omega$
10: $\boldsymbol{q} \leftarrow (1 - \gamma)\boldsymbol{q}_\omega - \gamma \boldsymbol{q}_{am}$
11: normalize $\boldsymbol{q}$

Appendix B

Sample outputs

Sample quaternion estimations using the Madgwick filter with magnetometer data on trial 20 from the BROAD dataset:

```
0.999516,0.00759894,-0.00291647,-0.0300307
0.999517,0.00733196,-0.00290373,-0.0300573
0.999512,0.00726223,-0.00299261,-0.0302426
0.999511,0.00708677,-0.00279546,-0.0303268
0.999509,0.00733184,-0.00269601,-0.0303442
0.999509,0.00744375,-0.0024627,-0.0303272
0.999506,0.00751326,-0.00271414,-0.0304042
0.999501,0.00765585,-0.00291673,-0.03051
0.999499,0.00745263,-0.00294511,-0.0306033
0.999497,0.00728532,-0.00306492,-0.0307021
0.999494,0.00709606,-0.00315186,-0.0308367
0.99949,0.0071618,-0.00338012,-0.0309395
```

Sample error quaternions from estimations using the Madgwick filter with magnetometer data on trial 20 from the BROAD dataset:

```
0.0709172,0.0369812,0.0605158
0.0708769,0.0367029,0.0606372
0.0711287,0.0366249,0.0609766
0.0713129,0.0366053,0.0612038
0.0700545,0.035968,0.0601215
0.0690188,0.0354269,0.0592364
```

```
0.068596,0.0350684,0.0589558
0.0684185,0.0347405,0.0589444
0.0683732,0.0345409,0.059009
0.0684847,0.0344303,0.0592039
0.068509,0.0341592,0.0593886
0.068415,0.0340263,0.0593569
```

Bibliography

[1] Sebastian O. H. Madgwick, Andrew J. L. Harrison, and Ravi Vaidyanathan. "Estimation of IMU and MARG orientation using a gradient descent algorithm". In: *2011 IEEE International Conference on Rehabilitation Robotics*. 2011, pp. 1–7.

[2] Daniel Laidig et al. "BROAD—A Benchmark for Robust Inertial Orientation Estimation". In: *Data* 6.7 (2021).

[3] Don Koks. *Using rotations to build aerospace coordinate systems*. Tech. rep. Edinburgh, S. Aust.: DSTO System Sciences Laboratory, 2005.

[4] Mark Pedley and Michael Stanley. *Calculation of Orientation Matrices from Sensor Data*. Tech. rep. Eindhoven, Netherlands: NXP Semiconductors, 2014. URL: `https://github.com/memsindustrygroup/Open-Source-Sensor-Fusion/blob/master/docs/Orientation\%20Matrices.docx`.

[5] Sir William Rowan Hamilton LL.D. P.R.I.A. F.R.A.S. Hon. M. R. Soc. Ed. and Dub. Hon. or Corr. M. "II. On quaternions; or on a new system of imaginaries in algebra". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 25.163 (1844), pp. 10–13.

[6] J.B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton paperbacks. Princeton University Press, 1999.

[7] Landis Markley and John Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. Jan. 2014.

[8] Stanley W. Shepperd. "Quaternion from Rotation Matrix". In: *Journal of Guidance and Control* 1.3 (1978), pp. 223–224.

[9] Soheil Sarabandi and Federico Thomas. "Accurate Computation of Quaternions from Rotation Matrices". In: *Advances in Robot Kinematics*. 2018.

[10] Angelo Maria Sabatini. "Kalman-Filter-Based Orientation Determination Using Inertial/Magnetic Sensors: Observability Analysis and Performance Evaluation". In: *Sensors* 11.10 (2011), pp. 9182–9206.

[11] Jouni Hartikainen, Arno Solin, and Simo Särkkä. *Optimal filtering with Kalman filters and smoothers*. Version 1.3. Department of Biomedical Engineering and Computational Science, Aalto University School of Science. 2011.

[12] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45.

[13] Manon Kok, Jeroen D. Hol, and Thomas B. Schön. "Using Inertial Sensors for Position and Orientation Estimation". In: *Foundations and Trends® in Signal Processing* 11.1-2 (2017), pp. 1–153.

[14] Khronos Group. *The industry's foundation for High Performance Graphics*. last accessed on 2022.12.01. URL: `https://www.opengl.org/`.

[15] Camilla Löwy. *GLFW*. `https://github.com/glfw/glfw`. 2016.

[16] Slavomir Kaslev. *gl3w: Simple OpenGL core profile loading*. `https://github.com/skaslev/gl3w`. 2010.

[17] Omar Cornut. *Dear ImGui*. `https://github.com/ocornut/imgui`. 2014.

[18] Lisa Avila, Charles Law, and Bill Hoffman. *VTK: The Visualization Toolkit*. `https://gitlab.kitware.com/vtk/vtk`. 1998.

[19] Kitware. *CMake*. last accessed on 2022.12.01. URL: `https://cmake.org`.

[20] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. "Nonlinear Complementary Filters on the Special Orthogonal Group". In: *IEEE Transactions on Automatic Control* 53.5 (2008), pp. 1203–1218.

List of Figures

List of Algorithms