Using Inertial Sensors for Position and Orientation Estimation

Manon Kok*, Jeroen D. Hol† and Thomas B. Schön‡

*Delft Center for Systems and Control, Delft University of Technology, the Netherlands[1]
E-mail: m.kok-1@tudelft.nl
†Xsens Technologies B.V., Enschede, the Netherlands
E-mail: jeroen.hol@xsens.com
‡Department of Information Technology, Uppsala University, Sweden
E-mail: thomas.schon@it.uu.se

• **Please cite this version:**

Manon Kok, Jeroen D. Hol and Thomas B. Schön (2017), "Using Inertial Sensors for Position and Orientation Estimation", Foundations and Trends in Signal Processing: Vol. 11: No. 1-2, pp 1-153. http://dx.doi.org/10.1561/2000000094

Abstract

In recent years, microelectromechanical system (MEMS) inertial sensors (3D accelerometers and 3D gyroscopes) have become widely available due to their small size and low cost. Inertial sensor measurements are obtained at high sampling rates and can be integrated to obtain position and orientation information. These estimates are accurate on a short time scale, but suffer from integration drift over longer time scales. To overcome this issue, inertial sensors are typically combined with additional sensors and models. In this tutorial we focus on the signal processing aspects of position and orientation estimation using inertial sensors. We discuss different modeling choices and a selected number of important algorithms. The algorithms include optimization-based smoothing and filtering as well as computationally cheaper extended Kalman filter and complementary filter implementations. The quality of their estimates is illustrated using both experimental and simulated data.

[1]At the moment of publication Manon Kok worked as a Research Associate at the University of Cambridge, UK. A major part of the work has been done while she was a PhD student at Linköping University, Sweden.

Contents

Chapter 1

Introduction

In this tutorial, we discuss the topic of position and orientation estimation using inertial sensors. We consider two separate problem formulations. The first is estimation of orientation only, while the other is the combined estimation of both position and orientation. The latter is sometimes called *pose estimation*. We start by providing a brief background and motivation in §1.1 and explain what inertial sensors are and give a few concrete examples of relevant application areas of pose estimation using inertial sensors. In §1.2, we subsequently discuss how inertial sensors can be used to provide position and orientation information. Finally, in §1.3 we provide an overview of the contents of this tutorial as well as an outline of subsequent chapters.

1.1 Background and motivation

The term *inertial sensor* is used to denote the combination of a three-axis accelerometer and a three-axis gyroscope. Devices containing these sensors are commonly referred to as inertial measurement units (IMUs). Inertial sensors are nowadays also present in most modern smartphone, and in devices such as Wii controllers and virtual reality (VR) headsets, as shown in Figure 1.1.

A gyroscope measures the sensor's *angular velocity*, *i.e.* the rate of change of the sensor's orientation. An accelerometer measures the *external specific force* acting on the sensor. The specific force consists of both the *sensor's acceleration* and the *earth's gravity*. Nowadays, many gyroscopes and accelerometers are based on microelectromechanical system (MEMS) technology. MEMS components are small, light, inexpensive, have low power consumption and short start-up times. Their accuracy has significantly increased over the years.

There is a large and ever-growing number of application areas for inertial sensors, see *e.g.* [**?** **?** **?** **?**]. Generally speaking, inertial sensors can be used to provide information about the pose of any object that they are rigidly attached to. It is also possible to combine multiple inertial sensors to obtain information about the pose of separate connected objects. Hence, inertial sensors can be used to track human motion as illustrated in Figure 1.2. This is often referred to as motion capture. The application areas are as diverse as robotics, biomechanical analysis and motion capture for the movie and gaming industries. In fact, the use of inertial sensors for pose estimation is now common practice in for instance robotics and human motion tracking, see *e.g.* [**?** **?** **?**]. A recent survey [**?**] shows that 28% of the contributions to the IEEE International Conference on Indoor Positioning and Indoor Navigation (IPIN) make use of inertial sensors. Inertial sensors are also frequently used for pose estimation of cars, boats, trains and aerial vehicles, see *e.g.* [**?** **?**]. Examples of this are shown in Figure 1.3.

There exists a large amount of literature on the use of inertial sensors for position and orientation estimation. The reason for this is not only the large number of application areas. Important reasons are also that the estimation problems are nonlinear and that different parametrizations of the orientation need to be considered [**?** **?**], each with its own specific properties. Interestingly, approximative and relatively simple position and orientation estimation algorithms work quite well in practice. However, careful modeling and a careful choice of algorithms do improve the accuracy of the estimates.

In this tutorial we focus on the signal processing aspects of position and orientation estimation using inertial sensors, discussing different modeling choices and a number of important algorithms. These algorithms will provide the reader with a starting point to implement their own position and orientation estimation algorithms.

(a) Left bottom: an Xsens MTx IMU [**?**]. Left top: a Trivisio Colibri Wireless IMU [**?**]. Right: a Samsung Galaxy S4 mini smartphone.

(b) A Samsung gear VR.[1]

(c) A Wii controller containing an accelerometer and a MotionPlus expansion device containing a gyroscope.[2]

Figure 1.1: Examples of devices containing inertial sensors.

[1] 'Samsung Gear VR' available at `flic.kr/photos/pestoverde/15247458515` under CC BY 2.0 (`http://creativecommons.org/licenses/by/2.0`).

[2] 'WiiMote with MotionPlus' by Asmodai available at `https://commons.wikimedia.org/wiki/File:WiiMote_with_MotionPlus.JPG` under CC BY SA (`https://creativecommons.org/licenses/by-sa/3.0/`).

(a) Back pain therapy using serious gaming. IMUs are placed on the chest-bone and on the pelvis to estimate the movement of the upper body and pelvis. This movement is used to control a robot in the game and promotes movements to reduce back pain.

(b) Actor Seth MacFarlane wearing 17 IMUs to capture his motion and animate the teddy bear Ted. The IMUs are placed on different body segments and provide information about the relative position and orientation of each of these segments.

Figure 1.2: Examples illustrating the use of multiple IMUs placed on the human body to estimate its pose. Courtesy of Xsens Technologies.

(a) Inertial sensors are used in combination with GNSS measurements to estimate the position of the cars in a challenge on cooperative and autonomous driving.

(b) Due to their small size and low weight, IMUs can be used to estimate the orientation for control of an unmanned helicopter.

Figure 1.3: Examples illustrating the use of a single IMU placed on a moving object to estimate its pose. Courtesy of Xsens Technologies.

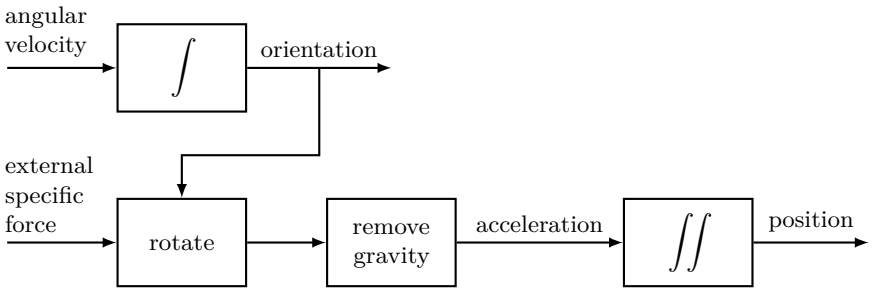


Figure 1.4: Schematic illustration of dead-reckoning, where the accelerometer measurements (external specific force) and the gyroscope measurements (angular velocity) are integrated to position and orientation.

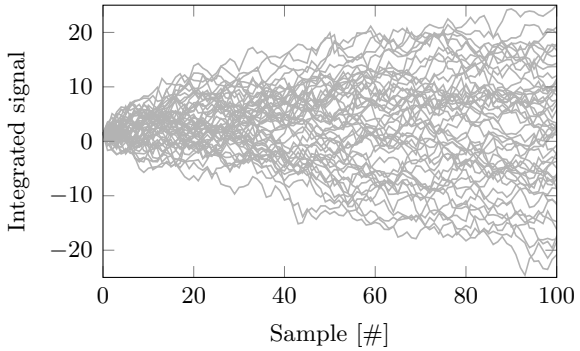


Figure 1.5: Integration of a white noise signal $y_t \sim \mathcal{N}(0, 1)$ for 50 noise realizations.

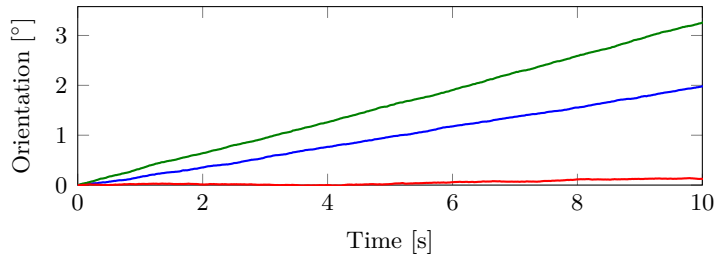
1.2 Using inertial sensors for position and orientation estimation

As illustrated in §1.1, inertial sensors are frequently used for navigation purposes where the position and the orientation of a device are of interest. Integration of the gyroscope measurements provides information about the orientation of the sensor. After subtraction of the earth's gravity, double integration of the accelerometer measurements provides information about the sensor's position. To be able to subtract the earth's gravity, the orientation of the sensor needs to be known. Hence, estimation of the sensor's position and orientation are inherently linked when it comes to inertial sensors. The process of integrating the measurements from inertial sensors to obtain position and orientation information, often called *dead-reckoning*, is summarized in Figure 1.4.

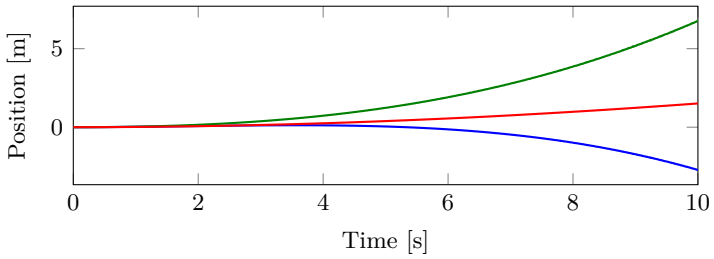
If the initial pose would be known, and if perfect models for the inertial sensor measurements would exist, the process illustrated in Figure 1.4 would lead to perfect pose estimates. In practice, however, the inertial measurements are noisy and biased as will be discussed in more detail in §2.4. Because of this, the integration steps from angular velocity to rotation and from acceleration to position introduce *integration drift*. This is illustrated in Example 1.1.

Example 1.1 (Integration drift) *Let us first focus on the general case of measuring a quantity that is constant and equal to zero. The integrated and double integrated signals are therefore also equal to zero. However, let us now assume that we measure this quantity using a non-perfect sensor. In case our measurements are corrupted by a constant bias, integration of these measurements will lead to a signal which grows linearly with time. Double integration leads to a signal that instead grows quadratically with time. If the sensor instead measures a zero-mean white noise signal, the expected value of the integrated measurements would be zero, but the variance would grow with time. This is illustrated in Figure 1.5 for the integration of a signal $y_t = e_t$ with $e_t \sim \mathcal{N}(0, 1)$. Hence, integration drift is both due to integration of a constant bias and due to integration of noise.*

*To illustrate integration drift using experimental data, a stationary data set is collected with a Sony Xperia Z5 Compact smartphone using the app described in [**?**]. The smartphone contains accelerometers and gyroscopes produced by Invensense [**?**]. We integrate the inertial measurements to obtain position*



(a) Integrated orientation for the position in x- (blue), y- (green) and z-direction (red).



(b) Integrated position for rotation around the x-axis (blue), the y-axis (green) and the z-axis (red).

Figure 1.6: Position and orientation estimates based on dead-reckoning of the inertial sensors only. The data is collected with a Sony Xperia Z5 Compact smartphone that is lying stationary on a table.

and orientation estimates. Since the smartphone is kept stationary during the data collection, we expect the position and orientation to remain the same. However, the orientation estimates drift a few degrees over 10 *seconds as shown in Figure 1.6(a). Note that the integration drift is not the same for all axes. This is mainly due to a different sensor bias in the different axes. This will be studied further in Example 2.3, where the same data set is used to study the sensor characteristics. As shown in Figure 1.6(b), the position drifts several meters over* 10s*. The reason for this is two-fold. First, the accelerometer measurements need to be integrated twice. Second, the orientation estimates need to be used to subtract the gravity and any errors in this will result in* leakage of gravity *into the other components.*

From the example above, it can be concluded that errors in the measurements have a large impact on the quality of the estimated position and orientation using inertial sensors only. This is particularly the case for position, which relies both on double integration of the acceleration and on accurate orientation estimates to subtract the earth's gravity. Because of this, inertial sensors need to be supplemented with other sensors and other models to obtain accurate position and orientation estimates.

Inertial sensors provide pose estimates at high sampling rates which are accurate on a short time scale but drift over longer time scales. They are therefore very suitable for being combined with sensors with a lower sampling rate but with information that does not drift over time. For pose estimation, inertial sensors are often combined with measurements from for instance a global navigation satellite system (GNSS) [**?** **?** **?**], an ultrawideband (UWB) system [**?** **?** **?** **?** **?**] or cameras [**?** **?** **?** **?**]. For orientation estimation, they are often used in combination with magnetometers, which measure the direction of the magnetic field [**?** **?**].

This tutorial aims at giving an introduction on how to use inertial sensors for position and orientation estimation, but also on how to combine them with additional information. These additional sensors are not the focus of this paper but simple models will be used for magnetometers and sensors providing position information to illustrate the combined use of these sensors.

1.3 Tutorial content and its outline

To obtain accurate position and orientation estimates using inertial sensors in combination with additional measurements and models, a number of important things need to be considered. First, the quantities measured by the inertial sensors need to be accurately described and the sources of error need

to be characterized. This is the topic of Chapter 2. Note that throughout the tutorial, we will focus on MEMS inertial sensors and consider both data from standalone IMUs and from smartphones. This implies that we do not focus on for instance mechanical or optical gyroscopes and on mechanical or solid-state accelerometers [**?**]. These sensors may have characteristics that are quite different from the MEMS inertial sensors considered here.

Based on the analysis of the sensors in Chapter 2 and on additional analysis of the application at hand, models can be constructed. This is the topic of Chapter 3, where we will also discuss different parametrizations of orientation. This will highlight the challenges in parametrizing and estimating orientations and show that the orientation estimation problem is inherently nonlinear. Furthermore, we will present two models that can be used for position and orientation estimation. The first is a model for pose estimation using inertial measurements in combination with position measurements. The second is a model for orientation estimation, using inertial and magnetometer measurements.

In Chapter 4, different algorithms for position and orientation estimation will be introduced. The general structure of the algorithms will be discussed, after which explicit algorithms for orientation estimation using inertial and magnetometer measurements are given. We will also discuss how the algorithms can be extended to pose estimation when position measurements are available. Some general characteristics of the two estimation problems will be given and the quality of the estimates from the different algorithms will be analyzed. Which algorithm is most suitable for which application depends strongly on the computational power that is available, the accuracy that is required and the characteristics of the problem at hand.

In Chapter 4, we assume that the sensors are properly calibrated. However, calibration of the sensors is important to for instance estimate the inertial sensor biases. Furthermore, calibration is specifically of concern when combining inertial data with other sensors. In these cases, it is important that the inertial sensor axes and the axes of the additional sensors are aligned. Sensor calibration is the topic of Chapter 5. As an illustrative example, we will consider the estimation of an unknown gyroscope bias.

Our focus in this tutorial is to present models and algorithms that can be used for position and orientation estimation using inertial measurements. Because of this, we assume fairly simple models for the additional information — the magnetometer and position measurements. In Chapter 6, we will discuss how the algorithms from Chapter 4 can be used in more complex settings. For example, we will consider the cases of more complex position information in the form of images from a camera, the presence of non-Gaussian measurement noise and the availability of additional information that can be exploited. The information provided by the inertial sensors remains one of the main building blocks of algorithms that can be used for these cases. Adaptations of the algorithms presented in Chapter 4 can therefore be used to also solve these more complex scenarios. We will end this tutorial with some concluding remarks in Chapter 7.

Chapter 2

Inertial Sensors

To combine inertial measurements with additional sensors and models for position and orientation estimation, it is important to accurately describe the quantities measured by the inertial sensors as well as to characterize the typical sensor errors. This will be the topic of this chapter. It will serve as a basis for the probabilistic models discussed in Chapter 3.

As discussed in Chapter 1, accelerometers and gyroscopes measure the specific force and the angular velocity, respectively. In §2.2 and §2.3, we will discuss these quantities in more detail. To enable a discussion about this, in §2.1 a number of coordinate frames and the transformations between them will be discussed. We assume that we have 3D accelerometers and 3D gyroscopes, *i.e.* that the sensors have three sensitive axes along which these physical quantities are measured. They are measured in terms of an output voltage which is converted to a physical measurement based on calibration values obtained in the factory. Even though the sensors are typically calibrated in the factory, (possibly time-varying) errors can still remain. In §2.4, the most commonly occurring sensor errors are discussed.

2.1 Coordinate frames

In order to discuss the quantities measured by the accelerometer and gyroscope in more detail, a number of coordinate frames need to be introduced:

The body frame b is the coordinate frame of the moving IMU. Its origin is located in the center of the accelerometer triad and it is aligned to the casing. All the inertial measurements are resolved in this frame.

The navigation frame n is a local geographic frame in which we want to navigate. In other words, we are interested in the position and orientation of the b-frame with respect to this frame. For most applications it is defined stationary with respect to the earth. However, in cases when the sensor is expected to move over large distances, it is customary to move and rotate the n-frame along the surface of the earth. The first definition is used throughout this tutorial, unless mentioned explicitly.

The inertial frame i is a stationary frame. The IMU measures linear acceleration and angular velocity with respect to this frame. Its origin is located at the center of the earth and its axes are aligned with respect to the stars.

The earth frame e coincides with the i-frame, but rotates with the earth. That is, it has its origin at the center of the earth and axes which are fixed with respect to the earth.

The n, i and e coordinate frames are illustrated in Figure 2.1. We use a superscript to indicate in which coordinate frame a vector is expressed. Vectors can be rotated from one coordinate frame to another using a rotation matrix. We use a double superscript to indicate from which coordinate frame to which coordinate frame the rotation is defined. An illustration is given in Example 2.1.

Example 2.1 (Rotation of vectors to different coordinate frames) *Consider a vector x expressed in the body frame b. We denote this vector by x^b. The rotation matrix R^{nb} rotates a vector from the body frame b to the navigation frame n. Conversely, the rotation from navigation frame n to body frame*

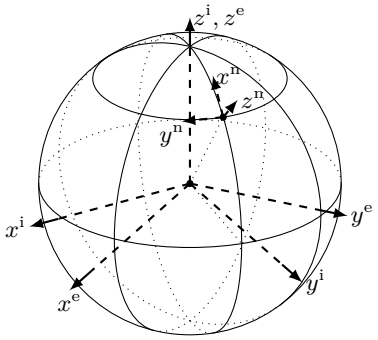


Figure 2.1: An illustration of three of the coordinate frames discussed in §2.1: the n-frame at a certain location on the earth, the e-frame rotating with the earth and the i-frame.

b is denoted $R^{bn} = (R^{nb})^{\mathsf{T}}$. Hence, the vector x expressed in the body frame (x^b) and expressed in the navigation frame (x^n) are related according to

$$x^n = R^{nb}x^b, \qquad x^b = (R^{nb})^{\mathsf{T}}x^n = R^{bn}x^n. \tag{2.1}$$

2.2 Angular velocity

The gyroscope measures the angular velocity of the body frame with respect to the inertial frame, expressed in the body frame [**?**], denoted by $\omega_{\mathrm{ib}}^{\mathrm{b}}$. This angular velocity can be expressed as

$$\omega_{\mathrm{ib}}^{\mathrm{b}} = R^{\mathrm{bn}}\left(\omega_{\mathrm{ie}}^{\mathrm{n}} + \omega_{\mathrm{en}}^{\mathrm{n}}\right) + \omega_{\mathrm{nb}}^{\mathrm{b}}, \tag{2.2}$$

where R^{bn} is the rotation matrix from the navigation frame to the body frame. The *earth rate*, *i.e.* the angular velocity of the earth frame with respect to the inertial frame is denoted by ω_{ie}. The earth rotates around its own z-axis in 23.9345 hours with respect to the stars [**?**]. Hence, the earth rate is approximately $7.29 \cdot 10^{-5}$ rad/s.

In case the navigation frame is not defined stationary with respect to the earth, the angular velocity ω_{en}, *i.e.* the *transport rate* is non-zero. The angular velocity required for navigation purposes — in which we are interested when determining the orientation of the body frame with respect to the navigation frame — is denoted by ω_{nb}.

2.3 Specific force

The accelerometer measures the specific force f in the body frame b [**?**]. This can be expressed as

$$f^{\mathrm{b}} = R^{\mathrm{bn}}(a_{\mathrm{ii}}^{\mathrm{n}} - g^{\mathrm{n}}), \tag{2.3}$$

where g denotes the gravity vector and $a_{\mathrm{ii}}^{\mathrm{n}}$ denotes the linear acceleration of the sensor expressed in the navigation frame, which is

$$a_{\mathrm{ii}}^{\mathrm{n}} = R^{\mathrm{ne}}R^{\mathrm{ei}}a_{\mathrm{ii}}^{\mathrm{i}}. \tag{2.4}$$

The subscripts on the linear acceleration a are used to indicate in which frame the differentiation is performed. For navigation purposes, we are interested in the position of the sensor in the navigation frame p^{n} and its derivatives as performed in the navigation frame

$$\tfrac{\mathrm{d}}{\mathrm{d}t}p^{\mathrm{n}}\big|_{\mathrm{n}} = v_{\mathrm{n}}^{\mathrm{n}}, \qquad \tfrac{\mathrm{d}}{\mathrm{d}t}v^{\mathrm{n}}\big|_{\mathrm{n}} = a_{\mathrm{nn}}^{\mathrm{n}}. \tag{2.5}$$

A relation between a_{ii} and a_{nn} can be derived by using the relation between two rotating coordinate frames. Given a vector x in a coordinate frame u,

$$\tfrac{\mathrm{d}}{\mathrm{d}t}x^{\mathrm{u}}\big|_{\mathrm{u}} = \tfrac{\mathrm{d}}{\mathrm{d}t}R^{\mathrm{uv}}x^{\mathrm{v}}\big|_{\mathrm{u}} = R^{\mathrm{uv}}\,\tfrac{\mathrm{d}}{\mathrm{d}t}x^{\mathrm{v}}\big|_{\mathrm{v}} + \omega_{\mathrm{uv}}^{\mathrm{u}} \times x^{\mathrm{u}}, \tag{2.6}$$

where $\omega^{\mathrm{u}}_{\mathrm{uv}}$ is the angular velocity of the v-frame with respect to the u-frame, expressed in the u-frame. For a derivation of this relation in the context of inertial navigation, see [**?** **?**]. For a general introduction, see any textbook on dynamics, *e.g.* [**?** **?**].

Using the fact that

$$p^{\mathrm{i}} = R^{\mathrm{ie}} p^{\mathrm{e}}, \tag{2.7}$$

the velocity v_{i} and acceleration a_{ii} can be expressed as

$$\begin{aligned} v^{\mathrm{i}}_{\mathrm{i}} &= \tfrac{\mathrm{d}}{\mathrm{d}t} p^{\mathrm{i}}\big|_{\mathrm{i}} = \tfrac{\mathrm{d}}{\mathrm{d}t} R^{\mathrm{ie}} p^{\mathrm{e}}\big|_{\mathrm{i}} = R^{\mathrm{ie}} \tfrac{\mathrm{d}}{\mathrm{d}t} p^{\mathrm{e}}\big|_{\mathrm{e}} + \omega^{\mathrm{i}}_{\mathrm{ie}} \times p^{\mathrm{i}} = v^{\mathrm{i}}_{\mathrm{e}} + \omega^{\mathrm{i}}_{\mathrm{ie}} \times p^{\mathrm{i}}, && \text{(2.8a)} \\ a^{\mathrm{i}}_{\mathrm{ii}} &= \tfrac{\mathrm{d}}{\mathrm{d}t} v^{\mathrm{i}}_{\mathrm{i}}\big|_{\mathrm{i}} = \tfrac{\mathrm{d}}{\mathrm{d}t} v^{\mathrm{i}}_{\mathrm{e}}\big|_{\mathrm{i}} + \tfrac{\mathrm{d}}{\mathrm{d}t} \omega^{\mathrm{i}}_{\mathrm{ie}} \times p^{\mathrm{i}}\big|_{\mathrm{i}} \\ &= a^{\mathrm{i}}_{\mathrm{ee}} + 2\omega^{\mathrm{i}}_{\mathrm{ie}} \times v^{\mathrm{i}}_{\mathrm{e}} + \omega^{\mathrm{i}}_{\mathrm{ie}} \times \omega^{\mathrm{i}}_{\mathrm{ie}} \times p^{\mathrm{i}}, && \text{(2.8b)} \end{aligned}$$

where we have made use of (2.5), (2.6), and the fact that the angular velocity of the earth is constant, *i.e.* $\frac{\mathrm{d}}{\mathrm{d}t}\omega^{\mathrm{i}}_{\mathrm{ie}} = 0$. Using the relation between the earth frame and the navigation frame

$$p^{\mathrm{e}} = R^{\mathrm{en}} p^{\mathrm{n}} + n^{\mathrm{e}}_{\mathrm{ne}}, \tag{2.9}$$

where n_{ne} is the distance from the origin of the earth coordinate frame to the origin of the navigation coordinate frame, expressions similar to (2.8) can be derived. Note that in general it can not be assumed that $\frac{\mathrm{d}}{\mathrm{d}t}\omega_{\mathrm{en}} = 0$. Inserting the obtained expressions into (2.8), it is possible to derive the relation between a_{ii} and a_{nn}. Instead of deriving these relations, we will assume that the navigation frame is fixed to the earth frame, and hence R^{en} and $n^{\mathrm{e}}_{\mathrm{ne}}$ are constant and

$$\begin{aligned} v^{\mathrm{e}}_{\mathrm{e}} &= \tfrac{\mathrm{d}}{\mathrm{d}t} p^{\mathrm{e}}\big|_{\mathrm{e}} = \tfrac{\mathrm{d}}{\mathrm{d}t} R^{\mathrm{en}} p^{\mathrm{n}}\big|_{\mathrm{e}} = R^{\mathrm{en}} \tfrac{\mathrm{d}}{\mathrm{d}t} p^{\mathrm{n}}\big|_{\mathrm{n}} = v^{\mathrm{e}}_{\mathrm{n}}, && \text{(2.10a)} \\ a^{\mathrm{e}}_{\mathrm{ee}} &= \tfrac{\mathrm{d}}{\mathrm{d}t} v^{\mathrm{e}}_{\mathrm{e}}\big|_{\mathrm{e}} = \tfrac{\mathrm{d}}{\mathrm{d}t} v^{\mathrm{e}}_{\mathrm{n}}\big|_{\mathrm{n}} = a^{\mathrm{e}}_{\mathrm{nn}}. && \text{(2.10b)} \end{aligned}$$

This is a reasonable assumption as long as the sensor does not travel over significant distances as compared to the size of the earth and it will be one of the model assumptions that we will use in this tutorial. More on the modeling choices will be discussed in Chapter 3.

Inserting (2.10) into (2.8) and rotating the result, it is possible to express $a^{\mathrm{n}}_{\mathrm{ii}}$ in terms of $a^{\mathrm{n}}_{\mathrm{nn}}$ as

$$a^{\mathrm{n}}_{\mathrm{ii}} = a^{\mathrm{n}}_{\mathrm{nn}} + 2\omega^{\mathrm{n}}_{\mathrm{ie}} \times v^{\mathrm{n}}_{\mathrm{n}} + \omega^{\mathrm{n}}_{\mathrm{ie}} \times \omega^{\mathrm{n}}_{\mathrm{ie}} \times p^{\mathrm{n}}, \tag{2.11}$$

where a_{nn} is the acceleration required for navigation purposes. The term $\omega^{\mathrm{n}}_{\mathrm{ie}} \times \omega^{\mathrm{n}}_{\mathrm{ie}} \times p^{\mathrm{n}}$ is known as the *centrifugal acceleration* and $2\omega^{\mathrm{n}}_{\mathrm{ie}} \times v^{\mathrm{n}}_{\mathrm{n}}$ is known as the *Coriolis acceleration*. The centrifugal acceleration is typically absorbed in the (local) gravity vector. In Example 2.2, we illustrate the magnitude of both the centrifugal and the Coriolis acceleration.

Example 2.2 (Magnitude of centrifugal and Coriolis acceleration) *The centrifugal acceleration depends on the location on the earth. It is possible to get a feeling for its magnitude by considering the property of the cross product stating that*

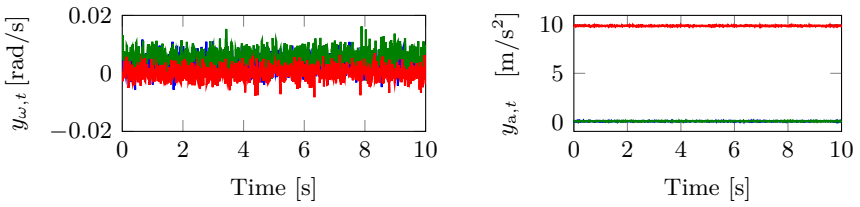
$$\|\omega^n_{ie} \times \omega^n_{ie} \times p^n\|_2 \leq \|\omega^n_{ie}\|_2 \|\omega^n_{ie}\|_2 \|p^n\|_2. \tag{2.12}$$

Since the magnitude of ω_{ie} *is approximately* $7.29 \cdot 10^{-5}$ rad/s *and the average radius of the earth is* 6371 km *[**?**], the magnitude of the centrifugal acceleration is less than or equal to* $3.39 \cdot 10^{-2}$ m/s^2.

The Coriolis acceleration depends on the speed of the sensor. Let us consider a person walking at a speed of 5 km/h. *In that case the magnitude of the Coriolis acceleration is approximately* $2.03 \cdot 10^{-4}$ m/s^2. *For a car traveling at* 120 km/h, *the magnitude of the Coriolis acceleration is instead* $4.86 \cdot 10^{-3}$ m/s^2.

2.4 Sensor errors

As discussed in §2.2 and §2.3, the gyroscope measures the angular velocity $\omega^{\mathrm{b}}_{\mathrm{ib}}$ and the accelerometer measures the specific force f^{b}. However, as already briefly mentioned in §1.2, there are several reasons for why this is not exactly the case. Two of these reasons are a slowly time-varying sensor bias and the presence of measurement noise. The sensor errors in the inertial measurements are illustrated in Example 2.3 using experimental data.



(a) Gyroscope measurements $y_{\omega,t}$ which we expect to consist only of the earth's angular velocity.

(b) Accelerometer measurements $y_{a,t}$ which we expect to consist of the gravity vector, the centrifugal acceleration and the Coriolis acceleration.

Figure 2.2: Inertial measurements for 10 seconds of stationary data. As can be seen, the measurements are corrupted by noise and have a bias.

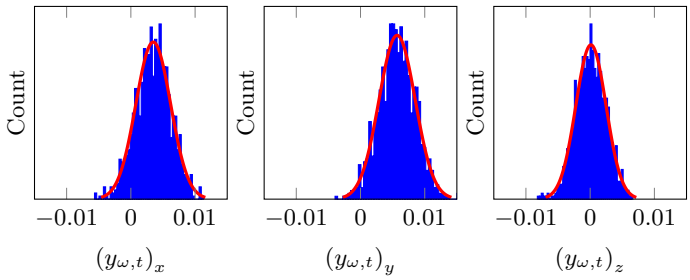


Figure 2.3: Histogram (blue) of the gyroscope measurements for 10 seconds of data from a stationary sensor and a Gaussian fit (red) to the data. As can be seen, the measurement noise looks quite Gaussian.

Example 2.3 (Inertial sensor measurements and their errors) *In Figures 2.2–2.4, gyroscope and accelerometer measurements are displayed for around* 10 *seconds of stationary data collected with a Sony Xperia Z5 Compact smartphone. Since the smartphone is stationary, the gyroscope is expected to only measure the earth's angular velocity. However, as can be seen in Figure 2.2(a), the gyroscope measurements are corrupted by noise. As shown in Figure 2.3, this noise can be seen to be quite Gaussian. Furthermore, the measurements can be seen to be biased.*

During the stationary period, we would expect the accelerometer to measure the gravity, the centrifugal acceleration and the Coriolis acceleration. Note that again the measurements are corrupted by noise, which can be seen to be quite Gaussian in Figure 2.4. The x- and y-components of the accelerometer measurements are not zero-mean. This can be due to the fact that the table on which the smartphone lies is not completely flat, implying that part of the gravity vector is measured in these components. It can also reflect a sensor bias. The z-component is actually larger than expected which indicates the presence of an accelerometer bias at least in this axis.

*Note that from the above discussion it can be concluded that it is more straightforward to determine the gyroscope bias than it is to determine the accelerometer bias. To be able to estimate the gyroscope bias, it is sufficient to leave the sensor stationary. For the accelerometer, however, it is in that case difficult to distinguish between a bias and a table that is not completely flat. For more information about accelerometer or general inertial sensor calibration, see [**? ? ?**].*

The gyroscope in the smartphone is automatically recalibrated during stationary time periods. The measurements shown in Figure 2.2(a) have not been corrected for this (so-called uncalibrated or raw data). The reason why the gyroscope is calibrated during stationary periods, is because its bias is slowly time-varying. As an example, let us consider a data set of 55 *minutes. The gyroscope bias during the first minute of the data set was* $\begin{pmatrix}35.67 & 56.22 & 0.30\end{pmatrix}^{\mathsf{T}} \cdot 10^{-4}$ rad/s, *while the gyroscope bias during the last minute of the data set was* $\begin{pmatrix}37.01 & 53.17 & -1.57\end{pmatrix}^{\mathsf{T}} \cdot 10^{-4}$ rad/s.

The performance of IMUs is often specified in terms of their *Allan variance* [**? ? ?**]. The Allan variance gives information about the sensor errors for stationary conditions, *i.e.* in a stable climate without exciting the system. It studies the effect of averaging measurements for different *cluster times*

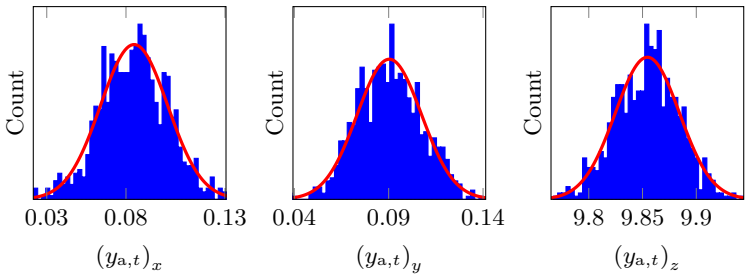


Figure 2.4: Histogram (blue) of the accelerometer measurements for 10 seconds of data from a stationary sensor and a Gaussian fit (red) to the data. As can be seen, the measurement noise looks quite Gaussian. Note the different scales on the horizontal axis.

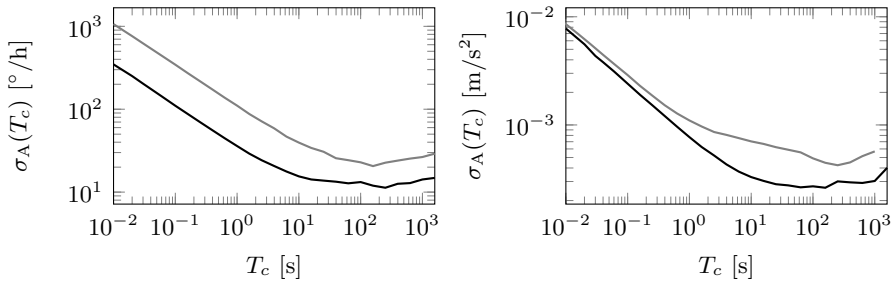


Figure 2.5: Left: Allan deviation for two gyroscopes. Right: Allan deviation for two accelerometers. Reproduced with permission from [**?**].

T_c. Typically, the Allan *standard deviation* $\sigma_A(T_c)$ is plotted against the cluster time T_c as illustrated in Figure 2.5. This figure shows the characteristic behavior of the Allan variance for inertial sensors. To study it more in detail, we will discuss two components of the Allan variance that are typically of concern for inertial sensors: the white noise and the bias instability.

Assume, as in Example 1.1, that we have a white noise signal with standard deviation σ. A longer averaging time would for this signal lead to values closer to zero. The contribution to the Allan standard deviation from the white noise component is given by $\sigma_A(T_c) = \frac{\sigma}{\sqrt{n}}$ where n is the number of samples averaged over. This corresponds to a line with slope $-1/2$ in a log–log plot. For instance in the Allan deviation for the gyroscope in Figure 2.5, the lines can be seen to have a slope of $-1/2$ until around $10-20$ s, which indicates that the white noise is the dominating source of error for these short integration times.

A constant bias does not have any effect on the Allan variance diagram. However, in case the bias changes, longer averaging times will no longer be beneficial. Hence, the Allan variance diagrams in Figure 2.5 show a deviation from the slope $-1/2$ for longer averaging times.

The Allan variance is a useful tool to study and compare the noise characteristics of inertial sensors. However, it only considers stationary conditions. In dynamic conditions, a large number of other error sources potentially come into play, see *e.g.* [**?** **?**]. These are for instance related to the fact that the sensors sample at discrete times. Hence, to capture high-frequency signals, high sampling frequencies are desired [**?** **?**]. Furthermore, large dynamics can lead to erroneous or saturated measurements. Other errors that are not included are for instance changes in the sensitivity of the axes due to changes in temperature. We should therefore never just rely on the Allan variance when deciding which sensor to use in a particular application.

Chapter 3

Probabilistic Models

Pose estimation is about estimating the position and orientation of the body frame b in the navigation frame n. This problem is illustrated in Figure 3.1, where the position and orientation of the body changes from time t_1 to time t_2. In this chapter, we will introduce the concept of probabilistic models and discuss different modeling choices when using inertial sensors for pose estimation.

The subject of probabilistic modeling is introduced in §3.1. Most complexity in pose estimation lies in the nonlinear nature of the orientation and the fact that orientation can be parametrized in different ways. How to parametrize the orientation is therefore a crucial modeling choice in any pose estimation algorithm. Because of this, we will discuss different parametrizations for the orientation in §3.2 and in §3.3 we will discuss how these different parametrizations can be used in probabilistic modeling.

Our probabilistic models consist of three main components. First, in §3.4, we introduce models describing the knowledge about the pose that can be inferred from the measurements. Second, in §3.5, we model how the sensor pose changes over time. Finally, in §3.6, models of the initial pose are introduced.

The chapter will conclude with a discussion on the resulting probabilistic models in §3.7. The models that will be used in the position and orientation estimation algorithms in Chapter 4 will be introduced in this section.

3.1 Introduction

Probabilistic models constitute the foundation of the estimation algorithms in Chapter 4. In this section we will introduce the concept of probabilistic modeling and the notation that is used in building our models. Models are used to describe the information about the *dynamics* and the available *measurements*. These models are subsequently used in combination with the measurements to *infer* some knowledge. The knowledge that we are interested in is the pose of the sensor and we use information about the *sensor dynamics* and the available measurements (amongst others, inertial measurements). A simplified case where probabilistic modeling is used to estimate the position of a sensor is given in Example 3.1.

Example 3.1 (Probabilistic modeling) *Let us estimate the 2D position p_t of a sensor at time t from*

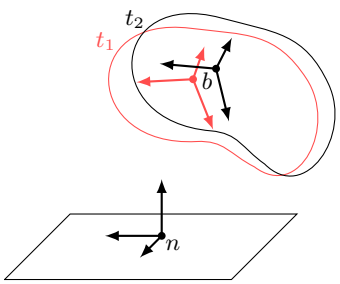


Figure 3.1: An illustration of the pose estimation problem. We want to express the position and orientation of the moving body frame b at times t_1 and t_2 with respect to the navigation frame n.

two position measurements

$$y_t^1 = \begin{pmatrix} 0 & 0 \end{pmatrix}^\mathsf{T}, \qquad y_t^2 = \begin{pmatrix} 2 & 0 \end{pmatrix}^\mathsf{T}.$$

A straightforward suggestion for an estimate of the position would be $\hat{p}_t = \begin{pmatrix} 1 & 0 \end{pmatrix}^\mathsf{T}$. *Let us now assume that we know the accuracy of the sensors and represent this in terms of the following probabilistic models*

$$\begin{aligned} y_t^1 &= p_t + e_t^1, & e_t^1 &\sim \mathcal{N}(0, 0.25\,\mathcal{I}_2), \\ y_t^2 &= p_t + e_t^2, & e_t^2 &\sim \mathcal{N}(0, \mathcal{I}_2), \end{aligned}$$

where $\mathcal{I}_2$ *denotes a* 2×2 *identity matrix. A reasonable position estimate would instead be*

$$p_t \sim \mathcal{N}\left(\begin{pmatrix} 0.4 \\ 0 \end{pmatrix}, 0.2\,\mathcal{I}_2 \right).$$

We will not go into details about how this estimate is derived. Instead, we would like to point out two differences between this position estimate and our initial suggestion $\hat{p}_t$. *First, based on the knowledge of the accuracy of the sensors, it is sensible to trust the measurement from the first sensor more than the measurement from the second sensor. Our improved position estimate is therefore closer to the measurement of the first sensor than our initial suggestion* $\hat{p}_t$. *Furthermore, based on the accuracy of the sensors, it is possible to derive the accuracy of our estimate.*

Now consider the case where we are also interested in estimating the position p_{t+1}. *Knowledge that the sensor is worn by a human or placed in a car, would give us information about how far the sensor can travel from time* t *to time* $t+1$. *If the sensor would be placed in for instance a train, the motion would even be constrained to be along the tracks. Incorporating this information about the dynamics of the sensor will improve the estimate of* p_{t+1}.

We split the knowledge that we want to infer into the unknown *time-varying states* x_t for $t = 1, \ldots, N$, or equivalently $x_{1:N}$, and the unknown *constant parameters* θ. We denote the measurements by y_k for $k = 1, \ldots, K$. The times k at which these measurements are obtained do not necessarily correspond with the times t at which the states are defined. It is also not necessary for all sensors to sample at the same frequency. As discussed in §2.4, the inertial sensors are typically sampled at fairly high rates to capture high-frequency dynamics. In stand-alone, wired IMUs, all sensors typically have the same, constant sampling frequency. Specifically in the case of wireless sensors and smartphones, however, the sampling frequencies can vary both over sensors and over time. In the remainder, we assume that the times t at which the states are defined coincide with the times k at which the gyroscopes sample. Hence, we denote the gyroscope measurements $y_{\omega,t}$ with $t = 1, \ldots N$. For notational convenience, we will also use the subscript t for the measurements from other sensors. Note that these are not required to actually sample at each time t for $t = 1, \ldots N$. For instance, magnetometers in smartphones often sample either at equal or at half the sampling frequencies of the inertial sensors, while position aiding sensors like for instance GNSS or UWB typically sample at much lower sampling frequencies.

Our aim is now to infer information about the states $x_{1:N}$ and the parameters θ using the measurements $y_{1:N}$ and the probabilistic models. This can be expressed in terms of a *conditional probability distribution*

$$p(x_{1:N}, \theta \mid y_{1:N}), \tag{3.1}$$

where $p(a \mid b)$ denotes the conditional probability of a given b. In the pose estimation problem, we are interested in obtaining *point estimates* which we denote $\hat{x}_{1:N}$ and $\hat{\theta}$. It is typically also highly relevant to know how *certain* we are about these estimates. This is often expressed in terms of a *covariance*. When the distribution (3.1) is Gaussian, the distribution is completely described in terms of its mean and covariance.

In (3.1) we assume that all measurements $y_{1:N}$ are used to obtain the posterior distribution of $x_{1:N}$ and θ. This is referred to as *smoothing*. Although it makes sense to use all available information to obtain the best estimates, a downside of smoothing is that we need to wait until all measurements are collected before the pose can be computed. Because of this, in many applications, we are also interested in *filtering*. In filtering we estimate x_t using all measurements up to and including time t. One way of dealing with constant parameters in filtering is to treat them as slowly time-varying. In this case, they

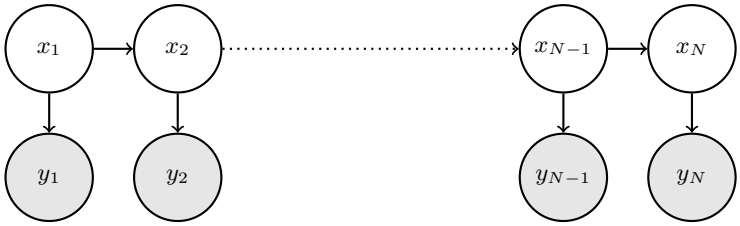


Figure 3.2: An illustration of the structure of the pose estimation problem.

can be considered to be included in the time-varying states x_t. The filtering problem can be expressed in terms of the conditional probability distribution

$$p(x_t \mid y_{1:t}). \tag{3.2}$$

We have now introduced smoothing, where the states $x_{1:N}$ are estimated simultaneously, and filtering, where at each time instance the state x_t is estimated. There is a large range of intermediate methods, where a batch of states $x_{t-L_1:t+L_2}$, with L_1 and L_2 being positive integers, is estimated using the measurements $y_{1:t}$. This is related to fixed-lag smoothing and moving horizon estimation [**?** **?**].

The topic of how to estimate the conditional probability distributions for position and orientation estimation will be introduced in Chapter 4. We will now instead take a closer look at these distributions and their different components. A fundamental assumption here is that we assume that our models possess the *Markov property*, implying that all information up to the current time t is contained in the state x_t. This is illustrated in Figure 3.2 in terms of a *probabilistic graphical model* [**?**]. The state x_{t+1} can be seen to depend on x_t and to result in the measurements y_{t+1}. It is conditionally independent of $x_{1:t-1}$ given the state x_t. Using Bayes' rule and the Markov property, the conditional distributions (3.1) and (3.2) can be decomposed as

$$p(x_{1:N}, \theta \mid y_{1:N}) \propto p(\theta)p(x_1 \mid \theta) \prod_{t=2}^{N} p(x_t \mid x_{t-1}, \theta) \prod_{t=1}^{N} p(y_t \mid x_t, \theta), \tag{3.3a}$$

$$p(x_t \mid y_{1:t}) \propto p(y_t \mid x_t)p(x_t \mid y_{1:t-1}). \tag{3.3b}$$

The predictive distribution $p(x_t \mid y_{1:t-1})$ can be computed by *marginalizing out* the previous state x_{t-1} as

$$p(x_t \mid y_{1:t-1}) = \int p(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1}) \, \mathrm{d}x_{t-1}. \tag{3.4}$$

In (3.3), $p(\theta)$ and $p(x_1 \mid \theta)$ encode our *prior* information of θ and the knowledge of the state x_1 given θ, respectively. The *dynamics* are modeled in terms of $p(x_{t+1} \mid x_t, \theta)$ and $p(x_{t+1} \mid x_t)$. The distributions $p(y_t \mid x_t, \theta)$ and $p(y_t \mid x_t)$ model the information given by the measurements about the state and the parameters.

The dynamics of the state can be modeled in terms of a nonlinear function $f_t(\,\cdot\,)$ as

$$x_{t+1} = f_t(x_t, w_t). \tag{3.5}$$

The uncertainty of the dynamic model is modeled in terms of w_t, which is often referred to as the *process noise*. The model (3.5) provides information about the distribution $p(x_{t+1} \mid x_t)$. More explicitly, if w_t is Gaussian additive noise with $w_t \sim \mathcal{N}(0, Q)$, then

$$p(x_{t+1} \mid x_t) \sim \mathcal{N}(x_{t+1}\,;\, f_t(x_t), Q), \tag{3.6}$$

where we use the notation $\mathcal{N}(x_{t+1}\,;\, f_t(x_t), Q)$ to explain that the random variable x_{t+1} is normal distributed with mean $f_t(x_t)$ and covariance Q.

The information given by the measurements about the state x_t can be modeled as

$$y_t = h_t(x_t, e_t), \tag{3.7}$$

where $h_t(\,\cdot\,)$ is a possibly nonlinear function and e_t is the measurement noise. The measurement model (3.7) provides information about the distribution $p(y_t \mid x_t)$. The combination of (3.5), (3.7) and a model of the prior $p(x_1)$ is referred to as a *state space model* [**?**] which is widely used in a large number of fields.

Figure 3.3: Left: clockwise rotation α of the vector x to the vector $x_\star$. Right: counterclockwise rotation α of the coordinate frame v to the coordinate frame u.

3.2 Parametrizing orientation

Rotating a vector in $\mathbb{R}^3$ changes the *direction* of the vector while retaining its *length*. The group of rotations in $\mathbb{R}^3$ is the special orthogonal group $SO(3)$. In this section we introduce four different ways of parametrizing orientations. Note that these describe the same quantity and can hence be used interchangeably. The different parametrizations can be converted to one another, see also Appendix A. There are differences in for instance the number of parameters used in the representation, the singularities and the uniqueness.

3.2.1 Rotation matrices

We encountered rotation matrices already in Chapter 2. Rotation matrices $R \in \mathbb{R}^{3\times 3}$ have the following properties

$$RR^\mathsf{T} = R^\mathsf{T}R = \mathcal{I}_3, \qquad \det R = 1. \tag{3.8}$$

The properties (3.8) provide an interpretation of the name special orthogonal group $SO(3)$. All orthogonal matrices of dimension 3×3 have the property $RR^\mathsf{T} = R^\mathsf{T}R = \mathcal{I}_3$ and are part of the orthogonal group $O(3)$. The notion *special* in $SO(3)$ specifies that only matrices with $\det R = 1$ are considered rotations.

Consider two coordinate frames denoted u and v. As was illustrated in Example 2.1, a vector x expressed in the v-frame can be rotated to the u-frame as

$$x^\mathrm{u} = R^\mathrm{uv}x^\mathrm{v}, \tag{3.9a}$$

and conversely we have

$$x^\mathrm{v} = (R^\mathrm{uv})^\mathsf{T}\, x^\mathrm{u} = R^\mathrm{vu}x^\mathrm{u}. \tag{3.9b}$$

A rotation matrix is a unique description of the orientation. It has 9 components which depend on each other as defined in (3.8).

3.2.2 Rotation vector

As described by Leonhard Euler in [**?**], a rotation around a point is always equivalent to a single rotation around some axis through this point, see [**?**] for a number of proofs. This is generally referred to as *Euler's rotation theorem*. Hence, it is possible to express the rotation between two coordinate frames in terms of an angle α and a unit vector n around which the rotation takes place. In this section, we will derive a relation between the representation α, n and the rotation matrix parametrization from the previous section. Instead of directly considering the rotation of a coordinate frame, we start by considering the rotation of a vector. Note that a counterclockwise rotation of the coordinate frame is equivalent to a clockwise rotation of a vector, see Example 3.2.

Example 3.2 (Rotation of a coordinate frame and rotation of a vector) *Consider the 2D example in Figure 3.3, where on the left, a vector x is rotated clockwise by an angle α to $x_\star$. This is equivalent to (on the right) rotating the coordinate frame v counterclockwise by an angle α. Note that $x^v_\star = x^u$.*

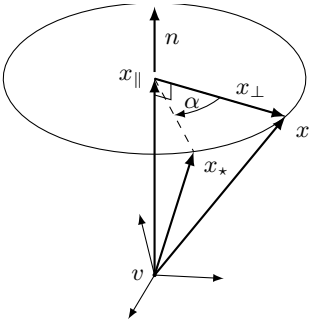


Figure 3.4: Clockwise rotation of a vector x by an angle α around the unit vector n. The rotated vector is denoted by $x_\star$. The vector x is decomposed in a component $x_\parallel$ that is parallel to the axis n, and a component $x_\perp$ that is orthogonal to it.

In Figure 3.4, a vector x is rotated an angle α around the unit vector n. We denote the rotated vector by $x_\star$. Suppose that x as expressed in the coordinate frame v is known (and denoted x^{v}) and that we want to express $x_\star^{\mathrm{v}}$ in terms of x^{v}, α and n. It can first be recognized that the vector x can be decomposed into a component parallel to the axis n, denoted $x_\parallel$, and a component orthogonal to it, denoted $x_\perp$, as

$$x^{\mathrm{v}} = x_\parallel^{\mathrm{v}} + x_\perp^{\mathrm{v}}. \tag{3.10a}$$

Based on geometric reasoning we can conclude that

$$x_\parallel^{\mathrm{v}} = (x^{\mathrm{v}} \cdot n^{\mathrm{v}})\, n^{\mathrm{v}}, \tag{3.10b}$$

where $\cdot$ denotes the inner product. Similarly, $x_\star^{\mathrm{v}}$ can be decomposed as

$$x_\star^{\mathrm{v}} = (x_\star^{\mathrm{v}})_\parallel + (x_\star^{\mathrm{v}})_\perp\,, \tag{3.11a}$$

where

$$(x_\star^{\mathrm{v}})_\parallel = x_\parallel^{\mathrm{v}}, \tag{3.11b}$$

$$(x_\star^{\mathrm{v}})_\perp = x_\perp^{\mathrm{v}} \cos\alpha + (x^{\mathrm{v}} \times n^{\mathrm{v}}) \sin\alpha. \tag{3.11c}$$

Hence, $x_\star^{\mathrm{v}}$ can be expressed in terms of x^{v} as

$$\begin{aligned} x_\star^{\mathrm{v}} &= (x^{\mathrm{v}} \cdot n^{\mathrm{v}})n^{\mathrm{v}} + (x^{\mathrm{v}} - (x^{\mathrm{v}} \cdot n^{\mathrm{v}})n^{\mathrm{v}}) \cos\alpha + (x^{\mathrm{v}} \times n^{\mathrm{v}}) \sin\alpha \\ &= x^{\mathrm{v}} \cos\alpha + n^{\mathrm{v}}(x^{\mathrm{v}} \cdot n^{\mathrm{v}})(1 - \cos\alpha) - (n^{\mathrm{v}} \times x^{\mathrm{v}}) \sin\alpha. \end{aligned} \tag{3.12}$$

Denoting the rotated coordinate frame the u-frame and using the equivalence between $x_\star^{\mathrm{v}}$ and x^{u} as shown in Example 3.2, this implies that

$$x^{\mathrm{u}} = x^{\mathrm{v}} \cos\alpha + n^{\mathrm{v}}(x^{\mathrm{v}} \cdot n^{\mathrm{v}})(1 - \cos\alpha) - (n^{\mathrm{v}} \times x^{\mathrm{v}}) \sin\alpha. \tag{3.13}$$

This equation is commonly referred to as the *rotation formula* or *Euler's formula* [**?**]. Note that the combination of n and α, or $\eta = n\alpha$, is denoted as the *rotation vector* or the *axis-angle parameterization*.

To show the equivalence between (3.13) and the rotation matrix parametrization, we will rewrite (3.13). Here, we make use of the fact that a cross product can be written as a matrix vector product. Given vectors u and v we have,

$$u \times v = [u\times]v = -[v\times]u, \qquad [u\times] \triangleq \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix}, \tag{3.14}$$

where u_1, u_2, u_3 denote the three components of the vector u. Furthermore, given vectors u, v and w, multiple cross products can be expanded in terms of the inner product as

$$u \times (v \times w) = v(w \cdot u) - w(u \cdot v). \tag{3.15}$$

Using these relations, (3.13) can be rewritten as

$$\begin{aligned} x^{\mathrm{u}} &= x^{\mathrm{v}}\cos\alpha + n^{\mathrm{v}}(x^{\mathrm{v}}\cdot n^{\mathrm{v}})(1-\cos\alpha) - (n^{\mathrm{v}}\times x^{\mathrm{v}})\sin\alpha \\ &= x^{\mathrm{v}}\cos\alpha + (n^{\mathrm{v}}\times(n^{\mathrm{v}}\times x^{\mathrm{v}}) + x^{\mathrm{v}})(1-\cos\alpha) - (n^{\mathrm{v}}\times x^{\mathrm{v}})\sin\alpha \\ &= \left(\mathcal{I}_3 - \sin\alpha[n^{\mathrm{v}}\times] + (1-\cos\alpha)[n^{\mathrm{v}}\times]^2\right)x^{\mathrm{v}}. \end{aligned} \tag{3.16}$$

Comparing (3.16) and (3.9a), it can be seen that a rotation matrix can be parametrized in terms of α, n as

$$R^{\mathrm{uv}}(n^{\mathrm{v}},\alpha) = \mathcal{I}_3 - \sin\alpha[n^{\mathrm{v}}\times] + (1-\cos\alpha)[n^{\mathrm{v}}\times]^2. \tag{3.17}$$

Note that equivalently, $R^{\mathrm{uv}}(n^{\mathrm{v}},\alpha)$ can also be written as

$$R^{\mathrm{uv}}(n^{\mathrm{v}},\alpha) = \exp\left(-\alpha[n^{\mathrm{v}}\times]\right), \tag{3.18}$$

since

$$\begin{aligned} \exp\left(-\alpha[n^{\mathrm{v}}\times]\right) &= \sum_{k=0}^{\infty} \tfrac{1}{k!}\left(-\alpha[n^{\mathrm{v}}\times]\right)^k \\ &= \mathcal{I}_3 - \alpha[n^{\mathrm{v}}\times] + \tfrac{1}{2!}\alpha^2[n^{\mathrm{v}}\times]^2 + \tfrac{1}{3!}\alpha^3[n^{\mathrm{v}}\times] - \tfrac{1}{4!}\alpha^4[n^{\mathrm{v}}\times]^2 - \dots \\ &= \mathcal{I}_3 - \left(\alpha - \tfrac{1}{3!}\alpha^3 + \dots\right)[n^{\mathrm{v}}\times] + \left(\tfrac{1}{2!}\alpha^2 - \tfrac{1}{4!}\alpha^4 + \dots\right)[n^{\mathrm{v}}\times]^2 \\ &= \mathcal{I}_3 - \sin\alpha[n^{\mathrm{v}}\times] + (1-\cos\alpha)[n^{\mathrm{v}}\times]^2. \end{aligned} \tag{3.19}$$

The rotation vector introduced in this section parametrizes the orientation in only three parameters. It is, however, not a unique parametrization since adding 2π to any angle α results in the same orientation. This is called *wrapping*. As shown in (3.17) and (3.18), the rotation matrix can straightforwardly be expressed in terms of the axis-angle representation.

3.2.3 Euler angles

Rotation can also be defined as a consecutive rotation around three axes in terms of so-called *Euler angles*. We use the convention (z, y, x) which first rotates an angle ψ around the z-axis, subsequently an angle θ around the y-axis and finally an angle ϕ around the x-axis. These angles are illustrated in Figure 3.5. Assuming that the v-frame is rotated by (ψ, θ, ϕ) with respect to the u-frame as illustrated in this figure, the rotation matrix R^{uv} is given by

$$\begin{aligned} R^{\mathrm{uv}} &= R^{\mathrm{uv}}(e_1,\phi)R^{\mathrm{uv}}(e_2,\theta)R^{\mathrm{uv}}(e_3,\psi) \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}\begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}\begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{pmatrix}, \end{aligned} \tag{3.20}$$

where we make use of the notation introduced in (3.17) and the following definition of the unit vectors

$$e_1 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^{\mathsf{T}}, \quad e_2 = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^{\mathsf{T}}, \quad e_3 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^{\mathsf{T}}. \tag{3.21}$$

The ψ, θ, ϕ angles are also often referred to as yaw (or heading), pitch and roll, respectively. Furthermore, roll and pitch together are often referred to as inclination.

Similar to the rotation vector, Euler angles parametrize orientation as a three-dimensional vector. Euler angle representations are not unique descriptions of a rotation for two reasons. First, due to wrapping of the Euler angles, the rotation $(0, 0, 0)$ is for instance equal to $(0, 0, 2\pi k)$ for any integer k. Furthermore, setting $\theta = \frac{\pi}{2}$ in (3.20), leads to

$$\begin{aligned} R^{\mathrm{uv}} &= \begin{pmatrix} 0 & 0 & -1 \\ \sin\phi\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi & 0 \\ \cos\phi\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\psi - \sin\phi\cos\psi & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & -1 \\ \sin(\phi-\psi) & \cos(\phi-\psi) & 0 \\ \cos(\phi-\psi) & -\sin(\phi-\psi) & 0 \end{pmatrix}. \end{aligned} \tag{3.22}$$

Hence, only the rotation $\phi - \psi$ can be observed. Because of this, for example the rotations $(\frac{\pi}{2}, \frac{\pi}{2}, 0)$, $(0, \frac{\pi}{2}, -\frac{\pi}{2})$, $(\pi, \frac{\pi}{2}, \frac{\pi}{2})$ are all three equivalent. This is called *gimbal lock* [**?**].

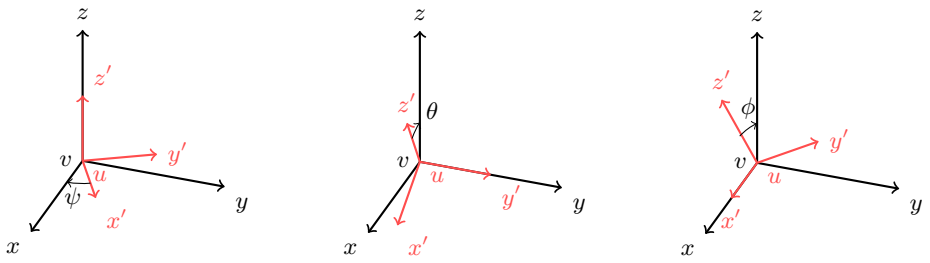


Figure 3.5: Definition of Euler angles as used in this work with left: rotation ψ around the z-axis, middle: rotation θ around the y-axis and right: rotation ϕ around the x-axis.

3.2.4 Unit quaternions

A commonly used parametrization of orientation is that of unit quaternions. Quaternions were first introduced by [**?**] and are widely used in orientation estimation algorithms, see *e.g.* [**?** **?**]. A unit quaternion use a 4-dimensional representation of the orientation according to

$$q = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix}^{\mathsf{T}} = \begin{pmatrix} q_0 \\ q_v \end{pmatrix}, \qquad q \in \mathbb{R}^4, \qquad \|q\|_2 = 1. \tag{3.23}$$

A unit quaternion is not a unique description of an orientation. The reason for this is that if q represents a certain orientation, then $-q$ describes the same orientation.

A rotation can be defined using unit quaternions as

$$\bar{x}^{\mathrm{u}} = q^{\mathrm{uv}} \odot \bar{x}^{\mathrm{v}} \odot (q^{\mathrm{uv}})^{\mathsf{c}}, \tag{3.24}$$

where $(q^{\mathrm{uv}})^{\mathsf{c}} = q^{\mathrm{vu}}$ denotes the quaternion conjugate, defined as

$$q^{\mathsf{c}} = \begin{pmatrix} q_0 & -q_v^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}, \tag{3.25}$$

and $\bar{x}^{\mathrm{v}}$ denotes the quaternion representation of x^{v} as

$$\bar{x}^{\mathrm{v}} = \begin{pmatrix} 0 & (x^{\mathrm{v}})^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}. \tag{3.26}$$

Note that (3.26) is typically not a unit quaternion. The notation $\odot$ denotes the quaternion multiplication given by

$$p \odot q = \begin{pmatrix} p_0 q_0 - p_v \cdot q_v \\ p_0 q_v + q_0 p_v + p_v \times q_v \end{pmatrix} = p^{\mathsf{L}} q = q^{\mathsf{R}} p, \tag{3.27}$$

where

$$p^{\mathsf{L}} \triangleq \begin{pmatrix} p_0 & -p_v^{\mathsf{T}} \\ p_v & p_0 \mathcal{I}_3 + [p_v \times] \end{pmatrix}, \qquad q^{\mathsf{R}} \triangleq \begin{pmatrix} q_0 & -q_v^{\mathsf{T}} \\ q_v & q_0 \mathcal{I}_3 - [q_v \times] \end{pmatrix}. \tag{3.28}$$

Using (3.25)–(3.28), (3.24) can be written as

$$\begin{aligned} \bar{x}^{\mathrm{u}} &= (q^{\mathrm{uv}})^{\mathsf{L}} (q^{\mathrm{vu}})^{\mathsf{R}} \bar{x}^{\mathrm{v}} \\ &= \begin{pmatrix} q_0 & -q_v^{\mathsf{T}} \\ q_v & q_0 \mathcal{I}_3 + [q_v \times] \end{pmatrix} \begin{pmatrix} q_0 & q_v^{\mathsf{T}} \\ -q_v & q_0 \mathcal{I}_3 + [q_v \times] \end{pmatrix} \begin{pmatrix} 0 \\ x^{\mathrm{v}} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0_{1\times 3} \\ 0_{3\times 1} & q_v q_v^{\mathsf{T}} + q_0^2 \mathcal{I}_3 + 2 q_0 [q_v \times] + [q_v \times]^2 \end{pmatrix} \begin{pmatrix} 0 \\ x^{\mathrm{v}} \end{pmatrix}. \end{aligned} \tag{3.29}$$

Comparing (3.29) to (3.17), it can be recognized that if we choose

$$q^{\mathrm{uv}}(n^{\mathrm{v}}, \alpha) = \begin{pmatrix} \cos \frac{\alpha}{2} \\ -n^{\mathrm{v}} \sin \frac{\alpha}{2} \end{pmatrix}, \tag{3.30}$$

the two rotation formulations are equivalent since

$$\begin{aligned}\bar{x}^{\mathrm{u}} &= \begin{pmatrix} 1 & 0_{1\times 3} \\ 0_{3\times 1} & \mathcal{I}_3 - 2\cos\frac{\alpha}{2}\sin\frac{\alpha}{2}[n^{\mathrm{v}}\times] + 2\sin^2\frac{\alpha}{2}[n^{\mathrm{v}}\times]^2 \end{pmatrix}\begin{pmatrix} 0 \\ x^{\mathrm{v}} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0_{1\times 3} \\ 0_{3\times 1} & \mathcal{I}_3 - \sin\alpha[n^{\mathrm{v}}\times] + (1-\cos\alpha)\,[n^{\mathrm{v}}\times]^2 \end{pmatrix}\begin{pmatrix} 0 \\ x^{\mathrm{v}} \end{pmatrix}. \end{aligned} \tag{3.31}$$

Here, we made use of standard trigonometric relations and the fact that since $\|n^{\mathrm{v}}\|_2 = 1$, $n^{\mathrm{v}}\,(n^{\mathrm{v}})^{\mathsf{T}} = \mathcal{I}_3 + [n^{\mathrm{v}}\times]^2$. Hence, it can be concluded that q^{uv} can be expressed in terms of α and n^{v} as in (3.30).

Equivalently, $q^{\mathrm{uv}}(n^{\mathrm{v}}, \alpha)$ can also be written as

$$q^{\mathrm{uv}}(n^{\mathrm{v}}, \alpha) = \exp(-\tfrac{\alpha}{2}\bar{n}^{\mathrm{v}}) = \sum_{k=0}^{\infty} \tfrac{1}{k!}\left(-\tfrac{\alpha}{2}\bar{n}^{\mathrm{v}}\right)^k, \tag{3.32}$$

where

$$(\bar{n}^{\mathrm{v}})^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^{\mathsf{T}}, \tag{3.33a}$$

$$(\bar{n}^{\mathrm{v}})^1 = \begin{pmatrix} 0 & (n^{\mathrm{v}})^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}, \tag{3.33b}$$

$$(\bar{n}^{\mathrm{v}})^2 = \bar{n}^{\mathrm{v}} \odot \bar{n}^{\mathrm{v}} = \begin{pmatrix} -\|n^{\mathrm{v}}\|_2^2 & 0_{3\times 1} \end{pmatrix}^{\mathsf{T}} = \begin{pmatrix} -1 & 0_{3\times 1} \end{pmatrix}^{\mathsf{T}}, \tag{3.33c}$$

$$(\bar{n}^{\mathrm{v}})^3 = \begin{pmatrix} 0 & -(n^{\mathrm{v}})^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}, \tag{3.33d}$$

This leads to

$$\begin{aligned} q^{\mathrm{uv}}(n^{\mathrm{v}}, \alpha) = \exp(-\tfrac{\alpha}{2}\bar{n}^{\mathrm{v}}) &= \sum_{k=0}^{\infty} \tfrac{1}{k!}\left(-\tfrac{\alpha}{2}\bar{n}^{\mathrm{v}}\right)^k \\ &= \begin{pmatrix} 1 - \frac{1}{2!}\frac{\alpha^2}{4} + \frac{1}{4!}\frac{\alpha^4}{16} - \dots \\ -\frac{\alpha}{2}n^{\mathrm{v}} + \frac{1}{3!}\frac{\alpha^3}{8}n^{\mathrm{v}} - \frac{1}{5!}\frac{\alpha^5}{32}n^{\mathrm{v}} + \dots \end{pmatrix} \\ &= \begin{pmatrix} \cos\frac{\alpha}{2} \\ -n^{\mathrm{v}}\sin\frac{\alpha}{2} \end{pmatrix}. \end{aligned} \tag{3.34}$$

Note the similarity to (3.18) and (3.19). The reason why both rotation matrices and unit quaternions can be described in terms of an exponential of a rotation vector will be discussed in §3.3.1.

3.3 Probabilistic orientation modeling

The four parametrizations of orientation discussed in §3.2 can be used interchangeably. However, the choice of which parametrization to use as states x_t in the filtering and smoothing problems introduced in §3.1 has significant impact on the workings of the algorithm. An important reason for this is that estimation algorithms typically assume that the unknown states and parameters are represented in Euclidean space. For instance, they assume that the subtraction of two orientations gives information about the difference in orientation and that the addition of two orientations is again a valid orientation. For the four parametrizations discussed in §3.2, this is generally not true. For instance, due to wrapping and gimbal lock, subtraction of Euler angles and rotation vectors can result in large numbers even in cases when the rotations are similar. Also, addition and subtraction of unit quaternions and rotation matrices do not in general result in a valid rotation. The equality constraints on the norm of unit quaternions and on the determinant and the orthogonality of rotation matrices are typically hard to include in the estimation algorithms. In §3.3.1, we will discuss a method to represent orientation in estimation algorithms that deals with the issues described above. It is frequently used in the algorithms that will be described in Chapter 4. In §3.3.2, we will also discuss some alternative methods to parametrize orientation for estimation purposes.

3.3.1 Linearization

As mentioned in §3.2, the group of rotations in three dimensions is the special orthogonal group $SO(3)$. More specifically, $SO(3)$ is a so-called *matrix Lie group*. For a discussion on the properties of matrix

Lie groups and on the reasons why $SO(3)$ is indeed such a group we refer the reader to *e.g.* [**?**]. Since rotations are a matrix Lie group, there exists an *exponential map* from a corresponding Lie algebra. Using this property, it is possible to represent orientations on $SO(3)$ using unit quaternions or rotation matrices, while orientation deviations are represented using rotation vectors on $\mathbb{R}^3$, see *e.g.* [**?**]. Hence, we encode an orientation q_t^{nb} in terms of a *linearization point* parametrized either as a unit quaternion $\tilde{q}_t^{\text{nb}}$ or as a rotation matrix $\tilde{R}_t^{\text{nb}}$ and an *orientation deviation* using a rotation vector η_t. Assuming that the orientation deviation is expressed in the body frame n,[1]

$$q_t^{\text{nb}} = \exp\left(\frac{\bar{\eta}_t^{\text{n}}}{2}\right) \odot \tilde{q}_t^{\text{nb}}, \qquad R_t^{\text{nb}} = \exp\left([\eta_t^{\text{n}}\times]\right) \tilde{R}_t^{\text{nb}}, \tag{3.35}$$

where analogously to (3.34) and (3.19),

$$\exp(\bar{\eta}) = \begin{pmatrix} \cos\|\eta\|_2 \\ \frac{\eta}{\|\eta\|_2}\sin\|\eta\|_2 \end{pmatrix}, \tag{3.36a}$$

$$\exp([\eta\times]) = \mathcal{I}_3 + \sin\left(\|\eta\|_2\right)\left[\frac{\eta}{\|\eta\|_2}\times\right] + \left(1-\cos\left(\|\eta\|_2\right)\right)\left[\frac{\eta}{\|\eta\|_2}\times\right]^2. \tag{3.36b}$$

For notational convenience, in the remainder we will use the mappings

$$q = \exp_{\mathsf{q}}(\eta), \qquad \exp_{\mathsf{q}} : \mathbb{R}^3 \to \{q \in \mathbb{R}^4 : \|q\|_2 = 1\}, \tag{3.37a}$$

$$R = \exp_{\mathsf{R}}(\eta), \quad \exp_{\mathsf{R}} : \mathbb{R}^3 \to \{R \in \mathbb{R}^{3\times 3} : RR^{\mathsf{T}} = \mathcal{I}_3,\ \det R = 1\}, \tag{3.37b}$$

which allow us to rewrite (3.35) as

$$q_t^{\text{nb}} = \exp_{\mathsf{q}}\left(\frac{\eta_t^{\text{n}}}{2}\right) \odot \tilde{q}_t^{\text{nb}}, \qquad R_t^{\text{nb}} = \exp_{\mathsf{R}}\left(\eta_t^{\text{n}}\right) \tilde{R}_t^{\text{nb}}. \tag{3.38}$$

The reverse mappings are defined as

$$\eta = \log_{\mathsf{q}}(q) = \frac{\arccos q_0}{\sin\arccos q_0} q_v = \frac{\arccos q_0}{\|q_v\|_2} q_v, \quad \log_{\mathsf{q}} : \{q \in \mathbb{R}^4 : \|q\|_2 = 1\} \to \mathbb{R}^3, \tag{3.39a}$$

$$\eta = \log_{\mathsf{R}}(R) = \begin{pmatrix} (\log R)_{32} \\ (\log R)_{13} \\ (\log R)_{21} \end{pmatrix}, \quad \log_{\mathsf{R}} : \{R \in \mathbb{R}^{3\times 3} : RR^{\mathsf{T}} = \mathcal{I}_3,\ \det R = 1\} \to \mathbb{R}^3, \tag{3.39b}$$

where $\log R$ is the standard matrix logarithm. Since we typically assume that η_t^{n} is small, we will frequently make use of the following approximations

$$\exp_{\mathsf{q}}(\eta) \approx \begin{pmatrix} 1 \\ \eta \end{pmatrix}, \qquad \log_{\mathsf{q}}(q) \approx q_v, \tag{3.40a}$$

$$\exp_{\mathsf{R}}(\eta) \approx \mathcal{I}_3 + [\eta\times], \qquad \log_{\mathsf{R}}(R) \approx \begin{pmatrix} R_{32} & R_{13} & R_{21} \end{pmatrix}^{\mathsf{T}}. \tag{3.40b}$$

The idea briefly outlined in this section is closely related to approaches used to estimate orientation in robotics, see *e.g.* [**? ? ? ? ?**]. It is also related to the so-called multiplicative extended Kalman filter (MEKF) frequently used in aeronautics, see *e.g.* [**? ?**].

3.3.2 Alternative methods

An alternative method to estimate orientation assumes that the states representing the orientation lie on a manifold. This can be done by modeling the orientation and its uncertainty using a *spherical distribution* which naturally restricts the orientation estimates and their uncertainties to be in $SO(3)$. In recent years, a number of approaches have been proposed to estimate the orientation using these kinds

[1] A similar derivation can be done by assuming an orientation deviation in the navigation frame b.

of distributions. For instance, in [**?** **?** **?**] algorithms are presented to estimate orientation by modeling it using a Bingham distribution.

The difficulties caused by directly using one of the four orientation parametrizations introduced in §3.2 in orientation estimation algorithms is widely recognized. Nevertheless, a large number of approaches directly uses these parametrizations in estimation algorithms. For instance, it is common practice to use unit quaternions in estimation algorithms and to normalize the resulting quaternions each time they loose their normalization, see *e.g.* [**?** **?** **?**]. Different approaches to handle the normalization of the quaternions in these algorithms are discussed in [**?**].

3.4 Measurement models

In the past two sections, we have focused on how orientations can be parametrized. In this section, we will go back to the probabilistic models for the position and orientation estimation problems introduced in §3.1 and provide different measurement models $p(y_t \mid x_t, \theta)$.

3.4.1 Gyroscope measurement models

As discussed in §2.2, the gyroscope measures the angular velocity $\omega_{\text{ib}}^{\text{b}}$ at each time instance t. However, as shown in §2.4, its measurements are corrupted by a slowly time-varying bias $\delta_{\omega,t}$ and noise $e_{\omega,t}$. Hence, the gyroscope measurement model is given by

$$y_{\omega,t} = \omega_{\text{ib},t}^{\text{b}} + \delta_{\omega,t}^{\text{b}} + e_{\omega,t}^{\text{b}}. \tag{3.41}$$

As was shown in Figure 2.3, the gyroscope measurement noise is quite Gaussian. Because of this, it is typically assumed that $e_{\omega,t}^{\text{b}} \sim \mathcal{N}(0, \Sigma_\omega)$. If the sensor is properly calibrated, the measurements in the three gyroscope axes are independent. In that case, it can be assumed that

$$\Sigma_\omega = \begin{pmatrix} \sigma_{\omega,x}^2 & 0 & 0 \\ 0 & \sigma_{\omega,y}^2 & 0 \\ 0 & 0 & \sigma_{\omega,z}^2 \end{pmatrix}. \tag{3.42}$$

The gyroscope bias $\delta_{\omega,t}^{\text{b}}$ is slowly time-varying, as discussed in §2.4. There are two conceptually different ways to treat this slowly time-varying bias. One is to treat the bias as a constant parameter, assuming that it typically changes over a longer time period than the time of the experiment. The bias can then either be pre-calibrated in a separate experiment, or it can be considered to be part of the unknown parameters θ as introduced in §3.1. Alternatively, it can be assumed to be slowly time-varying. This can be justified either by longer experiment times or by shorter bias stability. In the latter case, $\delta_{\omega,t}^{\text{b}}$ can instead be considered as part of the state vector x_t and can for instance be modeled as a random walk

$$\delta_{\omega,t+1}^{\text{b}} = \delta_{\omega,t}^{\text{b}} + e_{\delta_\omega,t}^{\text{b}}, \tag{3.43}$$

where $e_{\delta_\omega,t}^{\text{b}} \sim \mathcal{N}(0, \Sigma_{\delta_\omega,t})$ represents how constant the gyroscope bias actually is.

Modeling the sensor noise and bias is related to the sensor properties. However, there are also modeling choices related to the experiments that can be made. As described in §2.2, the angular velocity $\omega_{\text{ib}}^{\text{b}}$ can be expressed as

$$\omega_{\text{ib},t}^{\text{b}} = R_t^{\text{bn}} \left(\omega_{\text{ie},t}^{\text{n}} + \omega_{\text{en},t}^{\text{n}} \right) + \omega_{\text{nb},t}^{\text{b}}. \tag{3.44}$$

If the sensor does not travel over significant distances as compared to the size of the earth — which is often the case for the applications discussed in Chapter 1 — the navigation frame n can safely be assumed to be stationary. In that case, the transport rate $\omega_{\text{en},t}^{\text{n}}$ is zero. Although the earth rotation ω_{ie} as expressed in the body frame b is not constant, its magnitude as compared to the magnitude of the actual measurements is fairly small (see §2.2 and the experimental data presented in Example 2.3). Assuming that the earth rotation is negligible and the navigation frame is stationary leads to the following simplified measurement model

$$y_{\omega,t} = \omega_{\text{nb},t}^{\text{b}} + \delta_{\omega,t}^{\text{b}} + e_{\omega,t}^{\text{b}}. \tag{3.45}$$

3.4.2 Accelerometer measurement models

The accelerometer measures the specific force f_t^{b} at each time instance t, see also §2.3. As shown in §2.4, the accelerometer measurements are typically assumed to be corrupted by a bias $\delta_{\mathrm{a},t}$ and noise $e_{\mathrm{a},t}$ as

$$y_{\mathrm{a},t} = f_t^{\mathrm{b}} + \delta_{\mathrm{a},t}^{\mathrm{b}} + e_{\mathrm{a},t}^{\mathrm{b}}. \tag{3.46}$$

The accelerometer noise is typically quite Gaussian as was illustrated in Figure 2.4 and can hence be modeled as $e_{\mathrm{a},t}^{\mathrm{b}} \sim \mathcal{N}(0, \Sigma_{\mathrm{a}})$. For a properly calibrated sensor, the covariance matrix Σ_{a} can often be assumed to be diagonal.

The accelerometer bias $\delta_{\mathrm{a},t}^{\mathrm{b}}$ is slowly time-varying. Similar to the gyroscope bias, the accelerometer bias can either be modeled as a constant parameter, or as part of the time-varying state, for instance using a random walk model as in (3.43).

As introduced in §2.3, the specific force measured by the accelerometer is given by

$$f^{\mathrm{b}} = R^{\mathrm{bn}}(a_{\mathrm{ii}}^{\mathrm{n}} - g^{\mathrm{n}}). \tag{3.47}$$

Assuming that the navigation frame is fixed to the earth frame, we derived a relation for $a_{\mathrm{ii}}^{\mathrm{n}}$ as

$$a_{\mathrm{ii}}^{\mathrm{n}} = a_{\mathrm{nn}}^{\mathrm{n}} + 2\omega_{\mathrm{ie}}^{\mathrm{n}} \times v_{\mathrm{n}}^{\mathrm{n}} + \omega_{\mathrm{ie}}^{\mathrm{n}} \times \omega_{\mathrm{ie}}^{\mathrm{n}} \times p^{\mathrm{n}}. \tag{3.48}$$

The centrifugal acceleration $\omega_{\mathrm{ie}}^{\mathrm{n}} \times \omega_{\mathrm{ie}}^{\mathrm{n}} \times p^{\mathrm{n}}$ is typically absorbed in the local gravity vector. The magnitude of the Coriolis acceleration is small compared to the magnitude of the accelerometer measurements (see Example 2.2 and the experimental data presented in Example 2.3). Neglecting this term leads to the following simplified measurement model

$$y_{\mathrm{a},t} = R_t^{\mathrm{bn}}(a_{\mathrm{nn}}^{\mathrm{n}} - g^{\mathrm{n}}) + \delta_{\mathrm{a},t}^{\mathrm{b}} + e_{\mathrm{a},t}^{\mathrm{b}}. \tag{3.49}$$

Since the accelerometer measures both the local gravity vector and the linear acceleration of the sensor, it provides information both about the change in position and about the inclination of the sensor. For orientation estimation, only the information about the inclination is of concern. Hence, a model for the linear acceleration needs to be made to express the relation between the inclination and the measurements. To model this, it can be recognized that in practice, most accelerometer measurements are dominated by the gravity vector, as illustrated in Example 3.3.

Example 3.3 (Magnitude of a sensor's linear acceleration) *Let us consider a 1D example where a sensor has an initial velocity* $v_1 = 0$ m/s *and accelerates with* $a_{nn}^{n} = 9.82$ m/s^2. *After* 4.51 *seconds, the sensor will have traveled* 100 *meters. This is about twice as fast as the world record currently held by Usain Bolt. In fact, humans can reach fairly high accelerations but can only accelerate for a short time. Naturally, cars can accelerate to higher velocities than humans. The sensor in this example has reached a final velocity of* 160 km/h. *Even in the case of a car it is therefore unlikely that it can have an acceleration this high for a long period of time.*

Since the accelerometer measurements are typically dominated by the gravity vector, a commonly used model assumes the linear acceleration to be approximately zero

$$y_{\mathrm{a},t} = -R_t^{\mathrm{bn}} g^{\mathrm{n}} + \delta_{\mathrm{a},t}^{\mathrm{b}} + e_{\mathrm{a},t}^{\mathrm{b}}. \tag{3.50}$$

Naturally, the model (3.50) is almost never completely true. However, it can often be used as a sufficiently good approximation of reality. Note that the noise term $e_{\mathrm{a},t}^{\mathrm{b}}$ in this case does not only represent the measurement noise, but also the model uncertainty. The model (3.50) can for instance be used in combination with *outlier rejection* where measurements that clearly violate the assumption that the linear acceleration is zero are disregarded. It is also possible to adapt the noise covariance matrix Σ_{a}, depending on the sensor's acceleration [**?** **?**]. Furthermore, it is possible to instead model the acceleration based on physical reasoning [**?**].

3.4.3 Modeling additional information

In this section we will discuss models for the measurements we use to complement the inertial sensor measurements. For orientation estimation we use magnetometers, while for pose estimation we use position measurements.

Magnetometer models Magnetometers measure the local magnetic field, consisting of both the earth magnetic field and the magnetic field due to the presence of magnetic material. The (local) earth magnetic field is denoted m^{n} and it is illustrated in Figure 3.6. Its horizontal component points towards the earth's magnetic north pole. The ratio between the horizontal and vertical component depends on the location on the earth and can be expressed in terms of the so-called dip angle δ. The dip angle and the magnitude of the earth magnetic field are accurately known from geophysical studies, see *e.g.* [**?**].

Assuming that the sensor does not travel over significant distances as compared to the size of the earth, the local earth magnetic field can be modeled as being constant. In case no magnetic material is present in the vicinity of the sensor, orientation information can be deduced from the magnetometer. More specifically, magnetometers are typically used to complement accelerometers to provide information about the sensor heading, *i.e.* about the orientation around the gravity vector which can not be determined from the accelerometer measurements. Magnetometers provide information about the heading in all locations on the earth except on the magnetic poles, where the local magnetic field m^{n} is vertical. Orientation can be estimated based on the *direction* of the magnetic field. The *magnitude* of the field is irrelevant. Because of this, without loss of generality we model the earth magnetic field as

$$m^{\mathrm{n}} = \begin{pmatrix} \cos\delta & 0 & \sin\delta \end{pmatrix}^{\mathsf{T}}, \tag{3.51}$$

i.e. we assume that $\|m^{\mathrm{n}}\|_2 = 1$. Assuming that the magnetometer only measures the local magnetic field, its measurements $y_{\mathrm{m},t}$ can be modeled as

$$y_{\mathrm{m},t} = R_t^{\mathrm{bn}} m^{\mathrm{n}} + e_{\mathrm{m},t}, \tag{3.52}$$

where $e_{\mathrm{m},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{m}})$. The noise $e_{\mathrm{m},t}$ represents the magnetometer measurement noise as well as the model uncertainty. Note that using the models (3.51) and (3.52), we define the heading with respect to the magnetic north, which is sufficient for most applications. In case we would be interested in navigation with respect to the true north instead, the magnetic declination needs to be taken into account. Since the magnetic declination depends on the location on the earth it is necessary to know the location of the sensor to correct for this.

In practice, the actual magnetic field can differ significantly from the earth magnetic field. In indoor environments, for instance, presence of magnetic material in the structure of buildings and in furniture influences the magnetic field that is measured by the magnetometer. Furthermore, the magnetic field is affected in applications where the magnetometer is mounted in *e.g.* a vehicle, train or on a robot. In case the magnetic material is rigidly attached to the sensor, the magnetometer can be calibrated for its presence [**?** **?** **?** **?**]. The presence of magnetic material in the vicinity of the sensor that can not be calibrated for is of major concern for practical applications. Because of this, there is a vast amount of literature on the topic, see *e.g.* [**?** **?** **?**].

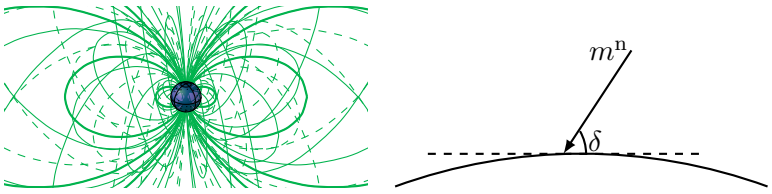
Note that when using magnetometers for orientation estimation, the presence of magnetic material is typically considered to be an undesired disturbance. However, the presence of magnetic material can also be considered to be a property which can be exploited. This is done in approaches which use the magnetic field as a source of position information [**?** **?** **?**].

Position information Position information can be obtained from for instance GNSS or UWB measurements. In this tutorial, we will consider a very basic measurement model where the sensors directly measure the position as

$$y_{\mathrm{p},t} = p_t^{\mathrm{n}} + e_{\mathrm{p},t}, \tag{3.53}$$

with $e_{\mathrm{p},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{p}})$.

Many sensors do not measure the position directly. Their measurements can, however, be pre-processed to obtain position estimates and their corresponding covariances [**?**]. For example, time of arrival measurements can be pre-processed using multilateration techniques. Measurements of this type often contain a significant amount of outliers. The reason is that the signals can be delayed due to multipath or non-line-of-sight (NLOS) conditions. Possible solutions deal with this by doing outlier rejection, by using robust algorithms, see *e.g.* [**?**], or by modeling the noise distribution as a non-Gaussian distribution, see *e.g.* [**?** **?** **?**]. We will discuss an example of this in more detail in §6.1.



(a) (b)

Figure 3.6: (a) Schematic of the earth magnetic field lines (green) around the earth (blue).[3] (b) Schematic of a part of the earth where the local earth magnetic field m^{n} makes an angle δ with the horizontal plane. This angle is called the dip angle.

3.5 Choosing the state and modeling its dynamics

The fundamental continuous-time relations that form the basis of our dynamic models are the fact that the position p^{n}, velocity $v^{\mathrm{n}}_{\mathrm{n}}$ and acceleration $a^{\mathrm{n}}_{\mathrm{nn}}$ are related as

$$v^{\mathrm{n}}_{\mathrm{n}} = \frac{\mathrm{d}p^{\mathrm{n}}}{\mathrm{d}t}\Big|_{\mathrm{n}}, \qquad a^{\mathrm{n}}_{\mathrm{nn}} = \frac{\mathrm{d}v^{\mathrm{n}}}{\mathrm{d}t}\Big|_{\mathrm{n}}, \tag{3.54}$$

and that the orientation and angular velocity $\omega^{\mathrm{b}}_{\mathrm{nb},t}$ are related as

$$\frac{\mathrm{d}q^{\mathrm{nb}}}{\mathrm{d}t} = q^{\mathrm{nb}} \odot \tfrac{1}{2}\bar{\omega}^{\mathrm{b}}_{\mathrm{nb}}, \qquad \frac{\mathrm{d}R^{\mathrm{nb}}}{\mathrm{d}t} = R^{\mathrm{nb}}[\omega^{\mathrm{b}}_{\mathrm{nb}}\times], \tag{3.55}$$

depending on orientation parametrization. For a derivation of (3.55), see *e.g.* [**?**]. Using an Euler discretization of (3.54) assuming that the acceleration is constant between samples, the dynamics of the position and velocity can be expressed in terms of the acceleration as

$$p^{\mathrm{n}}_{t+1} = p^{\mathrm{n}}_t + Tv^{\mathrm{n}}_{\mathrm{n},t} + \tfrac{T^2}{2}a^{\mathrm{n}}_{\mathrm{nn},t}, \tag{3.56a}$$
$$v^{\mathrm{n}}_{\mathrm{n},t+1} = v^{\mathrm{n}}_{\mathrm{n},t} + Ta^{\mathrm{n}}_{\mathrm{nn},t}, \tag{3.56b}$$

where T is the time between two samples. Similarly, the dynamics of the orientation can be expressed in terms of unit quaternions or rotation matrices as

$$q^{\mathrm{nb}}_{t+1} = q^{\mathrm{nb}}_t \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}\omega^{\mathrm{b}}_{\mathrm{nb},t}\right), \tag{3.57a}$$
$$R^{\mathrm{nb}}_{t+1} = R^{\mathrm{nb}}_t \exp_{\mathsf{R}}\left(T\omega^{\mathrm{b}}_{\mathrm{nb},t}\right). \tag{3.57b}$$

Dynamic models describe how the state changes over time. For the problem of position and orientation estimation using inertial sensors, there are two commonly used modeling alternatives for the dynamics [**?**]. In the first, the state vector x_t is chosen to consists of

$$x_t = \begin{pmatrix} (p^{\mathrm{n}}_t)^{\mathsf{T}} & (v^{\mathrm{n}}_{\mathrm{n},t})^{\mathsf{T}} & (a^{\mathrm{n}}_{\mathrm{nn,t}})^{\mathsf{T}} & (q^{\mathrm{nb}}_t)^{\mathsf{T}} & \left(\omega^{\mathrm{b}}_{\mathrm{nb},t}\right)^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}. \tag{3.58}$$

The change in position, velocity and orientation states can then be described in terms of the velocity, acceleration and angular velocity states, respectively. The dynamics of the acceleration and the angular velocity can be described in terms of a *motion model*. Examples of motion models that can be used are a constant acceleration model, which assumes that the dynamics of the acceleration can be described as

$$a^{\mathrm{n}}_{\mathrm{nn},t+1} = a^{\mathrm{n}}_{\mathrm{nn},t} + w_{\mathrm{a},t}, \tag{3.59}$$

with $w_{\mathrm{a},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{w,a}})$, and a constant angular velocity model, which describes the dynamics of the angular velocity as

$$\omega^{\mathrm{b}}_{\mathrm{nb},t+1} = \omega^{\mathrm{b}}_{\mathrm{nb},t} + w_{\omega,t}, \tag{3.60}$$

with $w_{\omega,t} \sim \mathcal{N}(0, \Sigma_{\mathrm{w},\omega})$. The process noise terms $w_{\mathrm{a},t}$ and $w_{\omega,t}$ model the assumptions on how constant the acceleration and angular velocity actually are.

[3] Adapted version of 'Dipolar magnetic field' by Cyril Langlois available at `http://texample.net` under CC BY 2.5 (`http://creativecommons.org/licenses/by/2.5`).

Alternatively, the state vector x_t can be chosen as

$$x_t = \begin{pmatrix} (p_t^{\mathrm{n}})^{\mathsf{T}} & (v_{\mathrm{n},t}^{\mathrm{n}})^{\mathsf{T}} & (q_t^{\mathrm{nb}})^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}. \tag{3.61}$$

To describe the dynamics of the states, the inertial measurements can then be used as an *input* to the dynamic equation (3.5). Hence, the change in position, velocity and orientation is directly modeled in terms of the inertial measurements. In this case, expressions for $a_{\mathrm{nn},t}^{\mathrm{n}}$ and $\omega_{\mathrm{nb},t}^{\mathrm{b}}$ in (3.56) and (3.57) are obtained from the accelerometer measurement model and the gyroscope measurement model, see §3.4. The process noise can explicitly be modeled in terms of the accelerometer measurement noise $e_{\mathrm{a},t}$ and the gyroscope measurement noise $e_{\omega,t}$.

The benefit of using a motion model for the state dynamics is that knowledge about the motion of the sensor can be included in this model. However, it comes at the expense of having a larger state vector. The benefit of using the inertial measurements as an input to the dynamics is that the process noise has the intuitive interpretation of representing the inertial measurement noise. Hence, the latter approach is often used for applications where it is difficult to obtain sensible motion models. Another difference between the two approaches is that changes in the acceleration and angular velocity will have a slightly faster effect on the state when the inertial measurements are used as an input to the dynamics. The reason for this is that the constant acceleration and angular velocity models delay the effects of changes in the dynamics. Because of this, their estimates typically look slightly more smooth.

3.6 Models for the prior

Looking back at §3.1, to solve the smoothing and filtering problems (3.3a) and (3.3b), we have now discussed different models for the dynamics $p(x_t \mid x_{t-1}, \theta)$ in §3.5 and for the measurements $p(y_t \mid x_t, \theta)$ in §3.4. The remaining distributions to be defined are $p(x_1 \mid \theta)$ and $p(\theta)$. Note that we typically assume that x_1 is independent of θ. Hence, in this section we focus on the prior distributions $p(x_1)$ and $p(\theta)$.

In many cases, there is fairly little prior information available about the parameters θ. It is, however, often possible to indicate reasonable values for the parameters. For example, it is reasonable to assume that the gyroscope bias is fairly small but can be both positive and negative. For instance, for the data presented in Example 2.3, the average bias over the 55-minute data set was $\begin{pmatrix} 35.67 & 54.13 & -1.07 \end{pmatrix}^{\mathsf{T}} \cdot 10^{-4}$ rad/s. If we would assume that in 68% of the cases, the bias is within the bounds $-\sigma_{\delta_\omega} \leq \delta_{\omega,t}^{\mathrm{b}} \leq \sigma_{\delta_\omega}$ with $\sigma_{\delta_\omega} = 5 \cdot 10^{-3}$, a reasonable prior would be

$$\delta_{\omega,t}^{\mathrm{b}} \sim \mathcal{N}(0, \sigma_{\delta_\omega}^2 \mathcal{I}_3). \tag{3.62}$$

For the prior $p(x_1)$, it is typically possible to get a reasonable estimate from data. For the position and velocity, this can be modeled as

$$p_1^{\mathrm{n}} = \breve{p}_1^{\mathrm{n}} + e_{\mathrm{p,i}}, \qquad e_{\mathrm{p,i}} \sim \mathcal{N}(0, \sigma_{\mathrm{p,i}}^2 \mathcal{I}_3), \tag{3.63a}$$

$$v_1^{\mathrm{n}} = \breve{v}_1^{\mathrm{n}} + e_{\mathrm{v,i}}, \qquad e_{\mathrm{v,i}} \sim \mathcal{N}(0, \sigma_{\mathrm{v,i}}^2 \mathcal{I}_3). \tag{3.63b}$$

Here, the estimate $\breve{p}_1^{\mathrm{n}}$ can for instance be determined based on the first position measurement. In that case, the uncertainty $\sigma_{\mathrm{p,i}}$ can also be chosen equal to the uncertainty of the position measurements. In case no additional information is available, the estimates $\breve{p}_1^{\mathrm{n}}$ and $\breve{v}_1^{\mathrm{n}}$ can be set to zero with an appropriate standard deviation instead.

A commonly used method to determine the initial orientation is to use the first accelerometer and magnetometer samples. This method is based on the fact that given two (or more) linearly independent vectors in two coordinate frames, the rotation between the two coordinate frames can be determined. The implicit assumption is that the accelerometer only measures the gravity vector and the magnetometer only measures the local magnetic field. Hence, the four vectors are given by the measurements $y_{\mathrm{a},t}$ and $y_{\mathrm{m},t}$, the local gravity vector g^{n} and the local magnetic field m^{n}. These vectors are linearly independent except when the measurements are obtained on the magnetic north or south poles where the dip angle is $\delta = 0$ and the magnetic field does not contain any heading information.

The accelerometer provides information about the sensor's inclination. Heading information is provided by the magnetometer. However, at all locations except on the equator, the magnetometer also provides information about the inclination due to its non-zero vertical component, see (3.51). In practice, the accelerometer typically provides more accurate inclination information. Hence, we choose to use the magnetometer only to provide heading information by projecting the magnetic field and the

magnetometer measurement on the horizontal plane. Furthermore, we normalize the vectors. Because of this, an adapted model uses the four normalized vectors

$$\hat{g}^{\mathrm{n}} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^{\mathsf{T}}, \qquad \hat{g}^{\mathrm{b}} = \tfrac{y_{\mathrm{a},1}}{\|y_{\mathrm{a},1}\|_2}, \tag{3.64a}$$

$$\hat{m}^{\mathrm{n}} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^{\mathsf{T}}, \qquad \hat{m}^{\mathrm{b}} = \hat{g}^{\mathrm{b}} \times \left(\tfrac{y_{\mathrm{m},1}}{\|y_{\mathrm{m},1}\|_2} \times \hat{g}^{\mathrm{b}} \right). \tag{3.64b}$$

A number of algorithms are available to estimate the orientation from these vectors. Well-known examples are the TRIAD algorithm, the QUEST algorithm, see *e.g.* [**?**], and the method presented in [**?**]. For our problem at hand, these methods give equivalent results, even though they use slightly different solution strategies. Generally speaking, they solve the problem of determining the rotation q^{nb} from

$$\begin{aligned} \underset{q^{\mathrm{nb}}}{\arg\min} \quad & \|\bar{\hat{g}}^{\mathrm{n}} - q^{\mathrm{nb}} \odot \bar{\hat{g}}^{\mathrm{b}} \odot q^{\mathrm{bn}}\|_2^2 + \|\bar{\hat{m}}^{\mathrm{n}} - q^{\mathrm{nb}} \odot \bar{\hat{m}}^{\mathrm{b}} \odot q^{\mathrm{bn}}\|_2^2 \\ \text{subj. to} \quad & \|q^{\mathrm{nb}}\|_2 = 1, \end{aligned} \tag{3.65}$$

Recall from (3.24) that $q^{\mathrm{nb}} \odot \bar{x}^{\mathrm{b}} \odot q^{\mathrm{bn}}$ is the rotation of the vector x^{b} to the n-frame. The optimization problem (3.65) therefore determines the orientation q^{nb} that minimizes the distance between the normalized magnetic field and gravity vectors measured in the first sample and the normalized magnetic field and gravity vectors in the navigation frame. These four vectors were defined in (3.64).

Defining

$$A = -\left(\bar{\hat{g}}^{\mathrm{n}}\right)^{\mathsf{L}} \left(\bar{\hat{g}}^{\mathrm{b}}\right)^{\mathsf{R}} - \left(\bar{\hat{m}}^{\mathrm{n}}\right)^{\mathsf{L}} \left(\bar{\hat{m}}^{\mathrm{b}}\right)^{\mathsf{R}}, \tag{3.66}$$

where the left and right quaternion multiplications are defined in (3.28), (3.65) can equivalently be written as

$$\begin{aligned} \breve{q}_1^{\mathrm{nb}} = \underset{q^{\mathrm{nb}}}{\arg\min} \quad & \left(q^{\mathrm{nb}}\right)^{\mathsf{T}} A q^{\mathrm{nb}} \\ \text{subj. to} \quad & \|q^{\mathrm{nb}}\|_2 = 1. \end{aligned} \tag{3.67}$$

For a derivation, see [**?**]. The solution to this problem is given by the eigenvector corresponding to the largest eigenvalue of A. Note that although this method can be used to compute the orientation from any two linearly independent vectors in two coordinate frames, we only use it to compute a prior on the orientation.

Based on the estimate $\breve{q}_1^{\mathrm{nb}}$ from (3.67), we can model the orientation at time $t = 1$ in terms of an orientation deviation

$$q_1^{\mathrm{nb}} = \exp_{\mathsf{q}}\left(\tfrac{e_{\eta,\mathrm{i}}}{2}\right) \odot \breve{q}_1^{\mathrm{nb}}, \qquad e_{\eta,\mathrm{i}} \sim \mathcal{N}(0, \Sigma_{\eta,\mathrm{i}}), \tag{3.68a}$$

or in terms of a quaternion as

$$q_1^{\mathrm{nb}} = \breve{q}_1^{\mathrm{nb}} + e_{\mathrm{q,i}}, \qquad e_{\mathrm{q,i}} \sim \mathcal{N}(0, \Sigma_{\mathrm{q,i}}). \tag{3.68b}$$

Explicit formulations for the covariance of the orientation estimates from the TRIAD and QUEST algorithms are discussed by [**?**]. In practice, however, the accuracy of the estimates from (3.67) highly depends on the validity of the model assumptions, *i.e.* on whether the sensor is indeed far from magnetic material and whether the linear acceleration is indeed zero. Because this has such a significant influence on the quality of the estimates, we choose $\Sigma_{\eta,\mathrm{i}}$ and $\Sigma_{\mathrm{q,i}}$ somewhat conservatively. Modeling that in 68% of the cases the orientation error is less than 20°,

$$\Sigma_{\eta,\mathrm{i}} = \sigma_{\eta,\mathrm{i}}^2 \mathcal{I}_3, \qquad \sigma_{\eta,\mathrm{i}} = \tfrac{20}{180}\pi, \tag{3.69a}$$

$$\Sigma_{\mathrm{q,i}} = \tfrac{1}{4}\left(\breve{q}_1^{\mathrm{nb}}\right)^{\mathsf{R}} \tfrac{\partial \exp_{\mathsf{q}}(e_{\eta,\mathrm{i}})}{\partial e_{\eta,\mathrm{i}}} \Sigma_{\eta,\mathrm{i}} \left(\tfrac{\partial \exp_{\mathsf{q}}(e_{\eta,\mathrm{i}})}{\partial e_{\eta,\mathrm{i}}}\right)^{\mathsf{T}} \left(\breve{q}_1^{\mathrm{bn}}\right)^{\mathsf{R}}, \tag{3.69b}$$

where we use the fact that $(q^{\mathsf{R}})^{\mathsf{T}} = (q^{\mathsf{c}})^{\mathsf{R}}$. Note that the covariance $\Sigma_{\mathrm{q,i}}$ is defined in terms of the covariance $\Sigma_{\eta,\mathrm{i}}$ to allow for explicit comparison between different algorithms in Chapter 4.

3.7 Resulting probabilistic models

The information from the previous sections can now be combined into one model which will be used in the algorithms in Chapter 4. In this section, we describe our modeling choices for the pose estimation problem and for the orientation estimation problem.

We assume that the sensor does not travel over significant distances as compared to the size of the earth and hence keep the navigation frame n fixed with respect to the earth frame e. Furthermore, we assume that the magnitude of the earth rotation and of the Coriolis acceleration are negligible. Our gyroscope and accelerometer models are hence given by

$$y_{\mathrm{a},t} = R_t^{\mathrm{bn}}(a_{\mathrm{nn}}^{\mathrm{n}} - g^{\mathrm{n}}) + \delta_{\mathrm{a},t}^{\mathrm{b}} + e_{\mathrm{a},t}^{\mathrm{b}}, \tag{3.70a}$$

$$y_{\omega,t} = \omega_{\mathrm{nb},t}^{\mathrm{b}} + \delta_{\omega,t}^{\mathrm{b}} + e_{\omega,t}^{\mathrm{b}}. \tag{3.70b}$$

In the remainder, for notational convenience we drop the subscripts n which indicate in which frame the differentiation is performed, see §2.3, and use the shorthand notation a^{n} for $a_{\mathrm{nn}}^{\mathrm{n}}$. Furthermore, we will denote $\omega_{\mathrm{nb}}^{\mathrm{b}}$ simply by ω and omit the superscript b on the noise terms $e_{\mathrm{a},t}$ and $e_{\omega,t}$ and the bias terms $\delta_{\mathrm{a},t}$ and $\delta_{\omega,t}$. We assume that the inertial measurement noise is given by

$$e_{\mathrm{a},t} \sim \mathcal{N}(0, \sigma_{\mathrm{a}}^2 \mathcal{I}_3), \qquad e_{\omega,t} \sim \mathcal{N}(0, \sigma_{\omega}^2 \mathcal{I}_3), \tag{3.71}$$

i.e. we assume that the three sensor axes are independent and have the same noise levels.

3.7.1 Pose estimation

For pose estimation, we model the accelerometer and gyroscope measurements as inputs to the dynamics. Hence, the state vector consists of the position p_t^{n}, the velocity v_t^{n} and a parametrization of the orientation. We use the inertial measurements in combination with position measurements to estimate the pose.

Using the accelerometer measurement model (3.70a) in (3.56), the dynamics of the position and velocity is given by

$$p_{t+1}^{\mathrm{n}} = p_t^{\mathrm{n}} + T v_t^{\mathrm{n}} + \tfrac{T^2}{2}\left(R_t^{\mathrm{nb}}(y_{\mathrm{a},t} - \delta_{\mathrm{a},t}) + g^{\mathrm{n}} + e_{\mathrm{a},t}\right), \tag{3.72a}$$

$$v_{t+1}^{\mathrm{n}} = v_t^{\mathrm{n}} + T\left(R_t^{\mathrm{nb}}(y_{\mathrm{a},t} - \delta_{\mathrm{a},t}) + g^{\mathrm{n}} + e_{\mathrm{a},t}\right), \tag{3.72b}$$

where without loss of generality, we switch the sign on the noise. Note that the noise term $e_{\mathrm{a},t}$ should be rotated to the navigation frame n by multiplying it with the rotation matrix R_t^{nb}. However, because of the assumption (3.71), the rotation matrix can be omitted without loss of generality. The dynamics of the orientation parametrized using quaternions is given by

$$q_{t+1}^{\mathrm{nb}} = q_t^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} - \delta_{\omega,t} - e_{\omega,t})\right). \tag{3.73}$$

Equivalent dynamic models can be obtained for the other parametrizations.

The position measurements are modeled as in (3.53). In summary, this leads to the following state space model for pose estimation

$$\begin{pmatrix} p_{t+1}^{\mathrm{n}} \\ v_{t+1}^{\mathrm{n}} \\ q_{t+1}^{\mathrm{nb}} \end{pmatrix} = \begin{pmatrix} p_t^{\mathrm{n}} + T v_t^{\mathrm{n}} + \frac{T^2}{2}\left(R_t^{\mathrm{nb}}(y_{\mathrm{a},t} - \delta_{\mathrm{a},t}) + g^{\mathrm{n}} + e_{\mathrm{p,a},t}\right) \\ v_t^{\mathrm{n}} + T\left(R_t^{\mathrm{nb}}(y_{\mathrm{a},t} - \delta_{\mathrm{a},t}) + g^{\mathrm{n}} + e_{\mathrm{v,a},t}\right) \\ q_t^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\frac{T}{2}(y_{\omega,t} - \delta_{\omega,t} - e_{\omega,t})\right) \end{pmatrix}, \tag{3.74a}$$

$$y_{\mathrm{p},t} = p_t^{\mathrm{n}} + e_{\mathrm{p},t}, \tag{3.74b}$$

where

$$e_{\mathrm{p,a},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{a}}), \qquad e_{\mathrm{v,a},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{a}}), \tag{3.74c}$$

$$e_{\mathrm{p},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{p}}), \qquad e_{\omega,t} \sim \mathcal{N}(0, \Sigma_{\omega}), \tag{3.74d}$$

with $\Sigma_{\mathrm{a}} = \sigma_{\mathrm{a}}^2 \mathcal{I}_3$ and $\Sigma_{\omega} = \sigma_{\omega}^2 \mathcal{I}_3$. Note that we model the process noise on the position and velocity states in terms of the accelerometer noise. However, we do not enforce these to have the same noise realization. Hence, we use the notation $e_{\mathrm{p,a},t}$ and $e_{\mathrm{v,a},t}$ for the two process noise terms. The covariance of both is equal to the covariance of the accelerometer noise. The initial position is assumed to be given by the first position measurement as

$$p_1^{\mathrm{n}} = y_{\mathrm{p},1} + e_{\mathrm{p},1}, \qquad e_{\mathrm{p},1} \sim \mathcal{N}(0, \Sigma_{\mathrm{p,i}}), \tag{3.74e}$$

while the initial velocity is assumed to be approximately zero as

$$v_1^{\mathrm{n}} = e_{\mathrm{v,i}}, \qquad e_{\mathrm{v,i}} \sim \mathcal{N}(0, \Sigma_{\mathrm{v,i}}). \tag{3.74f}$$

The orientation at time $t = 1$ is given by the QUEST algorithm described in §3.6, parametrized in terms of quaternions or rotation vectors as

$$q_1^{\mathrm{nb}} = \exp_{\mathsf{q}}\left(\tfrac{e_{\eta,\mathrm{i}}}{2}\right) \odot \breve{q}_1^{\mathrm{nb}}, \qquad e_{\eta,\mathrm{i}} \sim \mathcal{N}(0, \Sigma_{\eta,\mathrm{i}}), \tag{3.74g}$$

$$q_1^{\mathrm{nb}} = \breve{q}_1^{\mathrm{nb}} + e_{\mathrm{q,i}}, \qquad e_{\mathrm{q,i}} \sim \mathcal{N}(0, \Sigma_{\mathrm{q,i}}), \tag{3.74h}$$

where the initial orientation uncertainty is given in terms of a standard deviation of 20°.

In Chapter 4 we assume that the inertial measurement are properly calibrated. Hence, we assume that their biases $\delta_{\mathrm{a},t}$ and $\delta_{\omega,t}$ are zero. Calibration is the topic of Chapter 5 where we will also introduce possible extensions of the state space model in which the bias terms are included either as states or as unknown parameters.

3.7.2 Orientation estimation

For orientation estimation, the state vector only consists of a parametrization of the orientation. We use the inertial sensors in combination with the magnetometer measurements to estimate the orientation. The magnetometer measurements are modeled as in (3.52). Instead of using the accelerometer measurements model (3.70a), we use the model (3.50) where it is assumed that the linear acceleration is approximately zero. This leads to the following state space model for orientation estimation,

$$q_{t+1}^{\mathrm{nb}} = q_t^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} - \delta_\omega - e_{\omega,t})\right), \tag{3.75a}$$

$$y_{\mathrm{a},t} = -R_t^{\mathrm{bn}} g^{\mathrm{n}} + e_{\mathrm{a},t}, \tag{3.75b}$$

$$y_{\mathrm{m},t} = R_t^{\mathrm{bn}} m^{\mathrm{n}} + e_{\mathrm{m},t}, \tag{3.75c}$$

where (3.75a) describes the dynamics while (3.75b) and (3.75c) describe the measurement models and

$$e_{\omega,t} \sim \mathcal{N}(0, \Sigma_\omega), \qquad e_{\mathrm{a},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{a}}), \qquad e_{\mathrm{m},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{m}}), \tag{3.75d}$$

with $\Sigma_\omega = \sigma_\omega^2 \mathcal{I}_3$ and $\Sigma_{\mathrm{a}} = \sigma_{\mathrm{a}}^2 \mathcal{I}_3$. The initial orientation is given by the QUEST algorithm described in §3.6 and is modeled as in (3.74g) or (3.74h). Also for orientation estimation, in Chapter 4 we assume that the inertial measurements are properly calibrated. Hence, we assume that the bias $\delta_{\omega,t}$ is zero.

Chapter 4

Estimating Position and Orientation

In this chapter we will focus on position and orientation estimation using the models (3.74) and (3.75) derived in Chapter 3. In §4.1, we will first describe a method to solve the smoothing problem (3.3a). Subsequently, in §4.2–§4.4, we will derive different methods for solving the filtering problem (3.3b). In each section, after a general introduction of the estimation method, we will illustrate the method by explicitly deriving algorithms to estimate the orientation using the state space model (3.75). The orientation estimation problem also illustrates the most important parts of the pose estimation problem, since most complexities lie in the parametrization of the orientation and in the nonlinear nature of the orientation. In §4.5, we show some characteristics of the different algorithms for the orientation estimation problem. In §4.6, we will discuss how the algorithms for orientation estimation can be extended to also estimate the position. Throughout this section, we assume that the sensors are calibrated, *i.e.* we assume that we do not have any unknown parameters θ in our models. Because of this, the models that we use are the most basic models that can be used for position and orientation estimation using inertial sensors.

4.1 Smoothing in an optimization framework

Perhaps the most intuitive way to solve the smoothing problem is by posing it as an optimization problem, where a *maximum a posteriori (MAP)* estimate is obtained as

$$\begin{aligned}\hat{x}_{1:N} &= \arg\max_{x_{1:N}} p(x_{1:N} \mid y_{1:N}) \\ &= \arg\max_{x_{1:N}} p(x_1) \prod_{t=2}^{N} p(x_t \mid x_{t-1}) p(y_t \mid x_t). \end{aligned} \tag{4.1}$$

Here, we use the notation in terms of probability distributions as introduced in §3.1 and model the measurements and the state dynamics as described in §3.4 and §3.5, respectively. Furthermore, we assume that a prior on the initial state is obtained using the measurements at $t = 1$ as described in §3.6. Because of this, the measurement model $p(y_1 \mid x_1)$ from (3.3a) is explicitly omitted in (4.1). Note that in practice, we typically minimize $-\log p(x_{1:N} \mid y_{1:N})$ instead of maximizing $p(x_{1:N} \mid y_{1:N})$ itself, resulting in the optimization problem

$$\arg\min_{x_{1:N}} -\log p(x_1) - \sum_{t=2}^{N} \log p(x_t \mid x_{t-1}) - \sum_{t=2}^{N} \log p(y_t \mid x_t). \tag{4.2}$$

There are various ways to solve problems of this kind, for instance particle smoothers [**?**], an extended Rauch-Tung-Striebel (RTS) smoother [**?**] and optimization methods, see *e.g.* [**?** **?**]. The latter approach is closely related to iterated Kalman smoothers [**?** **?**]. We will solve the problem using an optimization method. Compared to extended RTS smoothers, optimization methods allow for more flexibility in the models that are being used. For instance, additional information outside of the standard state space model can straightforwardly be included. Optimization approaches are typically computationally heavier than extended RTS smoothers but less heavy than particle smoothers. The latter are capable of capturing the whole distribution, which is a clear advantage when the distributions are multi-modal. Optimization instead gives a point estimate and an associated measure of uncertainty. This is typically sufficient for position and orientation estimation using inertial sensors.

4.1.1 Gauss-Newton optimization

To obtain a smoothing estimate of the position and orientation using optimization, we first recognize that for our models (3.74) and (3.75), all probability distributions in (4.2) are Gaussian. Let us therefore consider a slightly more general problem where the objective function consists of the product of Gaussian probability functions $p(e_i(x_{1:N}))$, $i = 1, \ldots, M$. Hence, the optimization problem can be written as

$$\hat{x}_{1:N} = \arg\min_{x_{1:N}} -\sum_{i=1}^{M} \log p\left(e_i(x_{1:N})\right). \tag{4.3}$$

The probability distribution of $e_i(x)$ is given by

$$p\left(e_i(x_{1:N})\right) = \frac{1}{\sqrt{(2\pi)^{n_e} \det \Sigma_i}} \exp\left(-\tfrac{1}{2} e_i^{\mathsf{T}}(x_{1:N}) \Sigma_i^{-1} e_i(x_{1:N})\right). \tag{4.4}$$

Omitting the terms independent of $x_{1:N}$, the optimization problem (4.3) reduces to

$$\hat{x}_{1:N} = \arg\min_{x_{1:N}} \tfrac{1}{2} \sum_{i=1}^{M} \|e_i(x_{1:N})\|^2_{\Sigma_i^{-1}}, \tag{4.5}$$

with $\|e_i(x_{1:N})\|^2_{\Sigma_i^{-1}} = e_i^{\mathsf{T}}(x_{1:N}) \Sigma_i^{-1} e_i(x_{1:N})$. The function that is being minimized in optimization problems, is often referred to as the *objective function*.

The solution to (4.5) can be found by studying the shape of the objective function as a function of $x_{1:N}$. This can be characterized in terms of the *gradient* $\mathcal{G}(x_{1:N})$ and *Hessian* $\mathcal{H}(x_{1:N})$, which provide information about the slope and curvature of the function, respectively. Defining

$$e_i^{\mathsf{T}}(x_{1:N}) \Sigma_i^{-1} e_i(x_{1:N}) = \varepsilon_i^{\mathsf{T}} \varepsilon_i, \qquad \varepsilon_i = \Sigma_i^{-1/2} e_i(x_{1:N}),$$

and the stacked variables

$$\varepsilon = \begin{pmatrix} \varepsilon_1^{\mathsf{T}} & \cdots & \varepsilon_M^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}},$$

the gradient and the Hessian are given by

$$\mathcal{G}(x_{1:N}) = \sum_{i=1}^{M} \left(\frac{\partial \varepsilon_i}{\partial x_{1:N}}\right)^{\mathsf{T}} \varepsilon_i = \mathcal{J}^{\mathsf{T}}(x_{1:N}) \varepsilon, \tag{4.6a}$$

$$\begin{aligned} \mathcal{H}(x_{1:N}) &= \sum_{i=1}^{M} \left(\left(\frac{\partial \varepsilon_i}{\partial x_{1:N}}\right)^{\mathsf{T}} \frac{\partial \varepsilon_i}{\partial x_{1:N}} + \varepsilon_i^{\mathsf{T}} \frac{\partial^2 \varepsilon_i}{\partial x_{1:N}^2} \right) \\ &= \mathcal{J}^{\mathsf{T}}(x_{1:N}) \mathcal{J}(x_{1:N}) + \sum_{i=1}^{M} \varepsilon_i^{\mathsf{T}} \frac{\partial^2 \varepsilon_i}{\partial x_{1:N}^2}. \end{aligned} \tag{4.6b}$$

Note that for notational convenience, we have omitted the explicit dependence of ε on $x_{1:N}$. In (4.6), we introduced the notation $\mathcal{J}(x_{1:N})$, which is the *Jacobian* of the vector ε with respect to $x_{1:N}$ as

$$\mathcal{J}(x_{1:N}) = \begin{pmatrix} \frac{\partial \varepsilon_1}{\partial x_1} & \cdots & \frac{\partial \varepsilon_1}{\partial x_N} \\ \vdots & & \vdots \\ \frac{\partial \varepsilon_{Mn_\varepsilon}}{\partial x_1} & \cdots & \frac{\partial \varepsilon_{Mn_\varepsilon}}{\partial x_N} \end{pmatrix}, \tag{4.7}$$

where n_ε is the length of the vector ε_i. Instead of computing the true Hessian (4.6b), we compute an approximation of it [**?**], given by

$$\hat{\mathcal{H}}(x_{1:N}) = \mathcal{J}^{\mathsf{T}}(x_{1:N}) \mathcal{J}(x_{1:N}). \tag{4.8}$$

This has the benefit of not having to compute second derivatives, at the same time as it guarantees that the Hessian is positive semidefinite. The downside of using (4.8) is that it introduces an approximation.

The gradient and the (approximate) Hessian can be used to find the minimum of the objective function. For our models (3.74) and (3.75), in which the functions $e_i(x_{1:N})$ are nonlinear, an estimate $\hat{x}_{1:N}$ can iteratively be computed as

$$\hat{x}_{1:N}^{(k+1)} = \hat{x}_{1:N}^{(k)} - \beta^{(k)} \left(\hat{\mathcal{H}}(\hat{x}_{1:N}^{(k)})\right)^{-1} \mathcal{G}(\hat{x}_{1:N}^{(k)}), \tag{4.9}$$

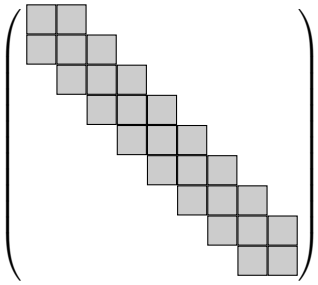


Figure 4.1: An illustration of the sparsity pattern that is present in the smoothing problem.

where k denotes the iteration number. The *step length* $\beta^{(k)}$ is computed for instance using a backtracking line search [**?** **?**]. The *search direction* is computed as $\left(\hat{\mathcal{H}}(\hat{x}_{1:N}^{(k)})\right)^{-1} \mathcal{G}(\hat{x}_{1:N}^{(k)})$. Note that an initial point $\hat{x}_{1:N}^{(0)}$ needs to be chosen close enough to the desired minimum to ensure convergence to this minimum.

In case the functions $e_i(x_{1:N})$ would be linear, the problem (4.5) would be a *least squares (LS)* problem for which the update (4.9) directly leads to the minimum of the objective function, irrespective of the initial point. In our case where the functions $e_i(x_{1:N})$ are nonlinear due to the nonlinear nature of the orientation, the problem (4.5) is instead a *nonlinear least squares (NLS)* problem. Each iteration in (4.9) can be interpreted as solving a LS problem around a linearization point. The linearization point is updated after each iteration, bringing it closer to the minimum of the objective function. Computing an estimate $\hat{x}_{1:N}$ by iterating (4.9) until convergence, making use of the approximate Hessian from (4.8), is called *Gauss-Newton optimization*.

Note that since the inertial sensors sample at high sampling rates, the length of the vector $x_{1:N}$ quickly becomes fairly large. For instance, for inertial sensors sampling at 100Hz for 10 seconds, $N = 1\,000$ and the size of the (approximate) Hessian $\hat{\mathcal{H}}(x)$ is $1\,000n_x \times 1\,000n_x$, where n_x is the length of the vector x_t. However, as can be seen in (4.2), the components of the objective function only depend on the current and next time steps x_t and x_{t+1}. Hence, the structure of the Hessian (4.8) is of the form given in Figure 4.1. There exist efficient algorithms to compute search directions for problems with this sparsity pattern, which can be exploited using sparse matrix packages, see *e.g.* [**?**], or by using tools like dynamic programming and message passing [**?** **?** **?**].

4.1.2 Smoothing estimates of the orientation using optimization

In this section, we will illustrate the use of Gauss-Newton optimization to obtain smoothing estimates of the orientation. As discussed in the previous section, the crucial part is to identify the objective function and its Jacobian. From this, the gradient and approximate Hessian can be computed using (4.6a) and (4.8), which can be used to iteratively update the estimates using (4.9).

Combining the general formulation of the smoothing problem (4.2) and using the model for orientation estimation (3.75), the orientation smoothing problem is given by

$$\hat{x}_{1:N} = \underset{x_{1:N}}{\arg\min} \underbrace{\|e_{\eta,\mathrm{i}}\|^2_{\Sigma^{-1}_{\eta,\mathrm{i}}}}_{\text{Prior}} + \underbrace{\sum_{t=2}^{N} \|e_{\omega,t}\|^2_{\Sigma^{-1}_{\omega}}}_{\text{Dynamics}} + \underbrace{\sum_{t=2}^{N} \left(\|e_{\mathrm{a},t}\|^2_{\Sigma^{-1}_{\mathrm{a}}} + \|e_{\mathrm{m},t}\|^2_{\Sigma^{-1}_{\mathrm{m}}} \right)}_{\text{Measurement models}}, \tag{4.10}$$

with

$$e_{\eta,\mathrm{i}} = 2\log_{\mathsf{q}}\left(q_1^{\mathrm{nb}} \odot \breve{q}_1^{\mathrm{bn}}\right), \qquad e_{\eta,\mathrm{i}} \sim \mathcal{N}(0, \Sigma_{\eta,\mathrm{i}}), \tag{4.11a}$$

$$e_{\omega,t} = \tfrac{2}{T}\log_{\mathsf{q}}\left(q_t^{\mathrm{bn}} \odot q_{t+1}^{\mathrm{nb}}\right) - y_{\omega,t}, \qquad e_{\omega,t} \sim \mathcal{N}(0, \Sigma_{\omega}), \tag{4.11b}$$

$$e_{\mathrm{a},t} = y_{\mathrm{a},t} + R_t^{\mathrm{bn}} g^{\mathrm{n}}, \qquad e_{\mathrm{a},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{a}}), \tag{4.11c}$$

$$e_{\mathrm{m},t} = y_{\mathrm{m},t} - R_t^{\mathrm{bn}} m^{\mathrm{n}}, \qquad e_{\mathrm{m},t} \sim \mathcal{N}(0, \Sigma_{\mathrm{m}}). \tag{4.11d}$$

Convex equality constraints can straightforwardly be incorporated in optimization problems. However, using N equality constraints to preserve the unit norm of the quaternions, severely complicates the

optimization problem since these norm constraints are non-convex. Instead, we encode an orientation in terms of a linearization point parametrized as a unit quaternion $\tilde{q}_t^{\mathrm{nb}}$ and an orientation deviation parametrized as a rotation vector η_t^{n} as discussed in §3.3.1. Hence, we model the orientation as

$$q_t^{\mathrm{nb}} = \exp_{\mathsf{q}}\left(\frac{\eta_t^{\mathrm{n}}}{2}\right) \odot \tilde{q}_t^{\mathrm{nb}}. \tag{4.12}$$

At each Gauss-Newton iteration (4.9), we estimate the state vector $\eta_{1:N}^{\mathrm{n}}$. Before starting the next iteration, the linearization points $\tilde{q}_{1:N}^{\mathrm{nb}}$ are updated and the state vector $\eta_{1:N}^{\mathrm{n}}$ is reset to zero.

Using the notation introduced in §3.3.1, the objective function (4.10) can be expressed in terms of the orientation deviation η_t^{n} by rewriting (4.11) as

$$e_{\eta,\mathrm{i}} = 2\log_{\mathsf{q}}\left(\exp_{\mathsf{q}}(\tfrac{\eta_1^{\mathrm{n}}}{2}) \odot \tilde{q}_1^{\mathrm{nb}} \odot \breve{q}_1^{\mathrm{bn}}\right), \tag{4.13a}$$

$$e_{\omega,t} = \tfrac{2}{T}\log_{\mathsf{q}}\left((\tilde{q}_t^{\mathrm{bn}} \odot \exp_{\mathsf{q}}(\tfrac{\eta_t^{\mathrm{n}}}{2}))^{\mathsf{c}} \odot \exp_{\mathsf{q}}(\tfrac{\eta_{t+1}^{\mathrm{n}}}{2}) \odot \tilde{q}_{t+1}^{\mathrm{nb}}\right) - y_{\omega,t}, \tag{4.13b}$$

$$e_{\mathrm{a},t} = y_{\mathrm{a},t} + \tilde{R}_t^{\mathrm{bn}}\left(\exp_{\mathsf{R}}(\eta_t^{\mathrm{n}})\right)^{\mathsf{T}} g^{\mathrm{n}}, \tag{4.13c}$$

$$e_{\mathrm{m},t} = y_{\mathrm{m},t} - \tilde{R}_t^{\mathrm{bn}}\left(\exp_{\mathsf{R}}(\eta_t^{\mathrm{n}})\right)^{\mathsf{T}} m^{\mathrm{n}}. \tag{4.13d}$$

Here, $\tilde{R}_t^{\mathrm{bn}}$ is the rotation matrix representation of the linearization point $\tilde{q}_t^{\mathrm{bn}}$. This leads to the following derivatives

$$\frac{\partial e_{\eta,\mathrm{i}}}{\partial \eta_1^{\mathrm{n}}} = \frac{\partial \log_{\mathsf{q}}(q)}{\partial q}\left(\tilde{q}_1^{\mathrm{nb}} \odot \breve{q}_1^{\mathrm{bn}}\right)^{\mathsf{R}} \frac{\partial \exp_{\mathsf{q}}(\eta_1^{\mathrm{n}})}{\partial \eta_1^{\mathrm{n}}}, \tag{4.14a}$$

$$\frac{\partial e_{\omega,t}}{\partial \eta_{t+1}^{\mathrm{n}}} = \frac{1}{T}\frac{\partial \log_{\mathsf{q}}(q)}{\partial q}\left(\tilde{q}_t^{\mathrm{bn}}\right)^{\mathsf{L}}\left(\tilde{q}_{t+1}^{\mathrm{nb}}\right)^{\mathsf{R}} \frac{\partial \exp_{\mathsf{q}}(\eta_{t+1}^{\mathrm{n}})}{\partial \eta_{t+1}^{\mathrm{n}}}, \tag{4.14b}$$

$$\frac{\partial e_{\omega,t}}{\partial \eta_t^{\mathrm{n}}} = \frac{1}{T}\frac{\partial \log_{\mathsf{q}}(q)}{\partial q}\left(\tilde{q}_t^{\mathrm{bn}}\right)^{\mathsf{L}}\left(\tilde{q}_{t+1}^{\mathrm{nb}}\right)^{\mathsf{R}} \frac{\partial\left(\exp_{\mathsf{q}}(\eta_t^{\mathrm{n}})\right)^{\mathsf{c}}}{\partial \exp_{\mathsf{q}}(\eta_t^{\mathrm{n}})}\frac{\partial \exp_{\mathsf{q}}(\eta_t^{\mathrm{n}})}{\partial \eta_t^{\mathrm{n}}}, \tag{4.14c}$$

$$\frac{\partial e_{\mathrm{a},t}}{\partial \eta_t^{\mathrm{n}}} \approx \tilde{R}_t^{\mathrm{bn}}[g^{\mathrm{n}}\times], \tag{4.14d}$$

$$\frac{\partial e_{\mathrm{m},t}}{\partial \eta_t^{\mathrm{n}}} \approx -\tilde{R}_t^{\mathrm{bn}}[m^{\mathrm{n}}\times], \tag{4.14e}$$

where, using (3.40) and the definition of the quaternion conjugate (3.25),

$$\frac{\partial \log_{\mathsf{q}}(q)}{\partial q} \approx \begin{pmatrix} 0_{1\times 3} \\ \mathcal{I}_3 \end{pmatrix}^{\mathsf{T}}, \qquad \frac{\partial\left(\exp_{\mathsf{q}}(\eta)\right)^{\mathsf{c}}}{\partial \exp \eta} = \begin{pmatrix} 1 & 0_{1\times 3} \\ 0_{3\times 1} & -\mathcal{I}_3 \end{pmatrix},$$
$$\frac{\partial \exp_{\mathsf{q}} \eta}{\partial \eta} \approx \begin{pmatrix} 0_{1\times 3} \\ \mathcal{I}_3 \end{pmatrix}. \tag{4.15}$$

Using the approximate derivatives (4.14), the gradient and approximate Hessian can be computed for the Gauss-Newton iterations. The resulting solution is summarized in Algorithm 1. One of the inputs to the algorithm is an initial estimate of the orientation $\tilde{q}_{1:N}^{\mathrm{nb},(0)}$, which will aid the convergence to the desired minimum. There are (at least) two ways to obtain good initial estimates $\tilde{q}_{1:N}^{\mathrm{nb},(0)}$. First, they can be obtained by direct integration of the gyroscope measurements. As discussed in §1.2, these estimates will suffer from integration drift. However, they still form a good initial estimate for the optimization problem. Second, one of the other, computationally cheaper estimation algorithms that will be discussed in the remainder of this section can be used to obtain initial estimates of the orientation.

4.1.3 Computing the uncertainty

As discussed in §3.1, we are not only interested in obtaining point estimates of the position and orientation, but also in estimating their uncertainty. In our case of Gaussian noise, this is characterized by the covariance. As shown in *e.g.* [**?** **?**], if our position and orientation estimation problems would be LS problems, the covariance of the estimates would be given by the inverse of the Hessian of the objective function (4.6b). Instead, we solve an NLS problem, for which a number of LS problems are solved around linearization points closer and closer to the minimum of the objective function. Hence, when the algorithm has converged, the problem can locally well be described by the quadratic approximation around its resulting linearization point. We can therefore approximate the covariance of our estimates as

$$\operatorname{cov}\left(\hat{x}_{1:N}\right) = \left(\mathcal{J}^{\mathsf{T}}(\hat{x}_{1:N})\mathcal{J}(\hat{x}_{1:N})\right)^{-1}. \tag{4.18}$$

Algorithm 1 Smoothing estimates of the orientation using optimization

INPUTS: An initial estimate of the orientation $\tilde{q}^{\mathrm{nb},(0)}_{1:N}$, inertial data $\{y_{\mathrm{a},t}, y_{\omega,t}\}_{t=1}^{N}$, magnetometer data $\{y_{\mathrm{m},t}\}_{t=1}^{N}$ and covariance matrices Σ_ω, Σ_{a} and Σ_{m}.
OUTPUTS: An estimate of the orientation $\hat{q}^{\mathrm{nb}}_{1:N}$ and optionally its covariance $\mathrm{cov}(\hat{\eta}^{\mathrm{n}}_{1:N})$.

1. Set $\hat{\eta}^{\mathrm{n},(0)}_t = 0_{3\times 1}$ for $t = 1, \ldots, N$, set $k = 0$ and compute $\breve{q}^{\mathrm{nb}}_1$ and $\Sigma_{\eta,\mathrm{i}}$ as described in §3.6.
2. **while** *termination condition is not satisfied* **do**
   (a) Compute the gradient (4.6a) and the approximate Hessian (4.8) of the orientation smoothing problem (4.10) using the expressions for the different parts of the cost function and their Jacobians (4.13) and (4.14).
   (b) Apply the update (4.9) to obtain $\hat{\eta}^{\mathrm{n},(k+1)}_{1:N}$.
   (c) Update the linearization point as
   $$\tilde{q}^{\mathrm{nb},(k+1)}_t = \exp_{\mathsf{q}}\left(\frac{\hat{\eta}^{\mathrm{n},(k+1)}_t}{2}\right) \odot \tilde{q}^{\mathrm{nb},(k)}_t, \tag{4.16}$$
   and set $\hat{\eta}^{\mathrm{n},(k+1)}_t = 0_{3\times 1}$ for $t = 1, \ldots, N$.
   (d) Set $k = k + 1$.
   end while
3. Set $\hat{q}^{\mathrm{nb}}_{1:N} = \tilde{q}^{\mathrm{nb},(k)}_{1:N}$.
4. Optionally compute
   $$\mathrm{cov}(\hat{\eta}^{\mathrm{n}}_{1:N}) = \left(\mathcal{J}^{\mathsf{T}}(\hat{\eta}^{\mathrm{n}}_{1:N})\mathcal{J}(\hat{\eta}^{\mathrm{n}}_{1:N})\right)^{-1}. \tag{4.17}$$

An intuition behind this expression is that the accuracy of the estimates is related to the sensitivity of the objective function with respect to the states. The covariance of the estimate will play a crucial role in the filtering approaches discussed in §4.2 and §4.3.

The matrix $\mathcal{J}^{\mathsf{T}}(\hat{x}_{1:N})\mathcal{J}(\hat{x}_{1:N})$ quickly becomes fairly large due to the high sampling rates of the inertial sensors. Hence, computing its inverse can be computationally costly. We are, however, typically only interested in a subset of the inverse. For instance, we are often only interested in diagonal or block diagonal elements representing $\mathrm{cov}(x_t)$. It is therefore not necessary to explicitly form the complete inverse, which makes the computation tractable also for larger problem sizes.

4.2 Filtering estimation in an optimization framework

One of the benefits of using a smoothing formulation is that all measurements $y_{1:N}$ are used to get the best estimates of the states $x_{1:N}$. However, both the computational cost and the memory requirements grow with the length of the data set. Furthermore, it is a post-processing solution in which we have to wait until all data is available. Alternatively, the filtering problem can be formulated as

$$\begin{aligned}\hat{x}_{t+1} &= \arg\min_{x_{t+1}} -\log p(x_{t+1} \mid y_{1:t+1}) \\ &= \arg\min_{x_{t+1}} -\log p(y_{t+1} \mid x_{t+1}) - \log p(x_{t+1} \mid y_{1:t}).\end{aligned} \tag{4.19}$$

Note that without loss of generality, we have shifted our time indices as compared to the notation in §3.1. The probability distribution $p(x_{t+1} \mid y_{1:t})$ can now be seen as a *prior* and can be obtained by marginalizing out the previous state x_t as

$$\begin{aligned}p(x_{t+1} \mid y_{1:t}) &= \int p(x_{t+1}, x_t \mid y_{1:t}) \mathsf{d}x_t \\ &= \int p(x_{t+1} \mid x_t) p(x_t \mid y_{1:t}) \mathsf{d}x_t.\end{aligned} \tag{4.20}$$

Assuming that

$$p(x_{t+1} \mid x_t) \sim \mathcal{N}(x_{t+1}\,;\, f(x_t), Q), \tag{4.21a}$$
$$p(x_t \mid y_{1:t}) \sim \mathcal{N}(x_t\,;\, \hat{x}_t, P_{t|t}), \tag{4.21b}$$

the integral in (4.20) can be approximated as

$$p(x_{t+1} \mid y_{1:t}) \approx \mathcal{N}\left(x_{t+1}\,;\, f(\hat{x}_t), F_t P_{t|t} F_t^{\mathsf{T}} + G_t Q G_t^{\mathsf{T}}\right). \tag{4.22}$$

Here, $F_t = \frac{\partial f(x_t)}{\partial x_t}\Big|_{\substack{w_t=0\\ x_t=\hat{x}_t}}$, $G_t = \frac{\partial f(x_t)}{\partial w_t}\Big|_{\substack{w_t=0\\ x_t=\hat{x}_t}}$ and w_t is the process noise defined in (3.5). Using (4.22), the filtering problem (4.19) can again be written as a nonlinear least squares problem and can be solved using Gauss-Newton optimization as described in §4.1.1. Instead of optimizing over the whole data set at once as in §4.1, we solve a small optimization problem at each time instance t. This is closely related to what is often called an iterated extended Kalman filter [**?** **?**].

4.2.1 Filtering estimates of the orientation using optimization

In this section, we will derive an algorithm to obtain filtering estimates of the orientation using Gauss-Newton optimization. The resulting algorithm is closely related to the iterated multiplicative extended Kalman filter presented in [**?**].

To derive the approximation (4.22) for the orientation estimation problem, we first derive an expression for the function f that expresses η^{n}_{t+1} in terms of η^{n}_t. Using (4.12) in combination with (3.75a),

$$\begin{aligned}\eta^{\mathrm{n}}_{t+1} &= f(\eta^{\mathrm{n}}_t, y_{\omega,t}, e_{\omega,t}) \\ &= 2\log_{\mathsf{q}}\left(\exp_{\mathsf{q}}(\tfrac{\eta^{\mathrm{n}}_t}{2}) \odot \tilde{q}^{\mathrm{nb}}_t \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} + e_{\omega,t})\right) \odot \tilde{q}^{\mathrm{bn}}_{t+1}\right).\end{aligned} \tag{4.23}$$

Here, η^{n}_t and $\tilde{q}^{\mathrm{nb}}_t$ provide information about the previous time step. We denote the orientation estimate from the previous time step as $\hat{q}^{\mathrm{nb}}_t$ and assume that $\hat{\eta}^{\mathrm{n}}_t$ is zero since the linearization point is updated after each iteration of the optimization algorithm. To start with a fairly good linearization point, we choose $\tilde{q}^{\mathrm{nb},(0)}_{t+1}$ at iteration $k = 0$ as

$$\tilde{q}^{\mathrm{nb},(0)}_{t+1} = \hat{q}^{\mathrm{nb}}_t \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2} y_{\omega,t}\right). \tag{4.24}$$

Following (4.22), the distribution $p(\eta^{\mathrm{n}}_{t+1} \mid y_{1:t})$ around this linearization point can now be written as

$$p(\eta^{\mathrm{n}}_{t+1} \mid y_{1:t}) \approx \mathcal{N}\left(\eta^{\mathrm{n}}_{t+1}\,;\, 0, F_t P_{t|t} F_t^{\mathsf{T}} + G_t \Sigma_\omega G_t^{\mathsf{T}}\right), \tag{4.25}$$

with

$$\begin{aligned} F_t &= \tfrac{\partial f(\eta^{\mathrm{n}}_t, y_{\omega,t}, e_{\omega,t})}{\partial \eta^{\mathrm{n}}_t}\Big|_{\substack{e_{\omega,t}=0\\ \eta^{\mathrm{n}}_t=0,\, \tilde{q}^{\mathrm{nb}}_t=\hat{q}^{\mathrm{nb}}_t}} \\ &= 2\,\tfrac{\partial}{\partial \eta^{\mathrm{n}}_t}\log_{\mathsf{q}}\left(\exp_{\mathsf{q}}(\tfrac{\eta^{\mathrm{n}}_t}{2})\right)\Big|_{\eta^{\mathrm{n}}_t=0} \\ &= \mathcal{I}_3, \end{aligned} \tag{4.26a}$$

$$\begin{aligned} G_t &= \tfrac{\partial f(\eta^{\mathrm{n}}_t, y_{\omega,t}, e_{\omega,t})}{\partial e_{\omega,t}}\Big|_{\substack{e_{\omega,t}=0\\ \eta^{\mathrm{n}}_t=0,\, \tilde{q}^{\mathrm{nb}}_t=\hat{q}^{\mathrm{nb}}_t}} \\ &= 2\,\tfrac{\partial}{\partial e_{\omega,t}}\log_{\mathsf{q}}\left(\hat{q}^{\mathrm{nb}}_t \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} + e_{\omega,t})\right)\right. \\ &\qquad \left. \odot \exp_{\mathsf{q}}\left(-\tfrac{T}{2} y_{\omega,t}\right) \odot \hat{q}^{\mathrm{bn}}_t\right)\Big|_{e_{\omega,t}=0} \\ &\approx T \tilde{R}^{\mathrm{nb},(0)}_{t+1}. \end{aligned} \tag{4.26b}$$

The covariance $P_{t|t}$ in (4.25) can be approximated as the inverse of the Hessian of the objective function from the previous time step, see also §4.1.3. We make use of the shorthand notation $P_{t+1|t} = F_t P_{t|t} F_t^{\mathsf{T}} + G_t Q G_t^{\mathsf{T}}$ and define

$$e_{\mathrm{f},t} = \eta^{\mathrm{n}}_t - 2\log_{\mathsf{q}}\left(\hat{q}^{\mathrm{nb}}_{t-1} \odot \exp_{\mathsf{q}}(\tfrac{T}{2} y_{\omega,t-1}) \odot \tilde{q}^{\mathrm{bn}}_t\right), \qquad \tfrac{\partial e_{\mathrm{f},t}}{\partial \eta^{\mathrm{n}}_t} = \mathcal{I}_3. \tag{4.27}$$

Note that $e_{\mathrm{f},t}$ is equal to zero for iteration $k = 0$ but can be non-zero for subsequent iterations. Using this notation, the filtering problem (4.19) results in the following optimization problem

$$\begin{aligned}\hat{x}_t &= \arg\min_{x_t} \; -\log p(x_t \mid y_{1:t}) \\ &= \arg\min_{x_t} \; \underbrace{\|e_{\mathrm{f},t}\|_{P_{t|t-1}^{-1}}}_{\substack{\text{Dynamics and} \\ \text{knowledge about } x_{t-1}}} + \underbrace{\|e_{\mathrm{a},t}\|_{\Sigma_{\mathrm{a}}^{-1}} + \|e_{\mathrm{m},t}\|_{\Sigma_{\mathrm{m}}^{-1}}}_{\text{Measurement models}}. \end{aligned} \tag{4.28}$$

Note the similarity of this optimization problem to the smoothing formulation in (4.10). The term $\|e_{\mathrm{f},t}\|_{P_{t|t-1}^{-1}}$ takes into account both the knowledge about the previous state x_t and the dynamics. Furthermore, due to the fact that $P_{t|t-1}^{-1}$ is time-varying, the uncertainty and cross-correlation of the states at the previous time instance is taken into consideration. Including this term is similar to the inclusion of an arrival cost in moving horizon estimation approaches [**?**].

After each Gauss-Newton iteration, we need to recompute the linearization point as

$$\tilde{q}_t^{\mathrm{nb},(k+1)} = \exp_{\mathsf{q}}\left(\frac{\hat{\eta}_t^{\mathrm{n},(k+1)}}{2}\right) \odot \tilde{q}_t^{\mathrm{nb},(k)}. \tag{4.29}$$

We also need to compute the covariance around this updated linearization point as

$$P_{t|t-1}^{(k+1)} = J_t^{(k)} P_{t|t-1}^{(k)} \left(J_t^{(k)}\right)^{\mathsf{T}}, \tag{4.30}$$

where $J_t^{(k)}$ can be derived similarly to the derivation of F_t in (4.26a). Since $J_t^{(k)} = \mathcal{I}_3$, in practice the relinearization of the covariance can be omitted. The process of estimating orientation using filtering is summarized in Algorithm 2.

4.3 Extended Kalman filtering

Instead of using optimization for position and orientation estimation, it is also possible to use extended Kalman filtering. Extended Kalman filters (EKFs) compute filtering estimates in terms of the conditional probability distribution (3.2). Hence, the approach is similar to the one discussed in §4.2. In fact, extended Kalman filtering can be interpreted as Gauss-Newton optimization of the filtering problem using only one iteration (4.9) with a step length of one, see *e.g.* [**?**]. In this section we first introduce how an EKF can be used to compute state estimates in a general nonlinear state space model. Subsequently, we illustrate the use of EKFs for position and orientation estimation by focusing on the orientation estimation problem. Two different implementations will be discussed. First, we introduce an EKF implementation that uses unit quaternions as states. Subsequently, we discuss an EKF which parametrizes the orientation in terms of an orientation deviation from a linearization point, similar to the approach used in §4.1 and §4.2.

An EKF makes use of a nonlinear state space model. Assuming that the measurement noise is additive and that both the process and the measurement noise are zero-mean Gaussian with constant covariance, the state space model is given by

$$x_{t+1} = f_t(x_t, u_t, w_t), \tag{4.34a}$$
$$y_t = h_t(x_t) + e_t, \tag{4.34b}$$

with $w_t \sim \mathcal{N}(0, Q)$ and $e_t \sim \mathcal{N}(0, R)$.

The state is estimated recursively by performing a *time update* and a *measurement update*. The time update uses the model (4.34a) to "predict" the state to the next time step according to

$$\hat{x}_{t+1|t} = f_t(\hat{x}_{t|t}, u_t), \tag{4.35a}$$
$$P_{t+1|t} = F_t P_{t|t} F_t^{\mathsf{T}} + G_t Q G_t^{\mathsf{T}}, \tag{4.35b}$$

with

$$F_t = \left.\frac{\partial f_t(x_t, u_t, w_t)}{\partial x_t}\right|_{\substack{w_t = 0 \\ x_t = \hat{x}_{t|t}}}, \qquad G_t = \left.\frac{\partial f_t(x_t, u_t, w_t)}{\partial v_t}\right|_{\substack{w_t = 0 \\ x_t = \hat{x}_{t|t}}}. \tag{4.36}$$

Algorithm 2 Filtering estimates of the orientation using optimization

INPUTS: Inertial data $\{y_{\mathrm{a},t}, y_{\omega,t}\}_{t=1}^{N}$, magnetometer data $\{y_{\mathrm{m},t}\}_{t=1}^{N}$ and covariance matrices Σ_ω, Σ_a and Σ_m.
OUTPUTS: An estimate of the orientation $\hat{q}_t^{\mathrm{nb}}$ and its covariance $P_{t|t}$ for $t = 1, \dots N$.

1. Compute $\breve{q}_1^{\mathrm{nb}}$ and Σ_i as described in §3.6 and set $\hat{q}_1^{\mathrm{nb}} = \breve{q}_1^{\mathrm{nb}}$ and $P_{1|1} = \Sigma_{\eta,\mathrm{i}}$.
2. **for** $t = 2, \dots N$ **do**
   (a) Set $\hat{\eta}_t^{\mathrm{n},(0)} = 0_{3\times 1}$, set $k = 0$, choose the linearization point $\tilde{q}_t^{\mathrm{nb},(0)}$ as
   $$\tilde{q}_t^{\mathrm{nb},(0)} = \hat{q}_{t-1}^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2} y_{\omega,t-1}\right), \tag{4.31a}$$
   and compute $P_{t|t-1}$ as
   $$P_{t|t-1} = P_{t-1|t-1} + G_{t-1}\Sigma_\omega G_{t-1}^{\mathsf{T}}, \tag{4.31b}$$
   with $G_{t-1} = T\tilde{R}_t^{\mathrm{nb},(0)}$.
   (b) **while** *termination condition is not satisfied* **do**
      i. Compute the gradient (4.6a) and the approximate Hessian (4.8) of the filtering problem (4.28) using the expressions for the different parts of the cost function and their Jacobians (4.27), (4.13c), (4.13d), (4.14d) and (4.14e).
      ii. Apply the update (4.9) to obtain $\hat{\eta}_t^{\mathrm{n},(k+1)}$.
      iii. Update the linearization point as
      $$\tilde{q}_t^{\mathrm{nb},(k+1)} = \exp_{\mathsf{q}}\left(\frac{\hat{\eta}_t^{\mathrm{n},(k+1)}}{2}\right) \odot \tilde{q}_t^{\mathrm{nb},(k)}, \tag{4.32}$$
      $\hat{\eta}_t^{\mathrm{n},(k+1)} = 0_{3\times 1}$.
      iv. Set $k = k + 1$.
   end while
   (c) Set $\hat{q}_t^{\mathrm{nb}} = \tilde{q}_t^{\mathrm{nb},(k)}$, $\hat{\eta}_t^{\mathrm{n}} = \hat{\eta}_t^{\mathrm{n},(k)}$ and compute $P_{t|t}$ as
   $$P_{t|t} = \left(\mathcal{J}^{\mathsf{T}}(\hat{\eta}_t^{\mathrm{n}})\mathcal{J}(\hat{\eta}_t^{\mathrm{n}})\right)^{-1}. \tag{4.33}$$

Here, the matrix P denotes the state covariance. The double subscripts on $\hat{x}_{t+1|t}$ and $P_{t+1|t}$ denote the state estimate and the state covariance at time $t+1$ given measurements up to time t. Similarly, $\hat{x}_{t|t}$ and $P_{t|t}$ denote the state estimate and the state covariance at time t given measurements up to time t.

The measurement update makes use of the measurement model (4.34b) in combination with the measurements y_t to update the "predicted" state estimate as

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t\varepsilon_t, \tag{4.37a}$$
$$P_{t|t} = P_{t|t-1} - K_t S_t K_t^{\mathsf{T}}, \tag{4.37b}$$

with

$$\varepsilon_t \triangleq y_t - \hat{y}_{t|t-1}, \quad S_t \triangleq H_t P_{t|t-1} H_t^{\mathsf{T}} + R, \quad K_t \triangleq P_{t|t-1} H_t^{\mathsf{T}} S_t^{-1}, \tag{4.38}$$

and

$$\hat{y}_{t|t-1} = h(\hat{x}_{t|t-1}), \qquad H_t = \left.\frac{\partial h_t(x_t)}{\partial x_t}\right|_{x_t=\hat{x}_{t|t-1}}. \tag{4.39}$$

Note that in (4.37) we have shifted our notation one time step compared to the notation in (4.35) to avoid cluttering the notation. The EKF iteratively performs a time update and a measurement update to estimate the state and its covariance.

4.3.1 Estimating orientation using quaternions as states

In this section, we will illustrate the use of an EKF to compute filtering estimates of the orientation. As discussed in the previous section, the crucial part is to compute the matrices F_t, G_t and H_t to perform

the EKF time and measurement updates. Using the state space model (3.75) and using unit quaternions as states in the EKF, the dynamic model is given by

$$\begin{aligned} q_{t+1}^{\mathrm{nb}} &= f_t(q_t^{\mathrm{nb}}, y_{\omega,t}, e_{\omega,t}) = q_t^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} - e_{\omega,t})\right) \\ &= \left(\exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} - e_{\omega,t})\right)\right)^{\mathsf{R}} q_t^{\mathrm{nb}} \\ &= \left(q_t^{\mathrm{nb}}\right)^{\mathsf{L}} \exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} - e_{\omega,t})\right). \end{aligned} \tag{4.40}$$

The following derivatives of the dynamic model (4.40) can be obtained

$$F_t = \left.\frac{\partial f_t(q_t^{\mathrm{nb}}, y_{\omega,t}, e_{\omega,t})}{\partial q_t^{\mathrm{nb}}}\right|_{\substack{e_{\omega,t}=0 \\ q_t^{\mathrm{nb}}=\hat{q}_{t|t}^{\mathrm{nb}}}} = \left(\exp_{\mathsf{q}}(\tfrac{T}{2} y_{\omega,t})\right)^{\mathsf{R}}, \tag{4.41a}$$

$$\begin{aligned} G_t &= \left.\frac{\partial f_t(q_t^{\mathrm{nb}}, y_{\omega,t}, e_{\omega,t})}{\partial e_{\omega,t}}\right|_{\substack{e_{\omega,t}=0 \\ q_t^{\mathrm{nb}}=\hat{q}_{t|t}^{\mathrm{nb}}}} \\ &= \frac{\partial}{\partial e_{\omega,t}} \left(q_t^{\mathrm{nb}}\right)^{\mathsf{L}} \exp_{\mathsf{q}}(\tfrac{T}{2}(y_{\omega,t} - e_{\omega,t}))\Big|_{\substack{e_{\omega,t}=0 \\ q_t^{\mathrm{nb}}=\hat{q}_{t|t}^{\mathrm{nb}}}} \\ &= -\tfrac{T}{2}\left(\hat{q}_{t|t}^{\mathrm{nb}}\right)^{\mathsf{L}} \frac{\partial \exp_{\mathsf{q}}(e_{\omega,t})}{\partial e_{\omega,t}}. \end{aligned} \tag{4.41b}$$

In the measurement update of the EKF, the state is updated using the accelerometer and magnetometer measurements. Using the measurement models

$$y_{\mathrm{a},t} = -R_t^{\mathrm{bn}} g^{\mathrm{n}} + e_{\mathrm{a},t}, \qquad y_{\mathrm{m},t} = R_t^{\mathrm{bn}} m^{\mathrm{n}} + e_{\mathrm{m},t}, \tag{4.42}$$

the matrix H_t in (4.39) is given by

$$H_t = \frac{\partial}{\partial q_t^{\mathrm{nb}}} \begin{pmatrix} R_t^{\mathrm{bn}} g^{\mathrm{n}} \\ R_t^{\mathrm{bn}} m^{\mathrm{n}} \end{pmatrix}\Bigg|_{q_t^{\mathrm{nb}}=\hat{q}_{t|t-1}^{\mathrm{nb}}} = \begin{pmatrix} -\left.\frac{\partial R_{t|t-1}^{\mathrm{bn}}}{\partial q_{t|t-1}^{\mathrm{nb}}}\right|_{q_{t|t-1}^{\mathrm{nb}}=\hat{q}_{t|t-1}^{\mathrm{nb}}} g^{\mathrm{n}} \\ \left.\frac{\partial R_{t|t-1}^{\mathrm{bn}}}{\partial q_{t|t-1}^{\mathrm{nb}}}\right|_{q_{t|t-1}^{\mathrm{nb}}=\hat{q}_{t|t-1}^{\mathrm{nb}}} m^{\mathrm{n}} \end{pmatrix}. \tag{4.43}$$

The derivative $\frac{\partial R_{t|t-1}^{\mathrm{bn}}}{\partial q_{t|t-1}^{\mathrm{nb}}}$ can be computed from the definition of the relation between and the rotation matrix and the quaternion representation given in (A.10).

Note that the quaternion obtained from the measurement update (4.37) is no longer normalized. We denote this unnormalized quaternion and its covariance by $\tilde{q}_{t|t}^{\mathrm{nb}}$ and $\tilde{P}_{t|t}$, respectively. The $\tilde{\cdot}$ instead of the $\hat{\cdot}$ is meant to explicitly indicate that the quaternion still needs to be updated. This is done by an additional renormalization step as compared to a standard EKF implementation. A possible interpretation of the renormalization is as an additional measurement update without measurement noise. Hence, we adjust both the quaternion and its covariance as

$$\hat{q}_{t|t}^{\mathrm{nb}} = \frac{\tilde{q}_{t|t}^{\mathrm{nb}}}{\|\tilde{q}_{t|t}^{\mathrm{nb}}\|_2}, \qquad P_{t|t} = J_t \tilde{P}_{t|t} J_t^{\mathsf{T}}, \tag{4.44a}$$

with

$$J_t = \frac{1}{\|\tilde{q}_{t|t}^{\mathrm{nb}}\|_2^3} \tilde{q}_{t|t}^{\mathrm{nb}} \left(\tilde{q}_{t|t}^{\mathrm{nb}}\right)^{\mathsf{T}}. \tag{4.44b}$$

Here, q^{T} is the standard vector transpose. The resulting EKF is summarized in Algorithm 3.

4.3.2 Estimating orientation using orientation deviations as states

An alternative EKF implementation parametrizes the orientation in terms of an orientation deviation around a linearization point. The linearization point is parametrized in terms of quaternions or rotation matrices and denoted $\tilde{q}_t^{\mathrm{nb}}$ or equivalently $\tilde{R}_t^{\mathrm{nb}}$. The orientation deviation η_t^{n} is the state vector in the EKF. This EKF implementation is sometimes referred to as a multiplicative EKF [**?** **?**]. One of its advantages is that its implementation is computationally attractive since it only uses a 3-dimensional state compared to the 4-dimensional state in Algorithm 3.

Algorithm 3 Orientation estimation using an EKF with quaternion states

INPUTS: Inertial data $\{y_{\mathrm{a},t}, y_{\omega,t}\}_{t=1}^{N}$, magnetometer data $\{y_{\mathrm{m},t}\}_{t=1}^{N}$ and covariance matrices Σ_ω, Σ_a and Σ_m.
OUTPUTS: An estimate of the orientation $\hat{q}_{t|t}^{\mathrm{nb}}$ and its covariance $P_{t|t}$ for $t = 1, \dots N$.

1. Compute $\breve{q}_1^{\mathrm{nb}}$ and Σ_i as described in §3.6 and set $\hat{q}_{1|1}^{\mathrm{nb}} = \breve{q}_1^{\mathrm{nb}}$ and $P_{1|1} = \Sigma_{\mathrm{q,i}}$.
2. **For** $t = 2, \dots, N$ **do**

   (a) Time update

$$\hat{q}_{t|t-1}^{\mathrm{nb}} = \hat{q}_{t-1|t-1}^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2} y_{\omega,t-1}\right), \tag{4.45a}$$
$$P_{t|t-1} = F_{t-1} P_{t-1|t-1} F_{t-1}^{\mathsf{T}} + G_{t-1} Q G_{t-1}^{\mathsf{T}}, \tag{4.45b}$$

   with $Q = \Sigma_\omega$ and

$$F_{t-1} = \left(\exp_{\mathsf{q}}(\tfrac{T}{2} y_{\omega,t-1})\right)^{\mathsf{R}}, \; G_{t-1} = -\tfrac{T}{2} \left(\hat{q}_{t-1|t-1}^{\mathrm{nb}}\right)^{\mathsf{L}} \tfrac{\partial \exp_{\mathsf{q}}(e_{\omega,t-1})}{\partial e_{\omega,t-1}}.$$

   (b) Measurement update

$$\tilde{q}_{t|t}^{\mathrm{nb}} = \hat{q}_{t|t-1}^{\mathrm{nb}} + K_t \varepsilon_t, \tag{4.46a}$$
$$\tilde{P}_{t|t} = P_{t|t-1} - K_t S_t K_t^{\mathsf{T}}, \tag{4.46b}$$

   with ε_t, K_t and S_t defined in (4.38) and

$$y_t = \begin{pmatrix} y_{\mathrm{a},t} \\ y_{\mathrm{m},t} \end{pmatrix}, \qquad \hat{y}_{t|t-1} = \begin{pmatrix} -\hat{R}_{t|t-1}^{\mathrm{bn}} g^{\mathrm{n}} \\ \hat{R}_{t|t-1}^{\mathrm{bn}} m^{\mathrm{n}} \end{pmatrix},$$
$$H_t = \begin{pmatrix} -\left.\frac{\partial R_{t|t-1}^{\mathrm{bn}}}{\partial q_{t|t-1}^{\mathrm{nb}}}\right|_{q_{t|t-1}^{\mathrm{nb}} = \hat{q}_{t|t-1}^{\mathrm{nb}}} g^{\mathrm{n}} \\ \left.\frac{\partial R_{t|t-1}^{\mathrm{bn}}}{\partial q_{t|t-1}^{\mathrm{nb}}}\right|_{q_{t|t-1}^{\mathrm{nb}} = \hat{q}_{t|t-1}^{\mathrm{nb}}} m^{\mathrm{n}} \end{pmatrix}, \qquad R = \begin{pmatrix} \Sigma_\mathrm{a} & 0 \\ 0 & \Sigma_\mathrm{m} \end{pmatrix}.$$

   (c) Renormalize the quaternion and its covariance as

$$\hat{q}_{t|t}^{\mathrm{nb}} = \frac{\tilde{q}_{t|t}^{\mathrm{nb}}}{\|\tilde{q}_{t|t}^{\mathrm{nb}}\|_2}, \qquad P_{t|t} = J_t \tilde{P}_{t|t} J_t^{\mathsf{T}}, \tag{4.47}$$

   with $J_t = \frac{1}{\|\tilde{q}_{t|t}^{\mathrm{nb}}\|_2^3} \tilde{q}_{t|t}^{\mathrm{nb}} \left(\tilde{q}_{t|t}^{\mathrm{nb}}\right)^{\mathsf{T}}$.

end for

Similarly to (4.24) in §4.2, the dynamics of the state η^{n} is given by

$$\begin{aligned} \eta_{t+1}^{\mathrm{n}} &= f_t\left(\eta_t^{\mathrm{n}}, y_{\omega,t}, e_{\omega,t}\right) \\ &= 2\log\left(\exp_{\mathsf{q}}(\tfrac{\eta_t^{\mathrm{n}}}{2}) \odot \tilde{q}_t^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} - e_{\omega,t})\right) \odot \tilde{q}_{t+1}^{\mathrm{bn}}\right). \end{aligned} \tag{4.48}$$

In the time update of the EKF, we first use (3.75a) to directly update the linearization point as

$$\tilde{q}_{t+1|t}^{\mathrm{nb}} = \tilde{q}_{t|t}^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2} y_{\omega,t}\right). \tag{4.49}$$

Around this linearization point, the derivatives of the dynamic model (4.48) are given by

$$\begin{aligned} F_t &= \left.\frac{\partial f_t(\eta_t^{\mathrm{n}}, y_{\omega,t}, e_{\omega,t})}{\partial \eta_t^{\mathrm{n}}}\right|_{\substack{e_{\omega,t}=0 \\ \eta_t^{\mathrm{n}}=0}} = \mathcal{I}_3, \\ G_t &= \left.\frac{\partial f_t(\eta_t^{\mathrm{n}}, y_{\omega,t}, e_{\omega,t})}{\partial e_{\omega,t}}\right|_{\substack{e_{\omega,t}=0 \\ \eta_t^{\mathrm{n}}=0}} = T \tilde{R}_{t+1|t}^{\mathrm{nb}}. \end{aligned} \tag{4.50a}$$

Note the similarity with (4.26).

In the measurement update of the EKF, the state η_t^{n} is updated using the accelerometer and magnetometer measurements. The accelerometer measurement equation can be written in terms of the state

η_t^{n} as

$$\begin{aligned} y_{\mathrm{a},t} &= -R_t^{\mathrm{bn}} g^{\mathrm{n}} + e_{\mathrm{a},t} \\ &= -\tilde{R}_t^{\mathrm{bn}} \left(\exp_{\mathsf{R}}([\eta_t^{\mathrm{n}}\times])\right)^{\mathsf{T}} g^{\mathrm{n}} + e_{\mathrm{a},t} \\ &\approx -\tilde{R}_t^{\mathrm{bn}} \left(\mathcal{I}_3 - [\eta_t^{\mathrm{n}}\times]\right) g^{\mathrm{n}} + e_{\mathrm{a},t} \\ &= -\tilde{R}_t^{\mathrm{bn}} g^{\mathrm{n}} - \tilde{R}_t^{\mathrm{bn}} [g^{\mathrm{n}}\times] \eta_t^{\mathrm{n}} + e_{\mathrm{a},t}. \end{aligned} \tag{4.51}$$

Equivalently, the magnetometer measurement equation can be written in terms of the state η_t^{n} as

$$\begin{aligned} y_{\mathrm{m},t} &= R_t^{\mathrm{bn}} m^{\mathrm{n}} + e_{\mathrm{m},t} \\ &= \tilde{R}_t^{\mathrm{bn}} \left(\exp_{\mathsf{R}}([\eta_t^{\mathrm{n}}\times])\right)^{\mathsf{T}} m^{\mathrm{n}} + e_{\mathrm{m},t} \\ &\approx \tilde{R}_t^{\mathrm{bn}} \left(\mathcal{I}_3 - [\eta_t^{\mathrm{n}}\times]\right) m^{\mathrm{n}} + e_{\mathrm{m},t} \\ &= \tilde{R}_t^{\mathrm{bn}} m^{\mathrm{n}} + \tilde{R}_t^{\mathrm{bn}} [m^{\mathrm{n}}\times] \eta_t^{\mathrm{n}} + e_{\mathrm{m},t}. \end{aligned} \tag{4.52}$$

From these equations, the derivatives H_t as defined in (4.39) can straightforwardly be computed.

After the measurement update, the orientation deviation $\hat{\eta}_t^{\mathrm{n}}$ is non-zero. Hence, as an additional step in the EKF, we update the linearization point and reset the state. In our algorithm, we consider the relinearization as the "measurement update" for the linearization point, *i.e.* the relinearization updates the linearization point $\tilde{q}_{t|t-1}^{\mathrm{nb}}$ to $\tilde{q}_{t|t}^{\mathrm{nb}}$ as

$$\tilde{q}_{t|t}^{\mathrm{nb}} = \exp_{\mathsf{q}}\left(\tfrac{\hat{\eta}_t^{\mathrm{n}}}{2}\right) \odot \tilde{q}_{t|t-1}^{\mathrm{nb}}. \tag{4.53}$$

For the same reasons as in §4.2, the relinearization of the covariance can be omitted. The resulting EKF is summarized in Algorithm 4. Note the similarities between this algorithm and Algorithm 2.

Algorithm 4 Orientation estimation using an EKF with orientation deviation states

INPUTS: Inertial data $\{y_{\mathrm{a},t}, y_{\omega,t}\}_{t=1}^N$, magnetometer data $\{y_{\mathrm{m},t}\}_{t=1}^N$ and covariance matrices Σ_ω, Σ_{a} and Σ_{m}.
OUTPUTS: An estimate of the orientation $\tilde{q}_{t|t}^{\mathrm{nb}}$ and the covariance $P_{t|t}$ for $t = 1, \ldots N$.

1. Compute $\breve{q}_1^{\mathrm{nb}}$ and Σ_{i} as described in §3.6 and set $\tilde{q}_{1|1}^{\mathrm{nb}} = \breve{q}_1^{\mathrm{nb}}$ and $P_{1|1} = \Sigma_{\eta,\mathrm{i}}$.
2. **For** $t = 2, \ldots, N$ **do**

   (a) Time update

   $$\tilde{q}_{t|t-1}^{\mathrm{nb}} = \tilde{q}_{t-1|t-1}^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2} y_{\omega,t-1}\right), \tag{4.54a}$$
   $$P_{t|t-1} = P_{t-1|t-1} + G_{t-1} Q G_{t-1}^{\mathsf{T}}, \tag{4.54b}$$

   with $G_{t-1} = T\tilde{R}_{t|t-1}^{\mathrm{nb}}$ and $Q = \Sigma_\omega$.

   (b) Measurement update

   $$\hat{\eta}_t^{\mathrm{n}} = K_t \varepsilon_t, \tag{4.55a}$$
   $$\tilde{P}_{t|t} = P_{t|t-1} - K_t S_t K_t^{\mathsf{T}}, \tag{4.55b}$$

   with ε_t, K_t and S_t defined in (4.38) and

   $$y_t = \begin{pmatrix} y_{\mathrm{a},t} \\ y_{\mathrm{m},t} \end{pmatrix}, \qquad \hat{y}_{t|t-1} = \begin{pmatrix} -\tilde{R}_{t|t-1}^{\mathrm{bn}} g^{\mathrm{n}} \\ \tilde{R}_{t|t-1}^{\mathrm{bn}} m^{\mathrm{n}} \end{pmatrix},$$
   $$H_t = \begin{pmatrix} -\tilde{R}_{t|t-1}^{\mathrm{bn}} [g^{\mathrm{n}}\times] \\ \tilde{R}_{t|t-1}^{\mathrm{bn}} [m^{\mathrm{n}}\times] \end{pmatrix}, \qquad R = \begin{pmatrix} \Sigma_{\mathrm{a}} & 0 \\ 0 & \Sigma_{\mathrm{m}} \end{pmatrix}.$$

   (c) Relinearize

   $$\tilde{q}_{t|t}^{\mathrm{nb}} = \exp_{\mathsf{q}}\left(\tfrac{\hat{\eta}_t^{\mathrm{n}}}{2}\right) \odot \tilde{q}_{t|t-1}^{\mathrm{nb}}. \tag{4.56}$$

   end for

4.4 Complementary filtering

An alternative to using EKFs for orientation estimation is to use complementary filters [**?** **?**]. This type of algorithms again estimates the orientation at time t given the measurements $y_{1:t}$. However, it does not use the probabilistic models presented in Chapter 3. Instead, complementary filters explicitly use the fact that both the gyroscope and the combination of the accelerometer and the magnetometer provide information about the orientation of the sensor. The orientation estimates obtained from the accelerometer and the magnetometer measurements are noisy but accurate over long periods of time. On the other hand, the orientation estimates using the gyroscope measurements are accurate on a short time scale but they drift over time. These properties can be interpreted and exploited in the frequency domain. The orientation estimates using the gyroscope have desirable properties at high frequencies. We would therefore like to filter these using a high-pass filter. Conversely, the orientation estimates obtained from the accelerometer and magnetometer measurements have desirable properties at low frequencies. We would therefore like to filter these orientation estimates using a low-pass filter.

This can be illustrated by considering the one-dimensional case of estimating an angle θ from gyroscope measurements y_ω and magnetometer measurements y_m. We denote the angle estimated by the complementary filter by $\hat{\theta}$, the angle obtained from the magnetometer measurements by θ_m and the angle obtained from the gyroscope measurements by θ_ω. Note that the latter is obtained by integrating the gyroscope measurements. The Laplace transforms of θ, θ_m, θ_ω and y_ω are denoted by $\Theta(s)$, $\Theta_\mathrm{m}(s)$, $\Theta_\omega(s)$ and $Y_\omega(s)$, respectively. The complementary filter computes $\Theta(s)$ as

$$\begin{aligned}\Theta(s) &= G(s)\Theta_\mathrm{m}(s) + (1 - G(s))\,\Theta_\omega(s) \\ &= G(s)\Theta_\mathrm{m}(s) + (1 - G(s))\,\tfrac{1}{s}Y_\omega(s),\end{aligned} \tag{4.57}$$

where $G(s)$ is a low-pass filter and $1 - G(s)$ is hence a high-pass filter. Note that the sum of the two filters, $G(s)$ and $1 - G(s)$, is equal to one, which is the reason behind the name *complementary* filter. Choosing $G(s)$ as a first-order low-pass filter $G(s) = \frac{1}{as+1}$ and using Euler backward discretization, the filter (4.57) can be written in discrete time as

$$\hat{\theta}_t = (1 - \gamma)\theta_{\mathrm{m},t} + \gamma\left(\hat{\theta}_{t-1} + Ty_{\omega,t}\right), \tag{4.58}$$

where $\gamma = \frac{a}{T+a}$. The relation (4.58) allows us to recursively estimate the angle $\hat{\theta}$. The only parameter that needs to be chosen to run the complementary filter (4.58) is the parameter a of the low-pass filter $G(s)$. Choosing a large a (and hence a γ close to one) results in a lower cut-off frequency and a more significant contribution of the gyroscope measurements on the orientation estimates. Conversely, a small a (and hence a γ close to zero) results in a higher cut-off frequency and a more significant contribution of the magnetometer measurements on the orientation estimates.

There is a strong relationship between complementary and Kalman filtering for linear models, see *e.g.* [**?** **?**]. To highlight the similarities and differences between complementary and extended Kalman filtering for orientation estimation, let us define the estimate from the complementary filter as $\hat{\theta}_{t|t}$ and write the recursion (4.58) equivalently as

$$\hat{\theta}_{t|t-1} = \hat{\theta}_{t-1} + Ty_{\omega,t}, \tag{4.59a}$$

$$\hat{\theta}_{t|t} = (1 - \gamma)\theta_{\mathrm{m},t} + \gamma\hat{\theta}_{t|t-1}. \tag{4.59b}$$

So far, we have discussed the one-dimensional case of estimating an angle θ. Let us now focus on estimating orientation in three dimensions and parametrize this as a unit quaternion q^nb. Two well-known complementary filter implementations to estimate the orientation q^nb are presented in [**?**] and [**?**]. Open-source implementations of both algorithms are available online [**?**]. The filter presented in [**?**] is specifically designed to be computationally efficient and has for instance been implemented in the robot operating system (ROS) [**?**]. In this section, we will derive a complementary filter that is inspired by this implementation but which is also meant to illustrate the similarities and differences between complementary filters and extended Kalman filters for orientation estimation. Because of that, the complementary filter that we will present is not as computationally efficient as the one in [**?**].

Similar to the one-dimensional case (4.59a), the orientation estimate of the complementary filter can be updated using the gyroscope measurements as

$$\hat{q}^\mathrm{nb}_{t|t-1} = \hat{q}^\mathrm{nb}_{t-1|t-1} \odot \exp_\mathsf{q}\left(\tfrac{T}{2}y_{\omega,t}\right), \tag{4.60}$$

where $\hat{q}^{\mathrm{nb}}$ is the orientation estimate from the complementary filter and the double subscripts are defined analogously to (4.59). Note that the update (4.60) is equivalent to the time update in the EKF in Algorithm 3.

The orientation from the accelerometer and magnetometer measurements, denoted $R^{\mathrm{nb}}_{\mathrm{am},t}$, can be obtained by solving

$$\begin{aligned} q^{\mathrm{nb}}_{\mathrm{am},t} = \underset{q^{\mathrm{nb}}_{\mathrm{am},t}}{\arg\min} \quad & \|\bar{y}_{\mathrm{a},t} + q^{\mathrm{bn}}_{\mathrm{am},t} \odot \bar{g}^{\mathrm{n}} \odot q^{\mathrm{nb}}_{\mathrm{am},t}\|^2_{\Sigma_{\mathrm{a}}^{-1}} + \\ & \qquad \|\bar{y}_{\mathrm{m},t} - q^{\mathrm{bn}}_{\mathrm{am},t} \odot \bar{m}^{\mathrm{n}} \odot q^{\mathrm{nb}}_{\mathrm{am},t}\|^2_{\Sigma_{\mathrm{m}}^{-1}} \\ \text{subj. to} \quad & \|q^{\mathrm{nb}}_{\mathrm{am},t}\|_2 = 1, \end{aligned} \tag{4.61}$$

see also §3.6. Similar to [**?**], we do not completely solve the optimization problem (4.61) in each recursion of the complementary filter. Instead, we use an initial estimate $R^{\mathrm{nb},(0)}_{\mathrm{am},t} = \hat{q}^{\mathrm{nb}}_{t|t-1}$ and only perform one iteration of an optimization algorithm. In [**?**], one iteration of a gradient descent algorithm is performed. We instead perform one Gauss-Newton update as in (4.9). Hence, using

$$\varepsilon_t = \begin{pmatrix} \left(y_{\mathrm{a},t} + \hat{R}^{\mathrm{bn}}_{t|t-1} g^{\mathrm{n}}\right) \Sigma_{\mathrm{a}}^{-1/2} \\ \left(y_{\mathrm{m},t} - \hat{R}^{\mathrm{bn}}_{t|t-1} m^{\mathrm{n}}\right) \Sigma_{\mathrm{m}}^{-1/2} \end{pmatrix}, \tag{4.62a}$$

$$\mathcal{J}_t = \frac{\partial \varepsilon_t}{\partial q^{\mathrm{nb}}_{\mathrm{am},t}} = \begin{pmatrix} \left.\frac{\partial R^{\mathrm{bn}}_{\mathrm{am},t}}{\partial q^{\mathrm{nb}}_{\mathrm{am},t}}\right|_{q^{\mathrm{nb}}_{\mathrm{am},t}=\hat{q}^{\mathrm{nb}}_{t|t-1}} g^{\mathrm{n}} \, \Sigma_{\mathrm{a}}^{-1/2} \\ -\left.\frac{\partial R^{\mathrm{bn}}_{\mathrm{am},t}}{\partial q^{\mathrm{nb}}_{\mathrm{am},t}}\right|_{q^{\mathrm{nb}}_{\mathrm{am},t}=\hat{q}^{\mathrm{nb}}_{t|t-1}} m^{\mathrm{n}} \, \Sigma_{\mathrm{m}}^{-1/2} \end{pmatrix}, \tag{4.62b}$$

the estimate $q^{\mathrm{nb}}_{\mathrm{am},t}$ is given by

$$q^{\mathrm{nb}}_{\mathrm{am},t} = \hat{q}^{\mathrm{nb}}_{t|t-1} - \beta \left(\mathcal{J}_t^{\mathsf{T}} \mathcal{J}_t\right)^{-1} \mathcal{J}_t^{\mathsf{T}} \varepsilon_t. \tag{4.63}$$

Analogously to (4.59b), the second update of the complementary filter is therefore given by

$$\begin{aligned} \hat{q}^{\mathrm{nb}}_{t|t} &= (1-\gamma)\left(\hat{q}^{\mathrm{nb}}_{t|t-1} - \beta \left(\mathcal{J}_t^{\mathsf{T}} \mathcal{J}_t\right)^{-1} \mathcal{J}_t^{\mathsf{T}} \varepsilon_t\right) + \gamma \hat{q}^{\mathrm{nb}}_{t|t-1} \\ &= \hat{q}^{\mathrm{nb}}_{t|t-1} - (1-\gamma)\beta \left(\mathcal{J}_t^{\mathsf{T}} \mathcal{J}_t\right)^{-1} \mathcal{J}_t^{\mathsf{T}} \varepsilon_t. \end{aligned} \tag{4.64}$$

There are many similarities between (4.64) and the measurement update in the EKF in Algorithm 3. First of all, if we neglect the presence of Σ_{a} and Σ_{m} for a moment, $\mathcal{J}_t$ and ε_t in (4.62) are equal to $-H_t$ and ε_t in the measurement update in Algorithm 3. Inclusion of Σ_{a} and Σ_{m} in (4.62) ensures that the uncertainty of the accelerometer and magnetometer measurements is properly taken into account when obtaining $q^{\mathrm{nb}}_{\mathrm{am},t}$. The contributions to the orientation estimate $\hat{q}^{\mathrm{nb}}_{t|t}$ from integration of the gyroscope measurements and from $q^{\mathrm{nb}}_{\mathrm{am},t}$ are determined by the factor $(1-\gamma)\beta$. In the EKF on the other hand, the uncertainty of the gyroscope, accelerometer and magnetometer measurements is taken into account through the covariance matrices Σ_ω, Σ_{a} and Σ_{m}.

Second of all, comparing (4.64) to the EKF measurement update (4.37), the update (4.64) can be interpreted as a scaled version of an EKF update with $P_{t|t-1} = \mathcal{I}_4$ and $R = 0_{3\times 3}$. In that case, the matrix K_t in the EKF would be given by $K_t = H_t^{\mathsf{T}} \left(H_t H_t^{\mathsf{T}}\right)^{-1}$. The matrix $H_t H_t^{\mathsf{T}}$ is rank deficient. Because of this, to compute K_t, we would have to use the pseudo-inverse, see *e.g.* [**?**]. Denoting the pseudo-inverse of a matrix as $\cdot^{\dagger}$, $K_t = H_t^{\mathsf{T}} \left(H_t H_t^{\mathsf{T}}\right)^{\dagger} = \left(H_t^{\mathsf{T}} H_t\right)^{-1} H_t^{\mathsf{T}}$. An important difference between the EKF and complementary update is that the scaling factor $(1-\gamma)\beta$ in (4.64) is constant, while the matrix $P_{t|t-1}$ in the EKF is time-varying. In the remainder, we will denote this constant $(1-\gamma)\beta$ by α for notational brevity.

Our complementary filter implementation is summarized in Algorithm 5. Note again the similarity to Algorithm 3. To stress this similarity even more, we call the update (4.60) the time update of the complementary filter and (4.64) the measurement update.

4.5 Evaluation based on experimental and simulated data

In this section, we apply the algorithms described in §4.1–§4.4 to both simulated and experimental data. Some general characteristics of the orientation estimation algorithms will be illustrated and the quality

Algorithm 5 Orientation estimation using a complementary filter

INPUTS: Inertial data $\{y_{\mathrm{a},t}, y_{\omega,t}\}_{t=1}^{N}$, magnetometer data $\{y_{\mathrm{m},t}\}_{t=1}^{N}$ and covariance matrices Σ_{a} and Σ_{m}.
OUTPUTS: An estimate of the orientation $\hat{q}_{t|t}^{\mathrm{nb}}$ for $t = 1, \dots N$.

1. Compute $\breve{q}_1^{\mathrm{nb}}$ as described in §3.6 and set $\hat{q}_{1|1}^{\mathrm{nb}} = \breve{q}_1^{\mathrm{nb}}$.
2. **For** $t = 2, \dots, N$ **do**

   (a) Time update

   $$\hat{q}_{t|t-1}^{\mathrm{nb}} = \hat{q}_{t-1|t-1}^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2} y_{\omega,t-1}\right), \tag{4.65}$$

   (b) Measurement update

   $$\tilde{q}_{t|t}^{\mathrm{nb}} = \hat{q}_{t|t-1}^{\mathrm{nb}} - \alpha \left(\mathcal{J}_t^{\mathsf{T}} \mathcal{J}_t\right)^{-1} \mathcal{J}_t^{\mathsf{T}} \varepsilon_t, \tag{4.66}$$

   with $\alpha = (1-\gamma)\beta$ and

   $$\varepsilon_t = \begin{pmatrix} \left(y_{\mathrm{a},t} + \hat{R}_{t|t-1}^{\mathrm{bn}} g^{\mathrm{n}}\right) \Sigma_{\mathrm{a}}^{-1/2} \\ \left(y_{\mathrm{m},t} - \hat{R}_{t|t-1}^{\mathrm{bn}} m^{\mathrm{n}}\right) \Sigma_{\mathrm{m}}^{-1/2} \end{pmatrix},$$

   $$\mathcal{J}_t = \frac{\partial \varepsilon_t}{\partial q_{\mathrm{am},t}^{\mathrm{nb}}} = \begin{pmatrix} \left.\frac{\partial R_{\mathrm{am},t}^{\mathrm{bn}}}{\partial q_{\mathrm{am},t}^{\mathrm{nb}}}\right|_{q_{\mathrm{am},t}^{\mathrm{nb}} = \hat{q}_{t|t-1}^{\mathrm{nb}}} g^{\mathrm{n}} \, \Sigma_{\mathrm{a}}^{-1/2} \\ -\left.\frac{\partial R_{\mathrm{am},t}^{\mathrm{bn}}}{\partial q_{\mathrm{am},t}^{\mathrm{nb}}}\right|_{q_{\mathrm{am},t}^{\mathrm{nb}} = \hat{q}_{t|t-1}^{\mathrm{nb}}} m^{\mathrm{n}} \, \Sigma_{\mathrm{m}}^{-1/2} \end{pmatrix}.$$

   (c) Renormalize the quaternion as

   $$\hat{q}_{t|t}^{\mathrm{nb}} = \frac{\tilde{q}_{t|t}^{\mathrm{nb}}}{\|\tilde{q}_{t|t}^{\mathrm{nb}}\|_2}. \tag{4.67}$$

end for

of the different algorithms will be analyzed. The simulated data allows for controlled analysis of the workings of the algorithms. Furthermore, it allows us to compare the different algorithms using Monte Carlo simulations. The experimental data shows the applicability to real-world scenarios. We will start by introducing the data sets.

Experimental data is collected using the setup shown in Figure 4.2, where data is collected using multiple mobile IMUs and smartphones. The algorithms presented in this section can be applied to measurements from any of these devices. However, we focus our analysis on the data from the Trivisio Colibri Wireless IMU [**?**]. In Figure 4.3, the inertial and magnetometer measurements from this IMU are displayed for around 100 seconds during which the sensor is rotated around all three axes. The experiments are performed in a lab equipped with multiple cameras [**?**], able to track the optical markers shown in Figure 4.2. This provides highly accurate ground truth reference position and orientation information, against which we can compare our estimates. For comparison, the optical and IMU data need to be time-synchronized and aligned. We synchronize the data by correlating the norms of the gyroscope measurements and of the angular velocity estimated by the optical system. Alignment is done using the orientation estimates in combination with Theorem 4.2 from [**?**].

In Figure 4.4, simulated inertial and magnetometer measurements are displayed. The data represents a sensor that is kept stationary for 100 samples, after which it is rotated around all three axes. The sensor is assumed to be rotated around the origin of the accelerometer triad. Hence, during the entire data set, the accelerometer is assumed to only measure the gravity vector. The magnitude of the simulated gravity vector is 9.82 m/s^2. The magnitude of the simulated local magnetic field is equal to one. Its direction is approximately equal to that in Linköping, Sweden, where a dip angle of 71° leads to a magnetic field $m^{\mathrm{n}} = \begin{pmatrix} 0.33 & 0 & -0.95 \end{pmatrix}^{\mathsf{T}}$. The simulated noise levels are

$$\begin{aligned} e_{\mathrm{a},t} &\sim \mathcal{N}(0, \sigma_{\mathrm{a}}^2 \mathcal{I}_3), & \sigma_{\mathrm{a}} &= 1 \cdot 10^{-1}, \\ e_{\omega,t} &\sim \mathcal{N}(0, \sigma_{\omega}^2 \mathcal{I}_3), & \sigma_{\omega} &= 1 \cdot 10^{-2}, \\ e_{\mathrm{m},t} &\sim \mathcal{N}(0, \sigma_{\mathrm{m}}^2 \mathcal{I}_3), & \sigma_{\mathrm{m}} &= 1 \cdot 10^{-1}. \end{aligned}$$

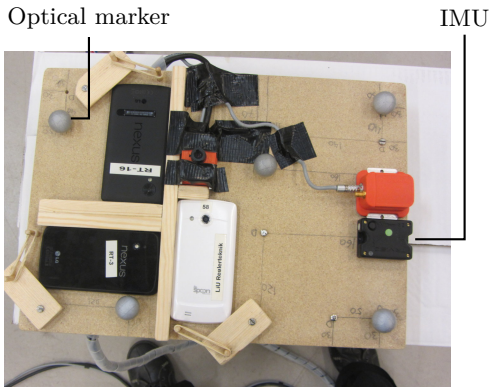


Figure 4.2: Experimental setup where an IMU is used to collected inertial and magnetometer measurements. Optical markers are tracked using multiple cameras, leading to accurate reference position and orientation estimates. Note that the experimental setup also contains additional IMUs and smartphones. The data from these sensors is not considered in this work.

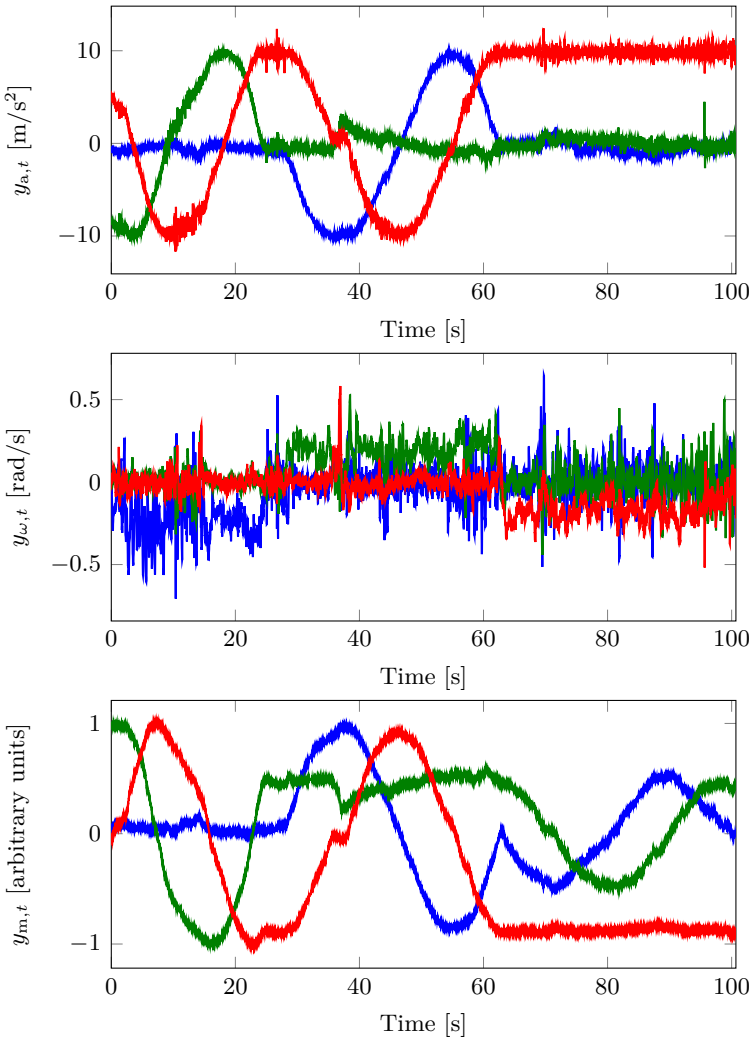


Figure 4.3: Measurements from an accelerometer ($y_{a,t}$, top), a gyroscope ($y_{\omega,t}$, middle) and a magnetometer ($y_{m,t}$, bottom) for 100 seconds of data collected with the IMU shown in Figure 4.2.

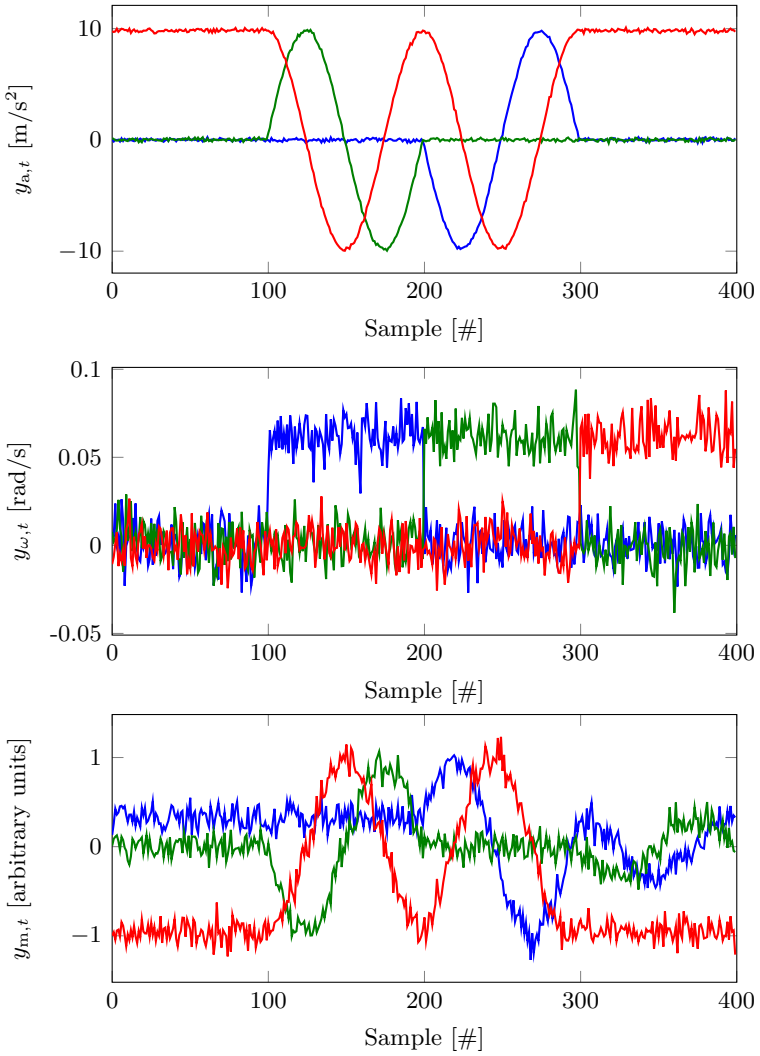


Figure 4.4: Simulated measurements from an accelerometer ($y_{a,t}$, top), a gyroscope ($y_{\omega,t}$, middle) and a magnetometer ($y_{m,t}$, bottom).

Note that we deliberately chose the noise levels to be fairly high, to clearly illustrate the workings of the different algorithms.

Although our algorithms parametrize the orientation as quaternions, it is typically more intuitive to visualize the orientation estimates in Euler angles. Hence, we visualize our results in terms of roll, pitch and heading (yaw) angles. Both for the experimental data and for the simulated data, we are able to compare our estimates $\hat{q}_t^{\text{nb}}$ to reference orientations denoted $q_{\text{ref},t}^{\text{nb}}$. To represent the orientation error, we compute a difference quaternion Δq_t as

$$\Delta q_t = \hat{q}_t^{\text{nb}} \odot \left(q_{\text{ref},t}^{\text{nb}}\right)^{\mathsf{c}}, \tag{4.68}$$

which can be converted to Euler angles for visualization. Note that using this definition, the orientation errors in Euler angles can be interpreted as the errors in roll, pitch and heading.

4.5.1 General characteristics

In this section, we will discuss some general characteristics of the orientation estimation problem and illustrate them in three different examples. Our goal is not to compare the different estimation algorithms, but to illustrate some characteristics common to all of them.

In Example 4.1 we focus on the accuracy of the orientation estimates that can be obtained if the state space model (3.75) is completely true. We illustrate that it is typically easier to obtain accurate roll and pitch estimates than it is to obtain accurate heading estimates.

Example 4.1 (Orientation estimation using inertial and magnetometer data) *The orientation errors from the smoothing optimization approach in Algorithm 1 using simulated inertial and magnetometer measurements as illustrated in Figure 4.4 are depicted in the top plot of Figure 4.5. For comparison we also show the orientation errors from dead-reckoning the gyroscope measurements in the bottom plot (see also §1.2). These errors can be seen to drift over time.*

Although the accelerometer and the magnetometer measurement noises are of equal magnitude, the heading angle is estimated with less accuracy compared to the roll and pitch angles. The reason for this is twofold. First, the signal to noise ratio for the magnetometer is worse than that of the accelerometer, since the magnetometer signal has a magnitude of 1 *while the accelerometer signal has a magnitude of* 9.82 m/s^2*. Second, only the horizontal component of the local magnetic field vector provides heading information. This component is fairly small due to the large dip angle (*71°*) in Linköping, Sweden.*

The accelerometer provides inclination information, while the magnetometer provides heading information, see §3.4. In case only inertial measurements and no magnetometer measurements are available, the heading can only be estimated using the gyroscope measurements and will therefore drift over time. This is illustrated in Example 4.2.

Example 4.2 (Orientation estimation using only inertial measurements) *The orientation errors from the smoothing optimization approach in Algorithm 1 using simulated inertial measurements as presented in Figure 4.4 can be found in Figure 4.6. The roll and pitch angles can be seen to be accurate, while the heading angle drifts significantly. To obtain the results in Figure 4.6, we used data with the same noise realization as in Example 4.1. Hence, we refer to Figure 4.5 for comparison to the orientation errors from dead-reckoning the gyroscope measurements. The drift in the heading angle can be seen to be similar to the drift from dead-reckoning of the gyroscope measurements.*

The two examples above assume that our state space model (3.75) is an accurate description of the measurements. In practice, however, this is not always the case, for instance due to the presence of magnetic material in the vicinity of the sensor. In Example 4.3 we illustrate that if the state space model does not accurately describe the data, it is not possible to obtain accurate orientation estimates.

Example 4.3 (Orientation estimation in the presence of magnetic material) *We simulate 400 samples of stationary data. Between samples 150 and 250, we simulate the presence of a magnetic material, causing a change in the magnetic field of* $\begin{pmatrix}0.1 & 0.3 & 0.5\end{pmatrix}^{\mathsf{T}}$*. In Figure 4.7, we show the adapted*

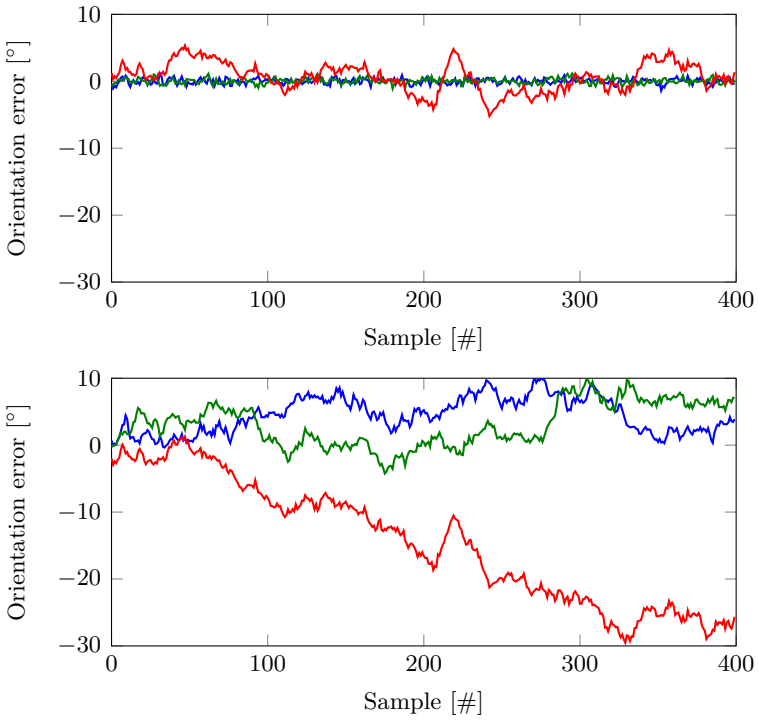


Figure 4.5: Orientation errors in roll (blue), pitch (green) and heading (red) using simulated measurements for (top) Algorithm 1 using inertial and magnetometer measurements and (bottom) dead-reckoning of the gyroscope measurements.

magnetometer data and the orientation estimates using the smoothing optimization approach from §4.1. As can be seen, the heading estimates show significant errors when the magnetic material is present. Depending on the amount of disturbance and the uncertainty in the inertial and magnetometer measurements, magnetic material can also result in errors in the roll and pitch estimates.

4.5.2 Representing uncertainty

So far, we have discussed the *quality* of the orientation estimates in three different examples. However, we did not discuss the *uncertainty* of the estimates. Algorithms 1–4 provide estimates of the uncertainties. We will now discuss how these uncertainties can be displayed and interpreted and highlight some difficulties with this.

Both the optimization and the EKF approaches discussed in §4.1–§4.3 compute the uncertainty of the estimates in terms of a covariance matrix. Let us use the more general notation $\operatorname{cov}(\hat{\eta}_t^{\mathrm{n}})$ for the covariance of the orientation deviation states $\hat{\eta}_t^{\mathrm{n}}$, $t = 1, \ldots, N$ computed in Algorithms 1, 2 and 4, and $\operatorname{cov}(\hat{q}_t^{\mathrm{nb}})$ for the covariance of the quaternion states $\hat{q}_t^{\mathrm{nb}}$, $t = 1, \ldots, N$ computed in Algorithm 3. If the states would be in normal, Euclidean space, the square root of the diagonal of these matrices would represent the standard deviation σ of the estimates in the different directions. These could then be visualized by for instance plotting 3σ confidence bounds around the estimates.

One could imagine that an equivalent way of visualizing the orientation deviation uncertainties, would be to compute the 3σ bounds in terms of orientation deviations in each of the three directions as

$$\left(\Delta\eta_{i,t}^{\mathrm{n}}\right)_{+3\sigma} = +3\sqrt{\left(\operatorname{cov}(\hat{\eta}_t^{\mathrm{n}})\right)_{ii}}, \qquad i = 1, \ldots, 3, \tag{4.69a}$$

$$\left(\Delta\eta_{i,t}^{\mathrm{n}}\right)_{-3\sigma} = -3\sqrt{\left(\operatorname{cov}(\hat{\eta}_t^{\mathrm{n}})\right)_{ii}}, \qquad i = 1, \ldots, 3, \tag{4.69b}$$

after which the bounds can be parametrized in terms of quaternions as

$$\left(q_t^{\mathrm{nb}}\right)_{+3\sigma} = \exp_{\mathsf{q}}\left(\Delta\eta_t^{\mathrm{n}}\right)_{+3\sigma} \odot \hat{q}_t^{\mathrm{nb}}, \tag{4.70a}$$

$$\left(q_t^{\mathrm{nb}}\right)_{-3\sigma} = \exp_{\mathsf{q}}\left(\Delta\eta_t^{\mathrm{n}}\right)_{-3\sigma} \odot \hat{q}_t^{\mathrm{nb}}. \tag{4.70b}$$

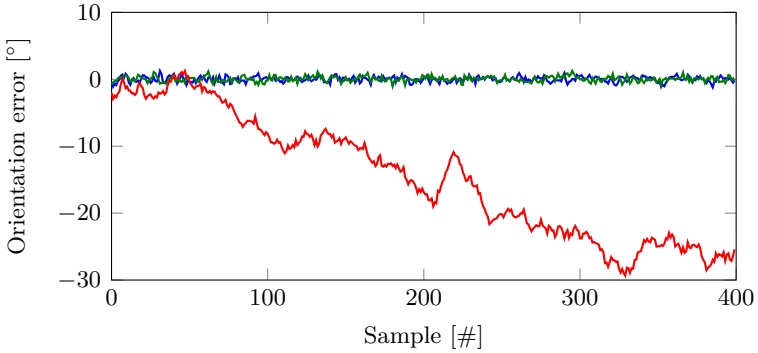


Figure 4.6: Orientation errors in roll (blue), pitch (green) and heading (red) using simulated measurements for Algorithm 1 using inertial measurements only.

The resulting estimates and bounds are visualized in terms of Euler angles in Figure 4.8 for simulated data similar to the data presented in Figure 4.4. The orientation and its covariance are estimated using Algorithm 4. As can be seen, the bounds are difficult to interpret due to the wrapping of the Euler angles.

As argued in [**?**], it is more intuitive to directly represent the uncertainty in terms of orientation deviations. The covariance $\operatorname{cov}(\hat{\eta}_t^{\mathrm{n}})$ can be interpreted as the uncertainty in the roll, pitch and heading angles as illustrated in Example 4.4.

Example 4.4 (Orientation estimation using only inertial measurements (continued)) *Since the accelerometer provides only inclination information, in the case of Example 4.2 where magnetometer measurements are unavailable, we expect only the roll and pitch angles to be estimated with small uncertainty. In fact, we expect the uncertainty of the heading at $t = 1$ to be equal to the uncertainty of the initial Σ_i from §3.6 and to steadily grow over time, depending on the amount of gyroscope noise. In Figure 4.9, we plot the standard deviation σ of the orientation estimates computed using the smoothing algorithm from §4.1 as the square root of the diagonal elements of* $\operatorname{cov}(\hat{\eta}_t^{n})$. *As can be seen, the standard deviation of the yaw angle at $t = 1$ is indeed* $20°$ *as modeled in §3.6. The increase in the uncertainty in the yaw angle exactly matches the increase of the uncertainty due to dead-reckoning.*

From Example 4.4 it can be concluded that $\operatorname{cov}(\hat{\eta}_t^{\mathrm{n}})$ seems to be an intuitive measure of the uncertainty of the orientation estimates. The covariance $\operatorname{cov}(\hat{q}_t^{\mathrm{nb}})$ computed by Algorithm 3 relates to $\operatorname{cov}(\hat{\eta}_t^{\mathrm{n}})$ as

$$\begin{aligned}
\operatorname{cov}(\hat{q}_t^{\mathrm{nb}}) &= \operatorname{cov}\left(\exp_{\mathsf{q}}(\tfrac{\hat{\eta}_t^{\mathrm{n}}}{2}) \odot \tilde{q}_t^{\mathrm{nb}}\right) \\
&= \tfrac{1}{4}\left(\tilde{q}_t^{\mathrm{nb}}\right)^{\mathsf{R}} \tfrac{\partial \exp_{\mathsf{q}}(\hat{\eta}_t^{\mathrm{n}})}{\partial \hat{\eta}_t^{\mathrm{n}}} \operatorname{cov}(\hat{\eta}_t^{\mathrm{n}}) \left(\tfrac{\partial \exp_{\mathsf{q}}(\hat{\eta}_t^{\mathrm{n}})}{\partial \hat{\eta}_t^{\mathrm{n}}}\right)^{\mathsf{T}} \left(\tilde{q}_t^{\mathrm{bn}}\right)^{\mathsf{R}}, \qquad (4.71a) \\
\operatorname{cov}(\hat{\eta}_t^{\mathrm{n}}) &= \operatorname{cov}\left(2\log_{\mathsf{q}}(\hat{q}_t^{\mathrm{nb}} \odot \tilde{q}_t^{\mathrm{bn}})\right) \\
&= 4\tfrac{\partial \log_{\mathsf{q}}(q)}{\partial q}\left(\tilde{q}_t^{\mathrm{bn}}\right)^{\mathsf{R}} \operatorname{cov}(\hat{q}_t^{\mathrm{nb}}) \left(\tilde{q}_t^{\mathrm{nb}}\right)^{\mathsf{R}} \left(\tfrac{\partial \log_{\mathsf{q}}(q)}{\partial q}\right)^{\mathsf{T}}. \qquad (4.71b)
\end{aligned}$$

The relation (4.71) allows us to compare the orientation estimates and covariances from the different algorithms in more detail in Example 4.5.

Example 4.5 (Orientation estimation using only inertial measurements (continued)) *As discussed in Example 4.4, using only inertial measurements and no magnetometer measurements, we can only expect to be able to accurately estimate the inclination. The uncertainty of the heading estimates grows over time. We will now analyze the behavior of the different algorithms in more detail for this specific example. In Table 4.1, we show the root mean square error (RMSE) values over* 100 *Monte Carlo simulations for Algorithms 1–4. In Figure 4.10, we also represent the orientation estimates from the four algorithms for one of these realizations. As can be seen from both Table 4.1 and Figure 4.10, as expected, the smoothing algorithm outperforms the other algorithms. However, more surprisingly, the EKF with quaternion states has much larger errors in the heading angle. In Figure 4.11, we also show*

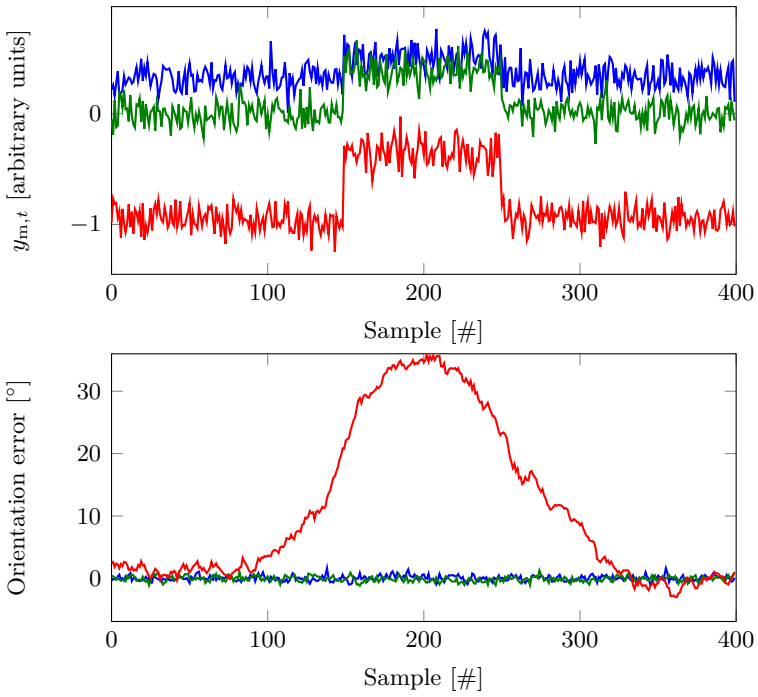


Figure 4.7: Top: Simulated magnetometer measurements $y_{\mathrm{m},t}$ for 400 samples of stationary data. Between samples 150 and 250 we simulate the presence of a magnetic material in the vicinity of the sensor. Bottom: Orientation estimates in roll (blue), pitch (green) and heading (red) using the simulated inertial and magnetometer measurements.

Table 4.1: Mean RMSE values over 100 Monte Carlo simulations estimating orientation using only inertial measurements.

RMSE	Roll [°]	Pitch [°]	Heading [°]
Smoothing optimization (Alg. 1)	0.39	0.39	7.46
Filtering optimization (Alg. 2)	0.46	0.46	7.46
EKF quaternions (Alg. 3)	0.46	0.46	17.49
EKF orientation deviation (Alg. 4)	0.46	0.46	7.46

the standard deviations of the estimates from all algorithms. As can be seen, the EKF with quaternion states over-estimates its confidence in the estimates of the heading direction. This can most likely be attributed to linearization issues.

From Example 4.5, it can be concluded that not properly estimating the covariance can have negative effects on the quality of the estimates. It can also be concluded from this section that covariances can best be represented in terms of $\mathrm{cov}(\hat{\eta}_t^{\mathrm{n}})$ but that they are difficult to visualize in Euler angles. Because of that, in the remainder of this section, we will typically plot the uncertainty in a separate plot like in Figure 4.11.

4.5.3 Comparing the different algorithms

We use the experimental data presented in Figure 4.3 to assess the quality of the estimates from the different algorithms. We also have a short sequence of stationary data available that allows us to determine the gyroscope bias and the sensor noise covariances. The gyroscope bias is subtracted from the data to allow for better orientation estimates. The standard deviation of the gyroscope noise is experimentally determined to be $\sigma_\omega = 4.9 \cdot 10^{-3}$. As discussed in §3.4, the covariance matrices in the measurement models reflect not only the sensor noise, but also the model uncertainty. Both the assumptions that the sensor's acceleration is zero and that the magnetic field is homogeneous are only

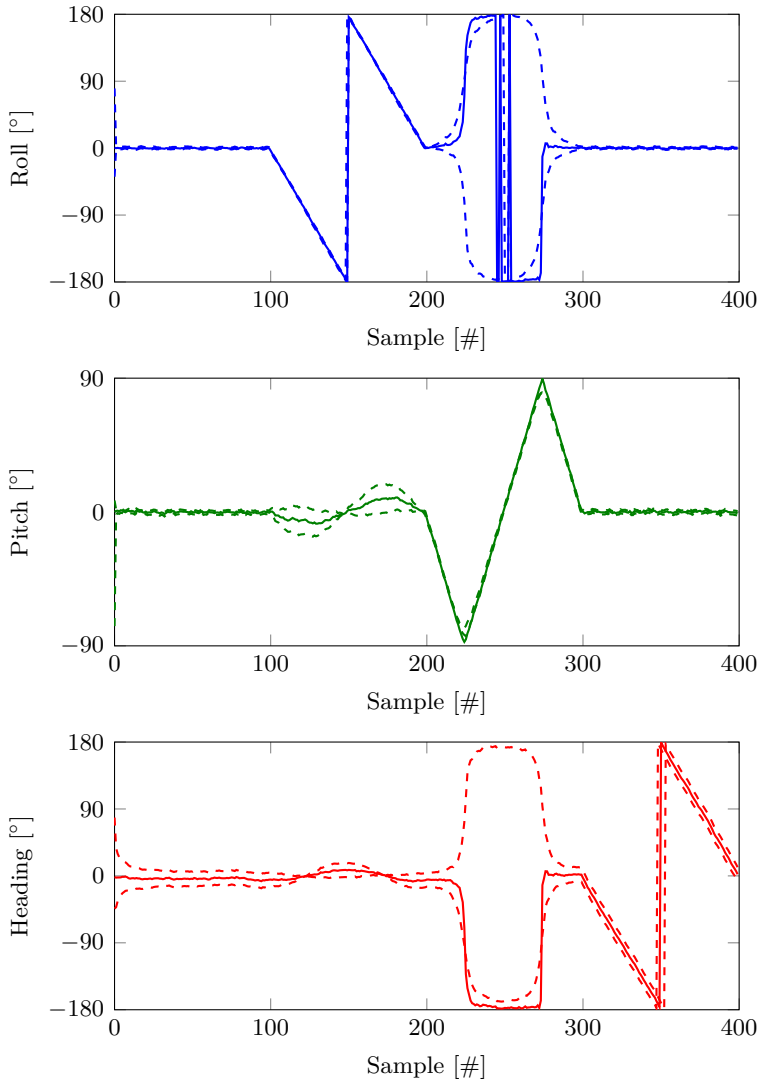


Figure 4.8: Orientation estimates (solid) and 3σ bounds (dashed) in roll (blue), pitch (green) and heading (red) using inertial and magnetometer measurements.

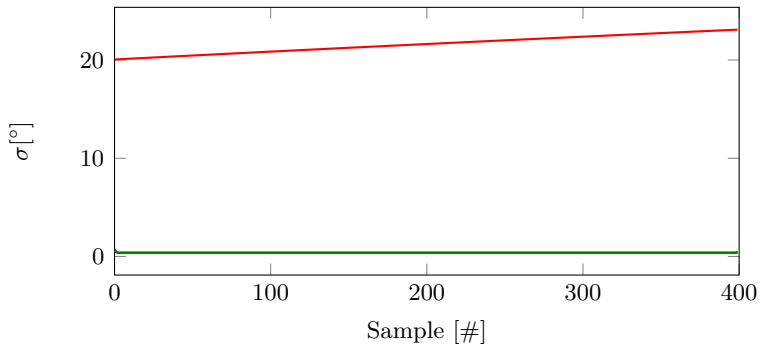
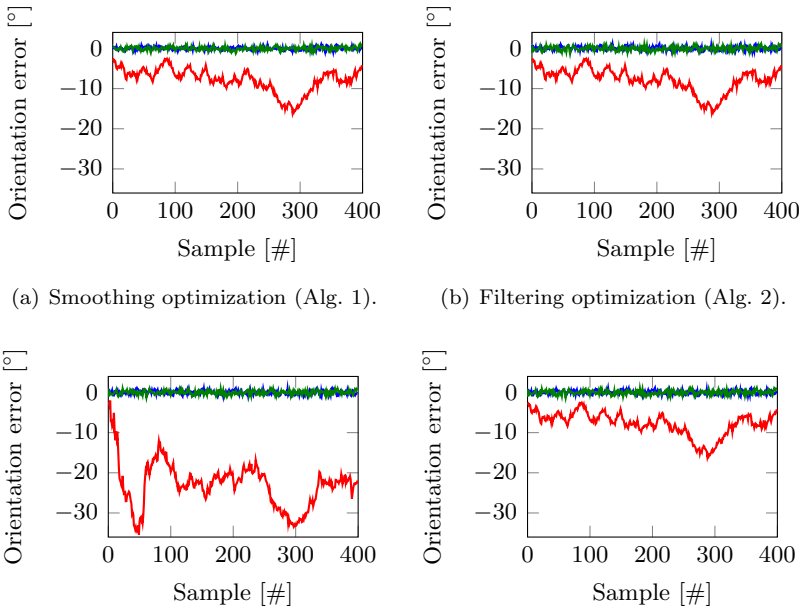


Figure 4.9: Standard deviation σ in degrees of the orientation estimates in roll (blue), pitch (green) and heading (red) using only inertial measurements.



(a) Smoothing optimization (Alg. 1).

(b) Filtering optimization (Alg. 2).

(c) EKF quaternions (Alg. 3)

(d) EKF orientation deviation (Alg. 4).

Figure 4.10: Orientation estimates of Algorithms 1–4 in roll (blue), pitch (green) and heading (red) using only inertial measurements.

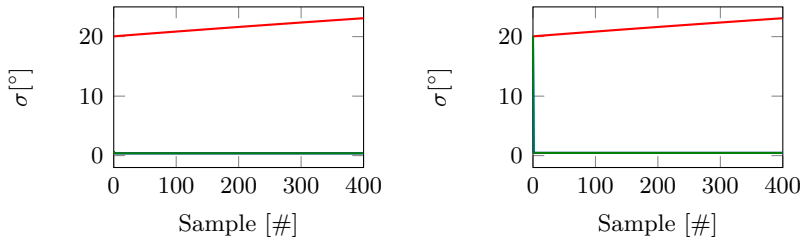
Table 4.2: RMSE of the orientation estimates obtained using Algorithms 1–5 and the experimental data presented in Figure 4.3.

RMSE	Roll [°]	Pitch [°]	Heading [°]
Smoothing optimization (Alg. 1)	1.03	0.48	0.81
Filtering optimization (Alg. 2)	1.14	0.56	1.28
EKF quaternions (Alg. 3)	1.13	0.52	1.04
EKF orientation deviation (Alg. 4)	1.14	0.56	1.28
Complementary filter (Alg. 5)	0.95	0.62	1.55

approximately true. Because of this, we choose $\sigma_{\mathrm{a}} = 2.6 \cdot 10^{-1}$ and $\sigma_{\mathrm{m}} = 2.5 \cdot 10^{-1}$. These values are a factor 10 respectively 100 larger than the experimentally determined noise values. Note that this choice is highly dependent on the data set. The RMSE values as compared to the optical reference system for the different methods described in this chapter are summarized in Table 4.2. A good value for α in the complementary filter is experimentally determined to be 0.001 for this specific data set. As an illustration of the estimates, the orientation estimates as obtained using the smoothing algorithm and the orientations from the optical reference system are shown in Figure 4.12.

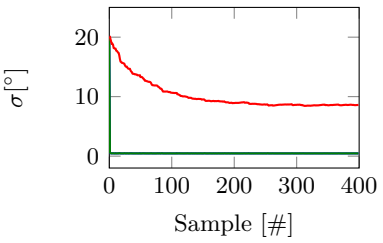
It is of course difficult to draw quantitative conclusions based on only one data set. The RMSE values in Table 4.2 should therefore mainly be seen as an indication of the performance. Comparing the different algorithms amongst each other is hard. In fact, the algorithms perform differently with respect to each other for different choices of the covariance matrices. Because of this, we will study the accuracy of the different methods using simulated data instead.

We run 100 Monte Carlo simulations where the simulated data illustrated in Figure 4.4 is generated with different noise realizations. Table 4.3 shows the mean RMSE for the five estimation algorithms. Algorithms 1–4 use the noise covariance matrices that are also used to generate the data. The complementary filter also uses these to determine the orientation from the accelerometer and magnetometer data. To combine this with the orientation from the gyroscope measurements, we need to determine a good value for α. We empirically choose a value that leads to good performance for this data set. The orientation from the accelerometer and magnetometer measurements can be expected to be more accurate in the roll and the pitch than in the heading, see Example 4.1. Since α is scalar, however,

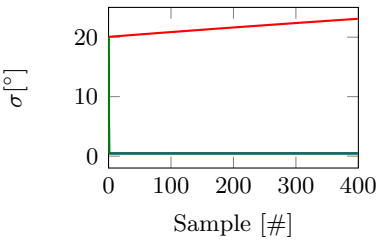


(a) Smoothing optimization (Alg. 1).

(b) Filtering optimization (Alg. 2).



(c) EKF quaternions (Alg. 3).



(d) EKF orientation deviation (Alg. 4).

Figure 4.11: Standard deviation σ in degrees of the orientation estimates of Algorithms 1–4 in roll (blue), pitch (green) and heading (red) using only inertial measurements.

Table 4.3: Mean RMSE of the orientation estimates from 100 Monte Carlo simulations using Algorithms 1–5.

RMSE	Roll [°]	Pitch [°]	Heading [°]
Smoothing optimization (Alg. 1)	0.39	0.39	2.30
Filtering optimization (Alg. 2)	0.45	0.45	3.54
EKF quaternions (Alg. 3)	0.45	0.45	3.57
EKF orientation deviation (Alg. 4)	0.45	0.45	3.55
Complementary filter (Alg. 5), $\alpha = 0.07$	1.44	1.43	4.39
Complementary filter (Alg. 5), $\alpha = 0.7$	0.47	0.47	12.98

it is not possible to weigh these contributions differently. Because of this, we present results both for $\alpha = 0.07$, which leads to small RMSE values in the heading but relatively larger RMSE in the roll and the pitch, and for $\alpha = 0.7$, which leads to small RMSE values in the roll and pitch but large RMSE in the heading.

From the results summarized in Table 4.3, it can be seen that the smoothing approach outperforms the filtering approaches. The estimates for one of the noise realizations are shown in Figure 4.13. For Algorithms 1–4, we also depict the covariance estimates in Figure 4.14. The filtering approaches from Algorithms 2–4 estimate the standard deviation of the orientation errors at $t = 1$ to be equal to 20°. After this, they can be seen to converge to around 3.16° degrees for the heading angle and 0.46° for roll and pitch angles. The smoothing algorithm estimates an uncertainty in the heading angle of around 3.17° for the first and last sample, while converging to a standard deviation of 2.25° for the middle of the data set. For the roll and pitch angles, the initial and final uncertainties are estimated to be around 0.73°, converging to 0.39° for the middle of the data set. Note that these values correspond fairly well with the RMSE values in Table 4.3.

For the Monte Carlo simulations described above, the three filtering algorithms perform similarly. However, differences can be seen when an update of the filter needs to correct the orientation estimates significantly. Examples for when this happens are when the initial orientation is not accurately known or when magnetometer measurements are not available for a longer period of time. In these cases, the uncertainty of the state is large and large corrections to the state estimates are needed when measurements become available. To analyze this case in more detail, we assume that the estimate of the initial orientation $\check{q}_1^{\text{nb}}$ is normal distributed around the true initial orientation with a standard deviation of 20°. Hence, we do not use the first accelerometer and magnetometer data for initialization. Note that a

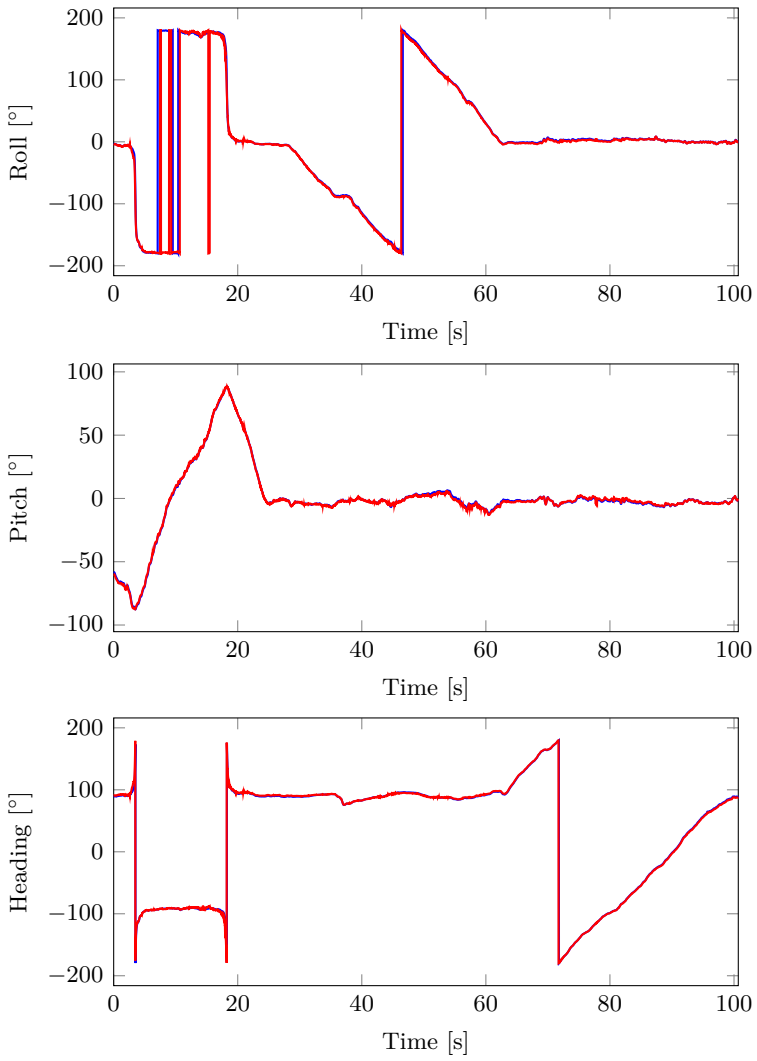
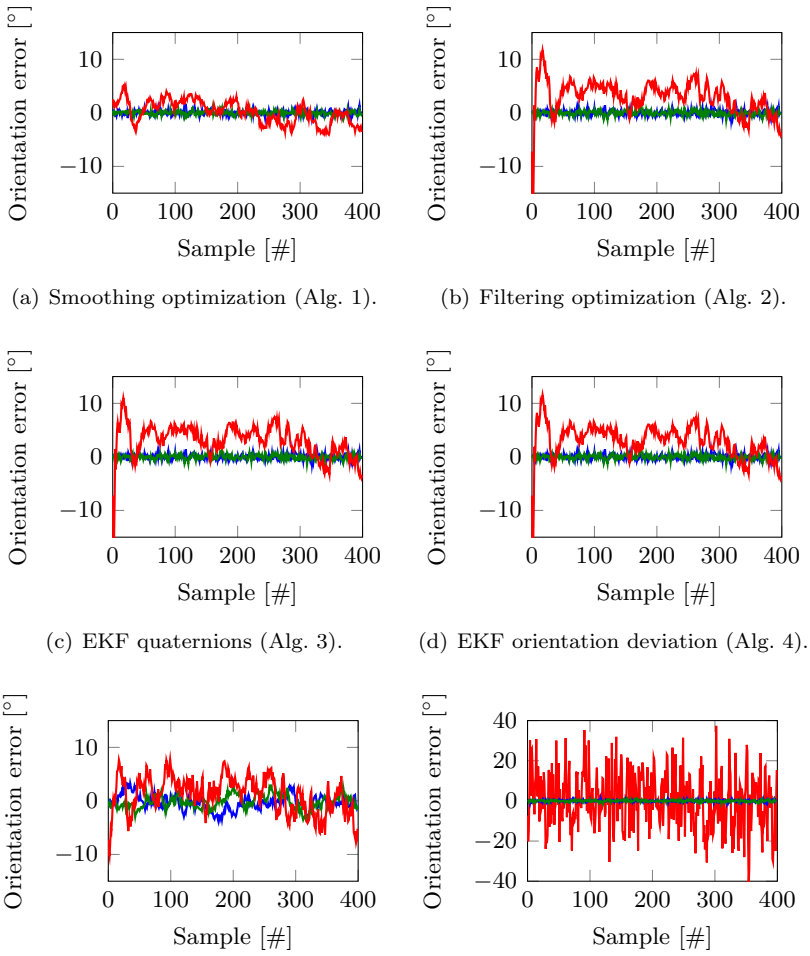


Figure 4.12: Red: Orientation from the optical reference system. Blue: Orientation estimates obtained using Algorithm 1 for the experimental data from Figure 4.3.

Table 4.4: Mean RMSE of the orientation estimates from 100 Monte Carlo simulations. The estimate of the initial orientation is assumed to be normal distributed around the true initial orientation with a standard deviation of 20°.

RMSE	Roll [°]	Pitch [°]	Heading [°]
Smoothing optimization (Alg. 1)	0.39	0.39	2.29
Filtering optimization (Alg. 2)	1.06	0.95	3.55
EKF quaternions (Alg. 3)	1.08	0.97	4.41
EKF orientation deviation (Alg. 4)	1.08	0.96	3.57
Complementary filter (Alg. 5), $\alpha = 0.07$	3.02	2.77	4.48
Complementary filter (Alg. 5), $\alpha = 0.7$	1.13	1.01	12.99



(a) Smoothing optimization (Alg. 1).

(b) Filtering optimization (Alg. 2).

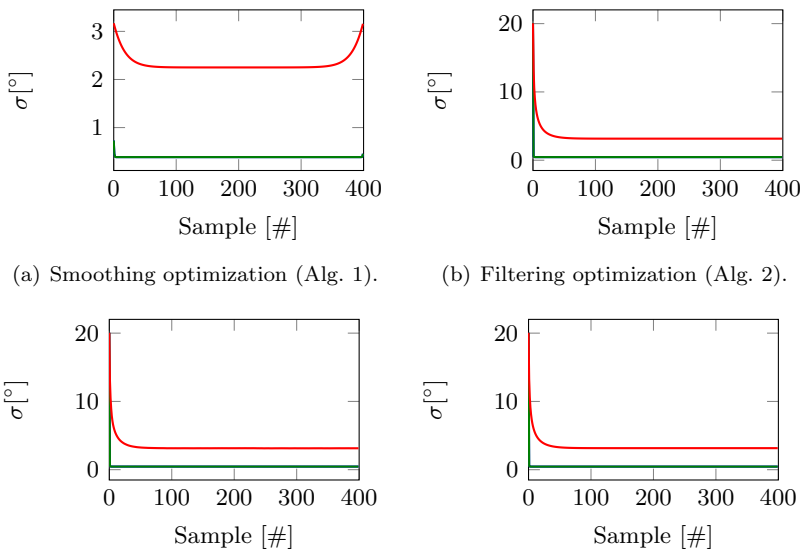
(c) EKF quaternions (Alg. 3).

(d) EKF orientation deviation (Alg. 4).

(e) Complementary filter (Alg. 5), $\alpha = 0.07$.

(f) Complementary filter (Alg. 5), $\alpha = 0.7$.

Figure 4.13: Orientation errors of Algorithms 1–5 in roll (blue), pitch (green) and heading (red) using simulated inertial and magnetometer measurements.



(a) Smoothing optimization (Alg. 1).

(b) Filtering optimization (Alg. 2).

(c) EKF quaternions (Alg. 3).

(d) EKF orientation deviation (Alg. 4).

Figure 4.14: Standard deviation σ in degrees of the orientation estimates of Algorithms 1–4 in roll (blue), pitch (green) and heading (red) using simulated inertial and magnetometer measurements. Note the different scale on the vertical axis of (a) as compared to (b)–(d).

standard deviation of 20° is equal to the uncertainty on the initial state assumed by the algorithms. The results for 100 Monte Carlo simulations are summarized in Table 4.4. As can be seen, specifically the EKF with quaternion states and the complementary filter with $\alpha = 0.07$ perform worse than Algorithm 2 and Algorithm 4 for this data.

Which algorithm to use is highly application-specific. However, in general it can be concluded that all five algorithms actually produce fairly good orientation estimates, assuming that the models from §3.7 are indeed valid. The smoothing algorithm performs better than the filtering approaches but it is also the most computationally expensive. The EKF with quaternion states and the complementary filter suffer from linearization issues when large orientation corrections need to be made or when magnetometer data is unavailable.

4.6 Extending to pose estimation

In §4.5, we have evaluated the performance of Algorithms 1–5 for orientation estimation. The estimation methods presented in §4.1–§4.3 can also be used to estimate the sensor's pose using the state space model (3.74). Complementary filtering is predominantly used for orientation estimation and will therefore not be considered in this section.

The pose estimation problem can be written as a smoothing optimization problem as

$$\hat{x}_{1:N} = \arg\min_{x_{1:N}} \underbrace{\|e_{\mathrm{p,i}}\|^2_{\Sigma^{-1}_{\mathrm{p,i}}} + \|e_{\mathrm{v,i}}\|^2_{\Sigma^{-1}_{\mathrm{v,i}}} + \|e_{\eta,\mathrm{i}}\|^2_{\Sigma^{-1}_{\eta,\mathrm{i}}}}_{\text{Prior}} + \underbrace{\sum_{t=2}^{N} \|e_{\mathrm{p},t}\|^2_{\Sigma^{-1}_{\mathrm{p}}}}_{\text{Measurement model}} + \underbrace{\sum_{t=2}^{N} \|e_{\omega,t}\|^2_{\Sigma^{-1}_{\omega}} + \|e_{\mathrm{a,p},t}\|^2_{\Sigma^{-1}_{\mathrm{a,p}}} + \|e_{\mathrm{a,v},t}\|^2_{\Sigma^{-1}_{\mathrm{a,v}}}}_{\text{Dynamics}}, \tag{4.72}$$

with $x_t = \begin{pmatrix} p_t^{\mathsf{T}} & v_t^{\mathsf{T}} & (\eta_t^{\mathrm{n}})^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}$ and

$$
\begin{aligned}
e_{\mathrm{p,i}} &= p_1^{\mathrm{n}} - y_{\mathrm{p},1}, & e_{\mathrm{p,i}} &\sim \mathcal{N}(0, \Sigma_{\mathrm{p,i}}), && \text{(4.73a)}\\
e_{\mathrm{v,i}} &= v_1, & e_{\mathrm{v,i}} &\sim \mathcal{N}(0, \Sigma_{\mathrm{v,i}}), && \text{(4.73b)}\\
e_{\eta,\mathrm{i}} &= 2\log_{\mathsf{q}}\left(q_1^{\mathrm{nb}} \odot \mathring{q}_1^{\mathrm{bn}}\right), & e_{\mathrm{i}} &\sim \mathcal{N}(0, \Sigma_{\eta,\mathrm{i}}), && \text{(4.73c)}\\
e_{\mathrm{p,a},t} &= \tfrac{2}{T^2}\left(p_{t+1}^{\mathrm{n}} - p_t^{\mathrm{n}} - T v_t^{\mathrm{n}}\right) - \\
&\quad R_t^{\mathrm{nb}} y_{\mathrm{a},t} - g^{\mathrm{n}}, & e_{\mathrm{p,a},t} &\sim \mathcal{N}(0, \Sigma_{\mathrm{a}}), && \text{(4.73d)}\\
e_{\mathrm{v,a},t} &= \tfrac{1}{T}\left(v_{t+1}^{\mathrm{n}} - v_t^{\mathrm{n}}\right) - R_t^{\mathrm{nb}} y_{\mathrm{a},t} - g^{\mathrm{n}}, & e_{\mathrm{v,a},t} &\sim \mathcal{N}(0, \Sigma_{\mathrm{a}}), && \text{(4.73e)}\\
e_{\omega,t} &= \tfrac{2}{T}\log_{\mathsf{q}}\left(q_t^{\mathrm{bn}} \odot q_{t+1}^{\mathrm{nb}}\right) - y_{\omega,t}, & e_{\omega,t} &\sim \mathcal{N}(0, \Sigma_{\omega}), && \text{(4.73f)}\\
e_{\mathrm{p},t} &= y_{\mathrm{p},t} - p_t^{\mathrm{n}}, & e_{\mathrm{p},t} &\sim \mathcal{N}(0, \Sigma_{\mathrm{p}}). && \text{(4.73g)}
\end{aligned}
$$

In this section, we will discuss some details about the workings of the pose estimation algorithm using this model. We will not go through a complete derivation of the four algorithms. However, the adaptations that are needed to use Algorithms 1–4 for pose estimation can be found in Appendix B.

An important observation is that $e_{\mathrm{a,p},t}$ and $e_{\mathrm{a,v},t}$ in (4.73d) and (4.73e) depend on the orientation R_t^{nb}. Because of this, the position, velocity and orientation states are coupled. The position measurements therefore do not only provide information about the position and velocity, but also about the orientation of the sensor. This is the reason why it is no longer essential to include magnetometer data and to assume that the acceleration is approximately zero. However, the accuracy of the orientation estimates depends on the movements of the sensor. This will be illustrated below. For this, we simulate 400 samples of inertial and position measurements for a non-rotating sensor with noise levels

$$
\begin{aligned}
e_{\mathrm{a},t} &\sim \mathcal{N}(0, \sigma_{\mathrm{a}}^2 \mathcal{I}_3), & \sigma_{\mathrm{a}} &= 1 \cdot 10^{-1},\\
e_{\omega,t} &\sim \mathcal{N}(0, \sigma_{\omega}^2 \mathcal{I}_3), & \sigma_{\omega} &= 1 \cdot 10^{-2},\\
e_{\mathrm{p},t} &\sim \mathcal{N}(0, \sigma_{\mathrm{p}}^2 \mathcal{I}_3), & \sigma_{\mathrm{p}} &= 1 \cdot 10^{-2}.
\end{aligned}
$$

We consider four different sensor motions. The results in this section are based on the solution to the smoothing optimization problem (4.72). First, we simulate data assuming that the sensor is stationary. For this case, the position measurements provide information about the inclination of the sensor, but not about its heading. This is illustrated in Example 4.6.

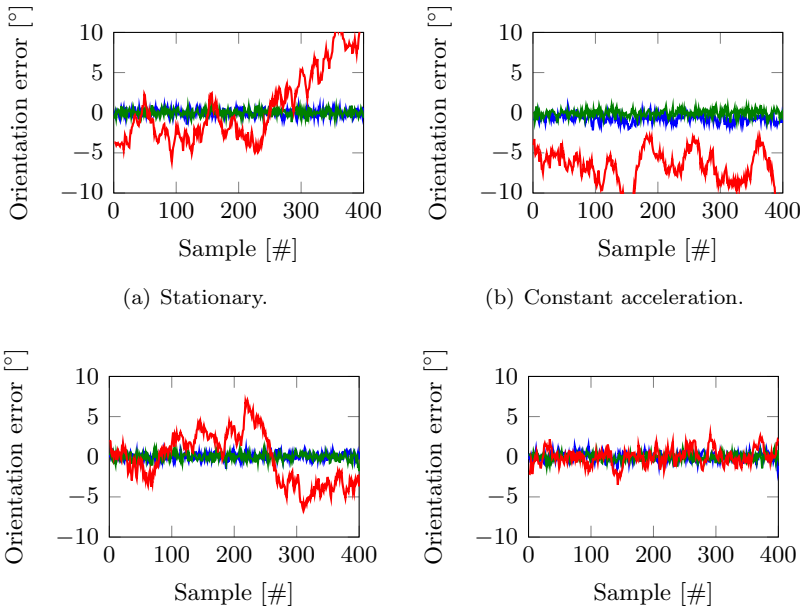
Example 4.6 (Pose estimation for a stationary sensor) *We estimate the pose of a stationary sensor using simulated data and a smoothing algorithm that solves* (4.72) *as described in §4.1. The orientation error for a specific noise realization is depicted in Figure 4.15(a). The inclination errors can be seen to be small, while the heading estimates drift.*

Next, in Example 4.7, we consider the case where the sensor has a constant linear acceleration. For this case, a drift in the orientation estimates can be seen in the direction that is orthogonal to the direction of the accelerometer measurements.

Example 4.7 (Pose estimation for a sensor with constant linear acceleration) *We estimate the pose of a sensor with an acceleration of* 1 $\mathrm{m/s^2}$ *in the y-direction using simulated data and obtain smoothed estimates by solving* (4.72)*. The orientation error for a specific noise realization is depicted in Figure 4.15(b). Again, a drift can be seen in the orientation estimates. This drift is no longer only in the heading direction, but there is also a small drift on the roll angle.*

Finally, in Example 4.8 we consider the case of a time-varying linear acceleration. Based on simulated data, we show that accurate heading estimates can be obtained for this case. Furthermore, we show that the larger the acceleration, the more accurate the heading estimates will be.

Example 4.8 (Pose estimation for a sensor with time-varying linear acceleration) *We estimate the pose of a sensor with an acceleration in the y-direction of* $(y_{a,t})_y \sim \mathcal{N}(0, 0.5)$ $\mathrm{m/s^2}$ *using simulated*



(a) Stationary.

(b) Constant acceleration.

(c) Acceleration $(y_{\mathrm{a},t})_y \sim \mathcal{N}(0, 0.5)$.

(d) Acceleration $(y_{\mathrm{a},t})_y \sim \mathcal{N}(0, 5)$.

Figure 4.15: Orientation errors of the roll (blue), pitch (green) and heading (red) using simulated inertial and magnetometer measurements.

Table 4.5: Mean RMSE of the position and orientation estimates from 100 Monte Carlo simulations. Considered are a stationary sensor, a sensor with constant acceleration and two cases of time-varying accelerations with different magnitudes.

RMSE	Roll [°]	Pitch [°]	Heading [°]	Position [cm]
Stationary	0.41	0.41	11.84	0.97
Constant acceleration	1.23	0.46	11.37	0.97
Acceleration $(y_{\mathrm{a},t})_y \sim \mathcal{N}(0, 0.5)$	0.41	0.41	2.68	0.97
Acceleration $(y_{\mathrm{a},t})_y \sim \mathcal{N}(0, 5)$	0.46	0.39	0.87	0.97

data and compute smoothed estimates by solving (4.72)*. The orientation error for a specific noise realization is depicted in Figure 4.15(c). Furthermore, we simulate data with* $(y_{a,t})_y \sim \mathcal{N}(0, 5)$ m/s^2*. The orientation errors based on this data can be found in Figure 4.15(d). As can be seen, for these cases, it is possible obtain reliable heading estimates using the state space model* (3.74)*. The larger the acceleration, the more accurate the heading estimates.*

In general, it can be concluded that it is possible to estimate position and orientation using the state space model (3.74). Except in the cases of constant or zero acceleration, it is possible to obtain drift-free orientation estimates. The heading accuracy depends on the amount of acceleration. This is summarized in Table 4.5 where the mean RMSE of the state estimates over 100 Monte Carlo simulations is shown. Four cases were considered, inspired by Examples 4.6–4.8.

Chapter 5

Calibration

In Chapter 4, we assumed that the sensors were properly calibrated. In practice, however, there are often calibration parameters to be taken into account. Examples of calibration parameters are the inertial sensor biases discussed in Chapter 2. Furthermore, calibration is specifically of concern when combining the inertial data with other sensors. In these cases, it is important that the inertial sensor axes and the axes of the additional sensors are aligned. Examples include using inertial sensors in combination with magnetometers [**?** **?** **?**] and with cameras [**?** **?** **?**].

In this chapter we will introduce several useful calibration methods. In §5.1 we explain how calibration parameters can be included as unknowns in the smoothing and filtering algorithms from §4.1–§4.3. This results in MAP estimates of the parameters. In §5.2 we instead focus on obtaining maximum likelihood (ML) estimates of the parameters. In §5.3, the workings of the calibration algorithms are illustrated by considering the gyroscope bias to be unknown in the orientation estimation problem. Finally, in §5.4, we discuss the topic of *identifiability*. Parameters are said to be identifiable if they can be estimated from the available data.

5.1 Maximum a posteriori calibration

As discussed in §3.1, unknown parameters θ can be estimated in the smoothing problem (3.1) as

$$\left\{\hat{x}_{1:N}, \hat{\theta}\right\} = \arg\max_{x_{1:N},\theta} p(x_{1:N}, \theta \mid y_{1:N}), \tag{5.1}$$

with

$$p(x_{1:N}, \theta \mid y_{1:N}) \propto p(\theta)p(x_1)\prod_{t=1}^{N} p(x_t \mid x_{t-1}, \theta)p(y_t \mid x_t, \theta). \tag{5.2}$$

Recall that a discussion on the choice of the prior of the parameters $p(\theta)$ and the states $p(x_1)$ can be found in §3.6.

Within the filtering context we typically model the parameters as slowly time-varying states. These can be estimated by solving

$$\left\{\hat{x}_t, \hat{\theta}_t\right\} = \arg\max_{x_t,\theta_t} p(x_t, \theta_t \mid y_{1:t}), \tag{5.3}$$

where

$$p(x_t, \theta_t \mid y_{1:t}) \propto p(y_t \mid x_t, \theta_t)p(x_t, \theta_t \mid y_{1:t-1}), \tag{5.4a}$$

and

$$p(x_t, \theta_t \mid y_{1:t-1}) = \iint p(x_t, \theta_t \mid x_{t-1}, \theta_{t-1})p(x_{t-1}, \theta_{t-1} \mid y_{1:t-1})\mathsf{d}x_{t-1}\mathsf{d}\theta_{t-1}. \tag{5.4b}$$

Note that compared to §3.1, in (5.4) we do not consider the parameters to be part of x_t but instead represent them explicitly. A prior $p(\theta_1)$ on the parameters at $t = 1$ has to be included as well as a dynamic model of the parameters.

Both the formulations (5.1) and (5.3) compute MAP estimates of the parameters. Algorithms 1–4 presented in Chapter 4 can straightforwardly be extended to also estimate these unknown parameters θ or $\theta_{1:N}$. This is illustrated in Example 5.1 for the case of orientation estimation in the presence of an unknown gyroscope bias. Note that Algorithm 5 can also be extended to estimate an unknown gyroscope bias. We will, however, not consider this. Instead, we refer the reader to [**?**].

Example 5.1 (MAP estimates of the gyroscope bias) *It is possible to estimate an unknown gyroscope bias in the state space model* (3.75)*. For this, the dynamic model* (3.75a) *in the smoothing problem described in §4.1 is assumed to include a constant gyroscope bias* δ_ω *as*

$$q_{t+1}^{nb} = q_t^{nb} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}\left(y_{\omega,t} - \delta_\omega - e_{\omega,t}\right)\right). \tag{5.5a}$$

In the filtering algorithms in §4.2 and §4.3, the dynamic model is instead assumed to include a slowly time-varying gyroscope bias $\delta_{\omega,t}$ *as*

$$q_{t+1}^{nb} = q_t^{nb} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}\left(y_{\omega,t} - \delta_{\omega,t} - e_{\omega,t}\right)\right), \tag{5.5b}$$

where the dynamics of the gyroscope bias can be described as a random walk (see also §3.5)

$$\delta_{\omega,t+1} = \delta_{\omega,t} + e_{\delta_\omega,t}, \qquad e_{\delta_\omega,t} \sim \mathcal{N}(0, \Sigma_{\delta_{\omega,t}}). \tag{5.5c}$$

The smoothing algorithm presented in §4.1 can be extended to also estimate δ_ω*. Furthermore, the filtering algorithms presented in §4.2 and §4.3 can be extended to estimate* $\delta_{\omega,t}$ *for* $t = 1, \ldots, N$*. Only minor changes to the algorithms presented in these sections are needed. These mainly concern including derivatives with respect to the additional unknowns* δ_ω *or* $\delta_{\omega,t}$*. Explicit expressions for these can be found in Appendix C.*

5.2 Maximum likelihood calibration

Alternatively, it is possible to obtain ML estimates of the parameters θ as

$$\hat{\theta}^{\mathrm{ML}} = \arg\max_{\theta \in \Theta} \mathcal{L}(\theta; y_{1:N}). \tag{5.6}$$

Here, $\Theta \subseteq \mathbb{R}^{n_\theta}$ and $\mathcal{L}(\theta; y_{1:N})$ is referred to as the likelihood function. It is defined as $\mathcal{L}(\theta; y_{1:N}) \triangleq p_\theta(Y_{1:N} = y_{1:N})$, where $Y_{1:N}$ are random variables and $y_{1:N}$ are a particular realization of $Y_{1:N}$. Using conditional probabilities and the fact that the logarithm is a monotonic function we have the following equivalent formulation of (5.6),

$$\hat{\theta}^{\mathrm{ML}} = \arg\min_{\theta \in \Theta} -\sum_{t=1}^{N} \log p_\theta(Y_t = y_t \mid Y_{1:t-1} = y_{1:t-1}), \tag{5.7}$$

where we use the convention that $y_{1:0} \triangleq \emptyset$. The ML estimator (5.7) enjoys well-understood theoretical properties including strong consistency, asymptotic normality, and asymptotic efficiency [**?**].

Due to the nonlinear nature of the orientation parametrization, our estimation problems are nonlinear, implying that there is no closed form solution available for the one step ahead predictor $p_\theta(Y_t = y_t \mid Y_{1:t-1} = y_{1:t-1})$ in (5.7). However, similar to the filtering approaches from Chapter 4, it is possible to approximate the one step ahead predictor according to

$$p_\theta(Y_t = y_t \mid Y_{1:t-1} = y_{1:t-1}) \approx \mathcal{N}\left(y_t\,;\, \hat{y}_{t|t-1}(\theta), S_t(\theta)\right), \tag{5.8}$$

where $\hat{y}_{t|t-1}(\theta)$ and $S_t(\theta)$ are defined in (4.39) and (4.38), respectively. Inserting (5.8) into (5.7) and neglecting all constants not depending on θ results in the following optimization problem,

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{2} \sum_{t=1}^{N} \|y_t - \hat{y}_{t|t-1}(\theta)\|^2_{S_t^{-1}(\theta)} + \log\det S_t(\theta). \tag{5.9}$$

Table 5.1: Mean RMSE of the orientation estimates from 100 Monte Carlo simulations in the presence of a gyroscope bias that is being estimated.

RMSE	Roll [°]	Pitch [°]	Heading [°]
Smoothing optimization	0.39	0.39	2.29
EKF orientation deviation	0.46	0.46	4.20

Unlike the optimization problems discussed so far, it is not straightforward to obtain an analytical expression of the gradient of (5.9). This is because it is defined recursively through the filtering update equations. In [**?**] and [**?**], different approaches to derive analytical expressions for objective functions of the same type as (5.9) are provided. They, however, consider the case of a *linear* model. Some methods for obtaining ML estimates of parameters in nonlinear models are explained in the tutorial by [**?**].

Instead of deriving analytical expressions for the gradient of (5.9), it is also possible to compute a numerical approximation of the gradient. Numerical gradients can be used in a number of different optimization algorithms, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, see *e.g.* [**?**]. Similar to the Gauss-Newton method, in the BFGS method, the parameters are iteratively updated until convergence. However, instead of using the Hessian approximation (4.8), BFGS iteratively estimates the Hessian using information from previous iterations. Hence, solving (5.9) using BFGS with numerical gradients, requires running at least $n_\theta + 1$ filtering algorithms for each iteration. These are required to evaluate the objective function and to compute the numerical gradients. More evaluations can be necessary to compute a step length, see also §4.1.

Example 5.2 (ML estimates of the gyroscope bias) *To obtain ML estimates of the gyroscope bias, we run the EKF with orientation deviation states from Algorithm 4 to obtain* $\hat{y}_{t|t-1}(\delta_\omega)$ *and* $S_t(\delta_\omega)$ *for a given value of* δ_ω*. This allows us to evaluate the objective function in* (5.9)*. To compute* $\hat{\delta}_\omega$*, the optimization problem* (5.9) *is solved iteratively using BFGS.*

5.3 Orientation estimation with an unknown gyroscope bias

We estimate the gyroscope bias in simulated data as described in §4.5 and illustrated in Figure 4.4. Compared to the data presented in §4.5, however, a constant gyroscope bias to be estimated is added. Using Monte Carlo simulations of this data, we illustrate a few specific features of the different ways to estimate the bias.

First, we focus on obtaining MAP estimates of the bias using the smoothing and filtering approaches as described in §5.1. We simulate the measurement noise as described in §4.5 and simulate the gyroscope bias as

$$\delta_\omega \sim \mathcal{N}(0, \sigma_{\delta_\omega}^2 \mathcal{I}_3), \qquad \sigma_{\delta_\omega} = 5 \cdot 10^{-2}. \tag{5.10}$$

Note that σ_{δ_ω} is a factor 10 larger than the value discussed in §3.6 to clearly illustrate the effect of the presence of a gyroscope bias. The priors $p(\theta)$ and $p(\theta_1)$ in the smoothing and filtering algorithms are set equal to the distribution in (5.10). The covariance of the random walk model (5.5c) is set as $\Sigma_{\delta_{\omega,t}} = \sigma_{\delta_{\omega,t}}^2 \mathcal{I}_3$ with $\sigma_{\delta_{\omega,t}} = 1 \cdot 10^{-10}$. This small value ensures that after convergence, the bias estimate is quite constant. The resulting mean RMSEs of the orientation over 100 Monte Carlo simulations are summarized in Table 5.1. Since the filtering algorithms typically have similar characteristics as discussed in §4.5, we only consider the EKF with orientation deviation states here. Comparing these results to the ones presented in Table 4.3, the RMSEs of the smoothing optimization algorithm are almost the same as when there was no gyroscope bias present. However, the filtering results are worse. This is because the bias needs some time to be properly estimated. This is illustrated in Figure 5.1 where the gyroscope bias estimates and their uncertainties are shown for the filtering algorithm.

A major difference between the MAP and the ML approaches, is that the MAP takes into account a prior on the gyroscope bias. We analyze the effect of this prior using 500 Monte Carlo simulations, simulating the gyroscope bias to be $\delta_\omega = \begin{pmatrix} 0.05 & 0.01 & -0.04 \end{pmatrix}^\mathsf{T}$ rad/s. We study the estimated gyroscope

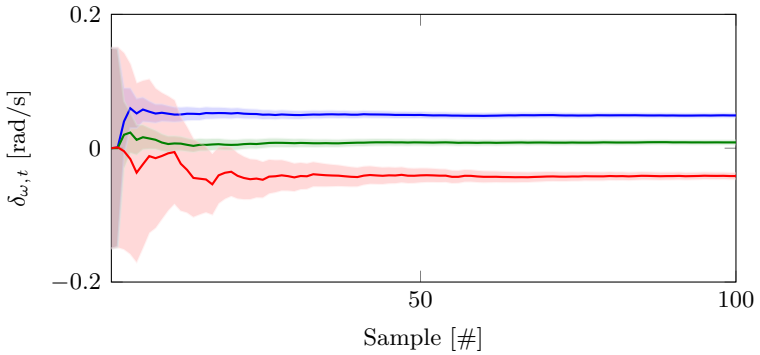


Figure 5.1: Filtering estimates of the gyroscope bias and their 3σ confidence intervals with the x-, y- and z-components in blue, green and red, respectively. Note that the estimates for only the first 100 samples are shown, to focus on the period in which the estimates converge.

biases using ML estimation, and using MAP estimation by including the gyroscope bias as an unknown in the smoothing algorithm. The smoothing algorithm assumes two different priors on the gyroscope bias $\delta_\omega \sim \mathcal{N}(0, \sigma_{\delta_\omega}^2 \mathcal{I}_3)$. In the first case, the prior on the gyroscope bias can well describe the data ($\sigma_{\delta_\omega} = 0.05$). In the other case, the prior is too tight ($\sigma_{\delta_\omega} = 1 \cdot 10^{-3}$). The mean and standard deviations for the gyroscope bias estimates are summarized in Table 5.2. As can be seen, when the prior is chosen appropriately, the ML and MAP estimates are comparable. If the prior is too tight, the MAP estimates can be seen to be biased towards zero.

Table 5.2: Mean and standard deviation of the gyroscope estimates over 500 Monte Carlo simulations with $(0.05 \;\; 0.01 \;\; -0.04)^\mathsf{T}$ rad/s. Considered are the cases of ML estimation and MAP estimation by including the gyroscope bias as an unknown in a smoothing algorithm with a prior on the gyroscope bias of $\delta_\omega \sim \mathcal{N}(0, \sigma_{\delta_\omega}^2 \mathcal{I}_3)$.

RMSE	Mean $\hat{\delta}_\omega$ $(\cdot 10^2)$			Standard deviation $\hat{\delta}_\omega$ $(\cdot 10^4)$		
	x	y	z	x	y	z
ML	5.0	1.0	-4.0	5.1	5.3	6.4
MAP $\sigma_{\delta_\omega} = 0.05$	5.0	1.0	-4.0	5.1	5.3	6.4
MAP $\sigma_{\delta_\omega} = 1 \cdot 10^{-3}$	3.9	0.8	-2.8	4.1	4.0	4.7

5.4 Identifiability

Parameters are said to be *identifiable* if it is possible to determine a unique parameter value from the data and if this value is equal to the true value [**?**]. The concept of identifiability is closely related to the concept of *observability* which is concerned with the question of if the time-varying states can be determined from the available data [**?**]. The states discussed in Chapter 4 are typically observable. Identifiability, however, becomes of concern when estimating calibration parameters. Specifically, in many applications, certain parameters are not identifiable when the sensor is completely stationary and sufficient excitation in terms of change in position and orientation is needed to make the parameters identifiable. This is illustrated in Example 5.3 for the case of identifiability of the gyroscope bias.

Example 5.3 (Identifiability of the gyroscope bias) *We consider the example of orientation estimation using only inertial measurements in the presence of a gyroscope bias. We simulate data as described in §4.5. The filtering estimates of the gyroscope bias and their uncertainties from an EKF with orientation deviation states are shown in Figure 5.2. Using only inertial measurements, the gyroscope bias of the sensor's z-axis is not identifiable when the sensor is placed horizontally. However, when the sensor is rotated, the accelerometer provides information that aids the estimation and the bias can be seen to converge. Note the difference with Figure 5.1, where only the first 100 samples were displayed*

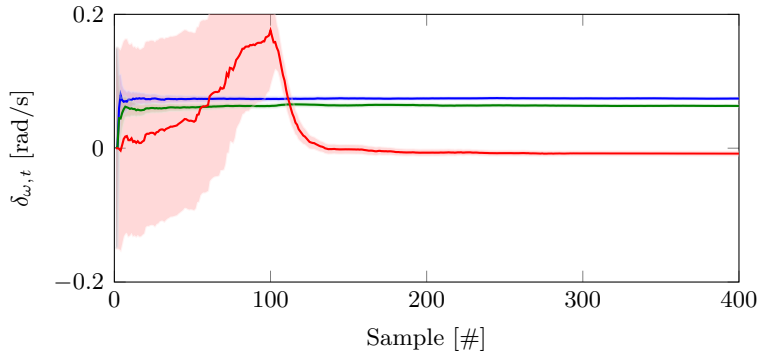


Figure 5.2: Filtering estimates of the gyroscope bias and their 3σ confidence intervals with the x-, y- and z-components in blue, green and red, respectively. The estimates are obtained using an EKF and inertial (but no magnetometer) measurements. As can be seen, the estimates of the gyroscope bias in the x- and y-axes converge quickly while the estimates of the gyroscope bias in the z-axis only start to converge after 100 samples, when the sensor starts to rotate around the x-axis.

and the bias estimates in the z-axis converged significantly faster due to the inclusion of magnetometer measurements.

Chapter 6

Sensor Fusion Involving Inertial Sensors

In this tutorial, we have presented a number of algorithms and modeling possibilities for position and orientation estimation using inertial sensors. The algorithms derived in Chapter 4 are based on the simple state space models (3.74) and (3.75). These models assume the availability of inertial measurements supplemented with either magnetometer or position measurements. In this chapter, we will illustrate how the same or similar algorithms can be used when additional and/or different supplementary sensors are available and discuss a number of possible extensions. The models used in this chapter will be more complex, but the information provided by the inertial sensors will remain one of the basic building blocks. We are not aiming for a full coverage, since that would require too much space. Instead we provide a number of case studies to illustrate what can be done when it comes to including the developments from Chapters 1–5 into a more complex setting. The case studies are the modeling of a non-Gaussian noise distribution to make use of radio-based time of arrival measurements (§6.1), the fusion of information from inertial sensors with cameras (§6.2), and motion capture based on information from multiple inertial sensors in combination with information about the human motion (§6.3 and §6.4).

6.1 Non-Gaussian noise distributions: time of arrival measurements

In the algorithms presented in Chapter 4, we assumed that the measurement noise of the sensors is properly described using a Gaussian distribution. This was shown to be a fair assumption for the inertial measurement noise based on experimental data in Example 2.3. In this section we will illustrate the possibility of modeling non-Gaussian measurement noise using a practical example. We will also describe possible adaptations of the algorithms presented in Chapter 4 to incorporate non-Gaussian noise models.

In the state space model (3.74), we assumed the presence of position measurements with additive Gaussian noise. Many sensors, however, do not directly measure the position, see *e.g.* [**?**]. An example of a type of measurements that can be obtained are *time of arrival (TOA)* measurements from a UWB system. These TOA measurements provide information about when a radio pulse sent out by a mobile transmitter arrives at a stationary receiver. This is graphically illustrated in Figure 6.1. These types of

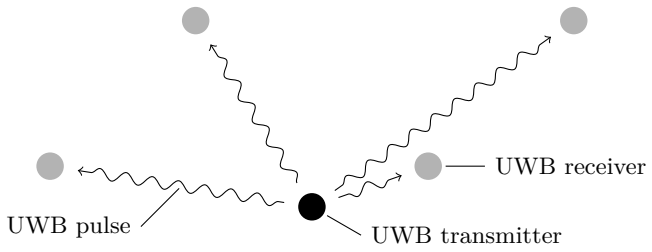


Figure 6.1: A UWB setup consisting of a number of stationary receivers obtaining time of arrival measurements of signal pulses originating from a mobile transmitter.

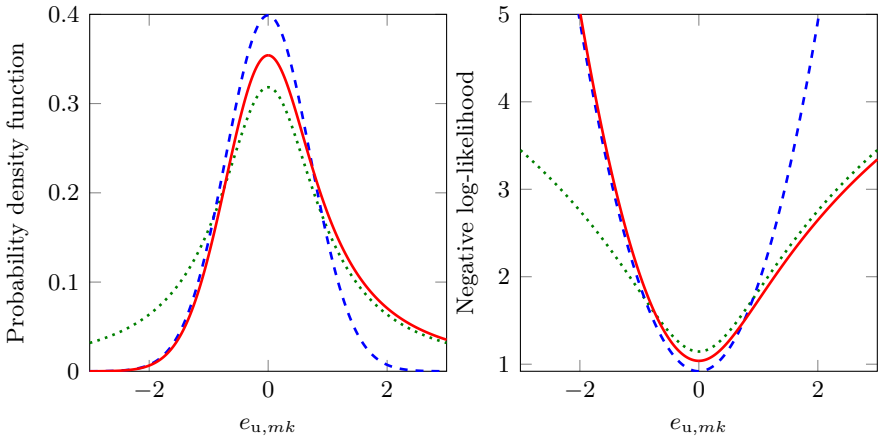


Figure 6.2: Probability density function (left) and negative log-likelihood (right) of a $\mathcal{N}(0,1)$ distribution (blue, dashed), a Cauchy$(0,1)$ distribution (green, dotted) and the asymmetric distribution (6.2) assuming $\sigma = \gamma = 1$ (red).

measurements can be modeled as

$$y_{\mathrm{u},mk} = \tau_k + \tfrac{1}{c}\|r_m - t_k\|_2 + \Delta\tau_m + e_{\mathrm{u},mk}. \tag{6.1}$$

Here, c denotes the speed of light, t_k is the position of the transmitter at the time of transmitting the k^{th} pulse, r_m is the position of the m^{th} receiver and $\Delta\tau_m$ is its clock-offset. The time when pulse k is transmitted is denoted by τ_k. Finally, $e_{\mathrm{u},mk}$ is measurement noise.

Using time of arrival measurements from a number of receivers it is possible to compute the position of the transmitter at the time when the pulse was sent. In many applications, the mobile transmitters have a much less accurate clock than the stationary receivers and τ_k is therefore assumed to be an unknown variable to be estimated. For a setup with M stationary transmitters, the position of the transmitter when transmitting the pulse k and the time when the pulse was transmitted can be estimated using multilateration [**?** **?** **?** **?**].

Due to non-line-of-sight conditions and/or multipath we expect a small number of measurements to be delayed. In [**?**], we present an approach that models the measurement noise $e_{\mathrm{u},mk}$ using a tailored asymmetric heavy-tailed distribution. Here, a heavy-tailed Cauchy distribution allows for measurement *delays*, while a Gaussian distribution excludes the physically unreasonable possibility of pulses traveling faster than the speed of light as

$$e_{\mathrm{u},mk} \sim \begin{cases} (2-\alpha)\,\mathcal{N}(0,\sigma^2) & \text{for } e_{\mathrm{u},mk} < 0, \quad (6.2a) \\ \alpha\,\text{Cauchy}(0,\gamma) & \text{for } e_{\mathrm{u},mk} \geq 0. \quad (6.2b) \end{cases}$$

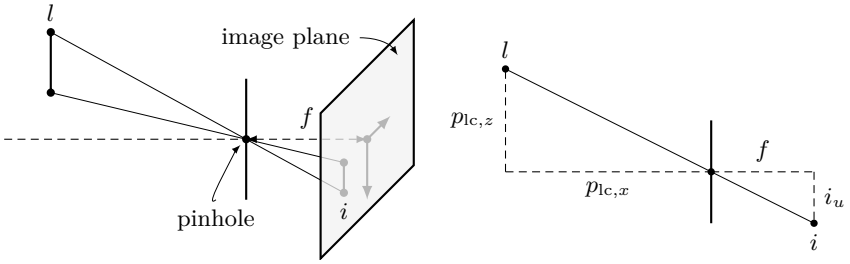
The presence of the constants α and $2-\alpha$ is motivated by the fact that the proposed asymmetric probability density function needs to integrate to one. The parameter α is defined as a function of σ and γ. The proposed asymmetric probability density function and its corresponding negative log-likelihood, given by

$$-\log p\left(e_{\mathrm{u},mk}\right) = \begin{cases} \mathcal{L}_G & \text{for } e_{\mathrm{u},mk} < 0, \quad (6.3a) \\ \mathcal{L}_C & \text{for } e_{\mathrm{u},mk} \geq 0, \quad (6.3b) \end{cases}$$

$$\mathcal{L}_G \triangleq \tfrac{e_{\mathrm{u},mk}^2}{2\sigma^2} + \tfrac{1}{2}\log\sigma^2 + \tfrac{1}{2}\log 2\pi - \log\left(2-\alpha\right),$$
$$\mathcal{L}_C \triangleq \log\left(1 + \tfrac{e_{\mathrm{u},mk}^2}{\gamma^2}\right) + \tfrac{1}{2}\log\gamma^2 + \log\pi - \log\alpha,$$

are both depicted in Figure 6.2 in red. For comparison, the Gaussian and Cauchy probability density functions are also depicted, in blue and green, respectively. Related studies have also modeled this presence of delayed measurements using skew-t distributions [**?** **?**], Gaussian mixture models [**?**] or by introducing a number of unknown delays [**?**].

The filtering and smoothing algorithms posed as optimization problems, see §4.1 and §4.2, can straightforwardly be adapted to include the asymmetric probability density function by using (6.3) in



(a) Illustration of the pinhole camera model where an object l results in an image i in the the position of the point l with respect to image plane. The image plane lies a distance f behind the pinhole.

(b) 2D visualization of the relation between the pinhole and the position of its image i.

Figure 6.3: Illustrations of the relation between a point or object l in the environment and its image i.

the objective function instead of the weighted least squares term representing the measurement model, see *e.g.* (4.72). Extended Kalman filters intrinsically assume that the process and measurement noise are Gaussian. For the extended Kalman filters implementations presented in §4.3.1 and §4.3.2, the extension to non-Gaussian noise is therefore less straightforward. Adaptations of the Kalman filter to non-Gaussian noise are, however, possible, see *e.g.* [**?**] and the references therein for a discussion on Kalman filtering in the presence of student's t distributed noise.

6.2 Using more complex sensor information: inertial and vision

In the model (3.74), we assumed that a sensor directly measuring the position was available. In §6.1, we instead considered the case of TOA measurements from a UWB system. In this section, we will focus on pose estimation using a camera and an IMU that are rigidly attached to each other. Inertial sensors and camera images have successfully been combined for pose estimation, see *e.g.* [**? ? ? ? ? ? ? ? ? ?**]. Application areas are for instance robotics, virtual reality and augmented reality.

To use camera images of the environment of the sensor to provide information about where the sensor is located in this environment, a number of *features* is typically extracted from each image. These features are subsequently associated with a point in the environment. Advanced feature extraction and association algorithms are available, see *e.g.* [**? ? ? ?**].

To be able to use the features and their associated points in the real world for pose estimation, a model is required that relates points in the image to the environment. A simple model that can be used for this is a pin-hole model. Such a model is illustrated in Figure 6.3(a), where an object l results in an image i in the image plane. The size of i depends on the size of the object, the distance of the object to the pinhole and on the *focal length* f. The focal length is an intrinsic property of a camera and is typically assumed to be known.

Let us now assume that the object l is simply a point and denote its position with respect to the pinhole center by $p_{\mathrm{lc}} = \begin{pmatrix} p_{\mathrm{lc},x} & p_{\mathrm{lc},y} & p_{\mathrm{lc},z} \end{pmatrix}^{\mathsf{T}}$. We denote the position of the object l in the image frame by $i = \begin{pmatrix} i_u & i_v \end{pmatrix}$. A two-dimensional illustration of this can be found in Figure 6.3(b). The position i can be expressed in terms of p_{lc} and the focal length f as

$$\begin{pmatrix} i_u \\ i_v \end{pmatrix} = f \begin{pmatrix} \frac{p^{\mathrm{c}}_{\mathrm{lc},z}}{p^{\mathrm{c}}_{\mathrm{lc},x}} & \frac{p^{\mathrm{c}}_{\mathrm{lc},z}}{p^{\mathrm{c}}_{\mathrm{lc},y}} \end{pmatrix}^{\mathsf{T}}. \tag{6.4}$$

Here, the superscript c indicates that the position p_{lc} is expressed in the *camera frame* c. The origin of the camera frame is located at the pinhole and its axes are aligned with the image plane. It is possible to express the position p_{lc} in terms of the position of the camera and the position of the object l as

$$p^{\mathrm{c}}_{\mathrm{lc},t} = R^{\mathrm{cn}}_t \left(p^{\mathrm{n}}_{\mathrm{l}} - p^{\mathrm{n}}_{\mathrm{c},t} \right). \tag{6.5}$$

Here, $p^{\mathrm{n}}_{\mathrm{c},t}$ denotes the position of the camera at time t expressed in the navigation frame. The rotation from the navigation frame n to the camera frame c at time t is denoted by R^{cn}_t. Finally, the position of the point l in the navigation frame is denoted by $p^{\mathrm{n}}_{\mathrm{l}}$. Note that the objects in the environment are assumed to be stationary, *i.e.* $p^{\mathrm{n}}_{\mathrm{l}}$ does not change over time. Assuming that the position of the object $p^{\mathrm{n}}_{\mathrm{l}}$ is known, using (6.4) and (6.5), the image i provides information about the position $p^{\mathrm{n}}_{\mathrm{c},t}$ and the orientation R^{cn}_t of the camera.

To be able to combine the information from the camera images with inertial measurements, it is possible to express the position $p_{\mathrm{c},t}$ and the orientation R_t^{cn} of the camera in terms of the position p_t^{n} and the orientation R_t^{bn} of the body frame as

$$p_{\mathrm{c},t}^{\mathrm{n}} = p_t^{\mathrm{n}} + d_{\mathrm{cb}}^{\mathrm{n}}, \tag{6.6a}$$

$$R_t^{\mathrm{cn}} = R^{\mathrm{cb}} R_t^{\mathrm{bn}}. \tag{6.6b}$$

Here, the distance between the body frame and the camera frame is denoted by d_{cb} and the rotation between these two frames is denoted by R^{cb}. Both are assumed to be constant and can be determined using dedicated calibration algorithms [**?** **?** **?** **?** **?**]. The relation (6.5) can therefore be expressed in terms of R_t^{bn} and p_t^{n} as

$$p_{\mathrm{lc},t}^{\mathrm{c}} = R^{\mathrm{cb}} R_t^{\mathrm{bn}} \left(p_{\mathrm{l}}^{\mathrm{n}} - p_t^{\mathrm{n}}\right) - d_{\mathrm{cb}}^{\mathrm{c}}. \tag{6.7}$$

Using (6.4), the measurement model (3.74b) can now be replaced with measurements $y_{\mathrm{c},t}$ from the camera as

$$y_{\mathrm{c},t} = \begin{pmatrix} i_{u,t} \\ i_{v,t} \end{pmatrix} = f\left(\begin{matrix} \frac{p_{\mathrm{lc},z,t}^{\mathrm{c}}}{p_{\mathrm{lc},x,t}^{\mathrm{c}}} & \frac{p_{\mathrm{lc},z,t}^{\mathrm{c}}}{p_{\mathrm{lc},y,t}^{\mathrm{c}}} \end{matrix}\right)^{\mathsf{T}} + e_{\mathrm{c},t}, \tag{6.8}$$

where $e_{\mathrm{c},t}$ is measurement noise and $p_{\mathrm{lc},t}^{\mathrm{c}}$ is given by (6.7). Including this measurement model in (3.74) allows us to combine inertial sensors with camera images to estimate the position and orientation of the sensor.

Note that at each time instance there can be multiple measurements from the camera image. One version of this adapted state space model estimates the position and orientation of the sensor assuming that the position of the objects $p_{\mathrm{l}}^{\mathrm{n}}$ in the environment is known and constant. Alternatively, the positions $p_{\mathrm{l}}^{\mathrm{n}}$ for objects $l = 1, \ldots, L$ are considered to be unknown and part of the state vector. The latter is called *simultaneous localization and mapping (SLAM)*, where a map of the environment is built from measurements of a mobile sensor whose position is unknown and to be estimated. There exist many algorithms for doing SLAM, both using EKF and optimization-based approaches, see *e.g.* the tutorial papers [**?** **?**] or [**?**] for a more recent overview.

6.3 Including non-sensory information: inertial motion capture

In the models (3.74) and (3.75), we have only made use of information obtained through sensor data. In some situations, however, additional non-sensory information is available. An example of this is inertial motion capture.

In inertial sensor human body motion capture, IMUs placed on different body segments are used to estimate the body's pose. This information can be useful for rehabilitation or for improving sports performance. An example can be seen in Figure 6.4 where Olympic and world champion speed skating Ireen Wüst wears sensors on her body that give information about her posture while ice skating. One can imagine that she can use this information to analyze which angles her knees and hips should have to skate as fast as possible and if her posture changes when she gets tired. It is also possible to use the information about how a person moves for motion capture in movies and games. This was illustrated in Figure 1.2(b), where the actor Seth MacFarlane wears sensors on his body that measure his movements to animate the bear Ted. Inertial motion capture is successfully used in a large number of applications, see *e.g.* [**?** **?** **?**].

In [**?**], we present an algorithm for inertial motion capture which solves a smoothing optimization problem similar to (4.72). A significant part of the model used in the algorithm in [**?**] is the state space model for pose estimation presented in (3.74) and used in §4.6. This model is used for every sensor that is placed on the human body. The following adaptations are made to the model:

Placement of the sensors on the body segments. The position and orientation of each sensor can be expressed in terms of its position and orientation on the corresponding body segment. Ideally, the sensor would be rigidly attached to the segment. However, it is physically impossible to place the sensor directly on the bone. Since it needs to be placed on the soft tissue instead, the sensor will inevitably move slightly with respect to the bone. We therefore model the position and orientation of each sensor on its corresponding body segment as approximately constant.

Figure 6.4: Left: Olympic and world champion speed skating Ireen Wüst wearing sensors on her body. Right: graphical representation of the estimated orientation and position of her body segments.

Joints between the body segments. A number of equality constraints enforce the body segments to be connected at the joint locations at all times. Note that equality constraints can straightforwardly be included in optimization problems [**?** **?**].

Exclusion of magnetometer measurements. The magnetic field in indoor environments is not constant, see *e.g.* [**?**]. This is specifically of concern in motion capture applications, since the magnetic field measured at the different sensor locations is typically different [**?** **?** **?** **?**]. Because of this, we do not include magnetometer measurements in the model. The inclusion of the equality constraints enforcing the body segments to be connected allows us to do so, since incorporating these constraints, the sensor's *relative* position and orientation become observable as long as the subject is not standing completely still [**?**].

As before, we denote the time-varying states by x_t and the constant parameters by θ. However, to highlight the different parts of the model, we split the states x_t in states $x_t^{\mathrm{S}_i}$ pertaining to the sensor S_i and states $x_t^{\mathrm{B}_i}$ pertaining to the body segment B_i, for $i = 1, \ldots, N_S$. Here, N_S is the number of sensors attached to the body and it is assumed to be equal to the number of body segments that are modeled. Using this notation, the optimization problem that is solved in [**?**] can be summarized as

$$\begin{aligned} \min_{x_{1:N},\theta} \quad & \underbrace{-\sum_{t=2}^{N_T}\sum_{i=1}^{N_S} \log p(x_t^{\mathrm{S}_i} \mid x_{t-1}^{\mathrm{S}_i})}_{\text{Dynamics of the state } x_t^{\mathrm{S}_i}} \underbrace{-\sum_{t=1}^{N_T}\sum_{i=1}^{N_S} \log p(x_t^{\mathrm{B}_i} \mid x_t^{\mathrm{S}_i})}_{\substack{\text{Placement of sensor } \mathrm{S}_i \\ \text{on body segment } \mathrm{B}_i}} \\ & \underbrace{-\sum_{i=1}^{N_S} \log p(x_1^{\mathrm{S}_i}) - \sum_{i=1}^{N_S} \log p(\theta^{\mathrm{S}_i})}_{\text{Prior}} \\ \text{subj. to} \quad & c(x_{1:N}) = 0, \end{aligned} \tag{6.9}$$

where the number of time steps is denoted N_T and $c(x_{1:N})$ denote the equality constraints from assuming that the body segments are connected at the joints. Note the similarities and differences between the structure of (6.9) as compared to (3.3a) or (4.72). The problem (6.9) is formulated as a smoothing algorithm. In [**?**], a similar method is presented in which a sliding window approach is used to allow for online estimation. In EKF formulations it is not straightforward to include equality constraints. However, as discussed in [**?**], it is possible to include the equality constraints as a measurement update in the EKF instead.

Optionally, additional knowledge about the human body can be included in (6.9). For instance, in [**?**], we include the knowledge that for some joints, it is known that their rotation is (mainly) limited to one or two axes. An example of this is the knee which is a hinge joint, although it can in practice flex a little around the other axes too. The optimization problem for that case would include a term that minimizes the rotation around certain axes. In [**?**], a solution to the inertial motion capture problem is presented that includes additional biomechanical information. For instance, the shape of the body segments is approximated by a capsule and the sensors are assumed to be placed on the surface of the capsule. In [**?**], the focus is instead on estimation of human motion using a much smaller amount of sensors. Because

of the smaller number of sensors, the information about the human body and its motions is more crucial in their problem formulation.

6.4 Beyond pose estimation: activity recognition

In this tutorial we have focused on position and orientation estimation using inertial sensors. A separate but related field of research is that of activity recognition using inertial sensors, where the aim is to determine the type of motion that is performed by a person wearing or carrying a device containing inertial sensors, see [**?** **?**] for an overview. Since these sensors have become smaller, cheaper and more publicly available over the years, the amount of data that can be used for activity recognition has increased dramatically. For instance, inertial measurements collected from a smartphone can be used to classify daily activities. It is also possible to classify these activities using measurements from fitness and activity trackers, see *e.g.* [**?** **?** **?**]. Standard activities which can be classified are for instance walking, running and sleeping, but there is a large number of other activities that one can think of as well.

A simple example of activity recognition that is relevant for the motion capture application discussed in §6.3, is footstep detection. As mentioned in §6.3, the *relative* position and orientation of the body can be determined using inertial sensors and a biomechanical model about the body segments being connected. However, the *absolute* position can not be determined from this information. Because of that, it is not possible to determine if the subject is standing on the floor or floating in the air by only using the model (6.9). Including the knowledge that during standing, walking and running, the feet are stationary on the floor during certain periods of time solves this problem [**?** **?**]. It is possible to detect these stationary periods for example by identifying when both the angular velocity and the sensor's acceleration are approximately zero, see [**?**] for a comparison between different detection methods. The knowledge that the sensor is stationary can subsequently be included in (6.9) by adding a term to the cost function modeling the velocity of the sensor as being almost zero during these time periods. These are so-called zero velocity updates (ZUPTs) [**?** **?**].

Footstep detection is a very simple example where pose estimation can benefit from activity recognition. Another example is presented in [**?**] where maps of indoor environments are build using inertial measurements from sensors placed on the human body, very similar to the approach presented in §6.3. In addition to the model presented in §6.3 and footstep detection, motions are detected of the subject opening and closing doors. Combining this information opens up for possibilities of performing SLAM. In conclusion, activity recognition can serve as a source of additional information that can be included in our models for pose estimation. Conversely, in [**?** **?**] it is argued that activity recognition can also benefit from combination with pose estimation.

Chapter 7

Concluding Remarks

The goal of this tutorial was not to give a complete overview of all algorithms that can be used for position and orientation estimation. Instead, our aim was to give a pedagogical introduction to the topic of position and orientation estimation using inertial sensors, allowing newcomers to this problem to get up to speed as fast as possible by reading one paper. By integrating the inertial sensor measurements (so-called dead-reckoning), it is possible to obtain information about the position and orientation of the sensor. However, errors in the measurements will accumulate and the estimates will drift. Because of this, to obtain accurate position and orientation estimates using inertial measurements, it is necessary to use additional sensors and additional models. In this tutorial, we have considered two separate estimation problems. The first is orientation estimation using inertial and magnetometer measurements, assuming that the acceleration of the sensor is approximately zero. Including magnetometer measurements removes the drift in the heading direction (as illustrated in Example 4.2), while assuming that the acceleration is approximately zero removes the drift in the inclination. The second estimation problem that we have considered is pose estimation using inertial and position measurements. Using inertial measurements, the position and orientation estimates are coupled and the position measurements therefore provide information also about the orientation.

A number of algorithms for position and orientation estimation have been introduced in Chapter 4. These include smoothing and filtering solved as an optimization problem, two different extended Kalman filter implementations and a version of the complementary filter. The filtering approaches use the data up to a certain time t to estimate the position and orientation at this time t. Smoothing instead makes use of all data from time $t = 1, \ldots, N$. In general, using all data to obtain the estimates will naturally lead to better estimates. The filtering approaches can be seen to be quite uncertain about their estimates for the first samples and require some time to "converge" to accurate estimates. This is even more pronounced in the presence of calibration parameters as discussed in Chapter 5. Although smoothing algorithms give better estimates, practical applications might not allow for computing smoothing estimates because of computational limitations or real-time requirements. For the examples discussed in this paper, the optimization-based filtering algorithm and the EKF with orientation deviation states perform very similarly. The EKF with quaternion states, however, was able to handle wrong initial orientations less well as shown in Table 4.4. Furthermore, it underestimated the uncertainty in the heading direction in the absence of magnetometer measurements, see also Example 4.5. The complementary filter is a good alternative to these methods, but its estimation accuracy decreases when orientation estimates from the accelerometer and magnetometer measurements are of significantly better quality in the inclination than in the heading direction. In Chapter 6 we have discussed possible extensions to the simple models considered earlier in this tutorial. One of the benefits of the optimization-based approaches is that these extensions can straightforwardly be incorporated into the framework.

Apart from the differences between the estimation algorithms discussed in this tutorial, it can also be concluded that the position and orientation estimation problems using inertial sensors are actually quite forgiving. Any of the algorithms introduced in this tutorial can give reasonably good estimates with fairly little effort. However, careful modeling is important since the quality of the estimates of the algorithms highly depends on the validity of the models. This was illustrated in Example 4.3 for the influence of magnetic material in the vicinity of the sensor on the quality of the orientation estimates. In recent years, inertial sensors have undergone major developments. The quality of their measurements has improved while their cost has decreased, leading to an increase in availability. Furthermore, available computational resources are steadily increasing. Because of these reasons, we believe that inertial sensors

can be used for even more diverse applications in the future.

Chapter 8

Notation and Acronyms

Notation	Coordinate frame
b	Body frame
e	Earth frame
i	Inertial frame
n	Navigation frame

Notation	Meaning
x_t	State vector x at time t.
y_t	Measurement vector y at time t.
θ	Parameter vector.
$x_{1:N}$, $y_{1:N}$	States and measurements for $t = 1, \ldots, N$.
$\hat{x}$	Estimate of x.
q^{nb}	Unit quaternion representing the rotation from the b-frame to the n-frame.
R^{nb}	Rotation matrix representing the rotation from the b-frame to the n-frame.
x^{n}	Vector x expressed in the n-frame.
$\mathbb{R}^N$	Set of real numbers in N dimensions.
$SO(3)$	Special orthogonal group in three dimensions.
$\in$	Element of.
$\subseteq$	Subset of.
$\emptyset$	Empty set.
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean μ and covariance Σ.
$\mathcal{N}(x\,;\,\mu, \Sigma)$	Random variable x that is normal distributed with mean μ and covariance Σ
$p(\,\cdot\,)$	Probability density function.
$p(a, b)$	Joint probability of a and b.
$p(a \mid b)$	Conditional distribution of a given b.
$\sim$	Distributed according to.
$\propto$	Proportional to.
$\mathcal{I}_n$	Identity matrix of size $n \times n$.
$0_{m \times n}$	Zero matrix of size $m \times n$.

Notation	Meaning
$\arg\max$	Maximizing argument.
$\arg\min$	Minimizing argument.
$a \cdot b$	Inner product of the vectors a and b.
$a \times b$	Cross product of the vectors a and b.
$[a\times]$	Cross product matrix of the vector a as defined in (3.14).
$\|a\|_2$	Two-norm of the vector a.
$\|a\|_W$	Weighted norm of the vector a.
$\bar{a}$	Quaternion representation of the vector a as defined in (3.26).
A^T	Transpose of the matrix A.
A^{-1}	Inverse of the matrix A.
$\det A$	Determinant of the matrix A.
$\exp_\mathsf{R}(\eta)$	Mapping from the vector η to a rotation matrix as defined in (3.37b).
$\log_\mathsf{R}(R)$	Mapping from the rotation matrix R to a vector as defined in (3.39b).
q^c	Conjugate of the unit quaternion q as defined in (3.25).
$\odot$	Quaternion multiplication as defined in (3.27).
q^L	Left quaternion multiplication matrix of the quaternion q as defined in (3.28).
q^R	Right quaternion multiplication matrix of the quaternion q as defined in (3.28).
$\exp_\mathsf{q}(\eta)$	Mapping from the vector η to a unit quaternion as defined in (3.37a).
$\log_\mathsf{q}(q)$	Mapping from the unit quaternion q to a vector as defined in (3.39a).

Acknowledgements

This research was financially supported by the projects *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524), *NewLEADS - New Directions in Learning Dynamical Systems* (Contract number: 621-2016-06079) and *CADICS*, all funded by the Swedish Research Council, the project *ASSEMBLE* (Contract number: RIT15-0012) funded by the Swedish Foundation for Strategic Research (SSF) and the EPSRC grant *Autonomous behaviour and learning in an uncertain world* (Grant number: EP/J012300/1). High accuracy reference measurements are provided through the use of the Vicon real-time tracking system courtesy of the UAS Technologies Lab, Artificial Intelligence and Integrated Computer Systems Division (AIICS) at the Department of Computer and Information Science (IDA), Linköping University, Sweden `http://www.ida.liu.se/divisions/aiics/aiicssite/index.en.shtml`. The authors would also like to thank Fredrik Olsson, Michael Lorenz, Dr. Oliver Woodford, Dr. Gustaf Hendeby, Prof. Fredrik Gustafsson and Prof. Magnus Herberthson for comments and suggestions that greatly improved this tutorial.

Appendix A

Orientation Parametrizations

In §A.1 of this appendix, we will summarize some important results on quaternion algebra that we make frequent use of throughout this tutorial. In §A.2, we summarize some results on how to convert between the different orientation parametrizations.

A.1 Quaternion algebra

A quaternion is a 4-dimensional vector q,

$$q = \begin{pmatrix} q_0 & q_v^\mathsf{T} \end{pmatrix}^\mathsf{T} = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix}^\mathsf{T}. \tag{A.1}$$

A special case is the unit quaternion, for which $\|q\|_2 = 1$. We use unit quaternions as a parametrization of orientations. An example of a quaternion that is not a unit quaternion and that we frequently encounter in this tutorial is the quaternion representation of a vector. For a vector v, its quaternion representation is given by

$$\bar{v} = \begin{pmatrix} 0 \\ v \end{pmatrix}. \tag{A.2}$$

The rotation of a vector v by a unit quaternion q can be written as

$$q \odot \bar{v} \odot q^\mathsf{c}. \tag{A.3}$$

Here, the quaternion multiplication $\odot$ of two quaternions p and q is defined as

$$p \odot q = \begin{pmatrix} p_0 q_0 - p_v \cdot q_v \\ p_0 q_v + q_0 p_v + p_v \times q_v \end{pmatrix}. \tag{A.4}$$

This can alternatively be defined in terms of the left and right multiplication matrices

$$p \odot q = p^\mathsf{L} q = q^\mathsf{R} p, \tag{A.5}$$

with

$$p^\mathsf{L} \triangleq \begin{pmatrix} p_0 & -p_v^\mathsf{T} \\ p_v & p_0 \mathcal{I}_3 + [p_v \times] \end{pmatrix}, \qquad q^\mathsf{R} \triangleq \begin{pmatrix} q_0 & -q_v^\mathsf{T} \\ q_v & q_0 \mathcal{I}_3 - [q_v \times] \end{pmatrix}, \tag{A.6}$$

where $[q_v \times]$ denotes the cross product matrix

$$[q_v \times] = \begin{pmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{pmatrix}. \tag{A.7}$$

The quaternion conjugate is given by

$$q^\mathsf{c} = \begin{pmatrix} q_0 \\ -q_v \end{pmatrix}. \tag{A.8}$$

Hence, the rotation of the vector v is given by

$$\begin{aligned} q \odot \bar{v} \odot q^{\mathsf{c}} &= q^{\mathsf{L}} \left(q^{\mathsf{c}}\right)^{\mathsf{R}} \bar{v} \\ &= \begin{pmatrix} q_0 & -q_v^{\mathsf{T}} \\ q_v & q_0 \mathcal{I}_3 + [q_v \times] \end{pmatrix} \begin{pmatrix} q_0 & -q_v^{\mathsf{T}} \\ q_v & q_0 \mathcal{I}_3 - [q_v \times] \end{pmatrix} \begin{pmatrix} 0 \\ v \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0_{1\times 3} \\ 0_{3\times 1} & q_v q_v^{\mathsf{T}} + q_0^2 \mathcal{I}_3 + 2q_0[q_v \times] + [q_v \times]^2 \end{pmatrix} \begin{pmatrix} 0 \\ v \end{pmatrix}. \end{aligned} \tag{A.9}$$

A.2 Conversions between different parametrizations

A quaternion q can be converted into a rotation matrix R as

$$\begin{aligned} R &= q_v q_v^{\mathsf{T}} + q_0^2 \mathcal{I}_3 + 2q_0[q_v \times] + [q_v \times]^2 \\ &= \begin{pmatrix} 2q_0^2 + 2q_1^2 - 1 & 2q_1 q_2 - 2q_0 q_3 & 2q_1 q_3 + 2q_0 q_2 \\ 2q_1 q_2 + 2q_0 q_3 & 2q_0^2 + 2q_2^2 - 1 & 2q_2 q_3 - 2q_0 q_1 \\ 2q_1 q_3 - 2q_0 q_2 & 2q_2 q_3 + 2q_0 q_1 & 2q_0^2 + 2q_3^2 - 1 \end{pmatrix}. \end{aligned} \tag{A.10}$$

Note the similarity with (A.9), where the rotation of the vector v can equivalently be expressed as Rv. Conversely, a rotation matrix

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}, \tag{A.11}$$

can be converted into a quaternion as

$$q_0 = \tfrac{\sqrt{1+\operatorname{Tr} R}}{2}, \qquad q_v = \tfrac{1}{4q_0} \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix}. \tag{A.12}$$

Note that a practical implementation needs to take care of the fact that the conversion (A.12) only leads to sensible results if $1 + \operatorname{Tr} R > 0$ and $q_0 \neq 0$. To resolve this issue, the conversion is typically performed in different ways depending on the trace of the matrix R and its diagonal values, see *e.g.* [**?**].

A rotation vector η can be expressed in terms of a unit quaternion q via the quaternion exponential as

$$q = \exp_{\mathsf{q}} \eta = \begin{pmatrix} \cos \|\eta\|_2 \\ \frac{\eta}{\|\eta\|_2} \sin \|\eta\|_2 \end{pmatrix}. \tag{A.13}$$

Note that any practical implementation needs to take care of the fact that this equation is singular at $\eta = 0$, in which case $\exp_{\mathsf{q}} \eta = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^{\mathsf{T}}$. The inverse operation is executed by the quaternion logarithm,

$$\eta = \log_{\mathsf{q}} q = \tfrac{\arccos q_0}{\sin \arccos q_0} q_v = \tfrac{\arccos q_0}{\|q_v\|_2} q_v. \tag{A.14}$$

Note that this equation is singular at $q_v = 0$. In this case, $\log q$ should return $0_{3\times 1}$.

The rotation vector η can also be converted into a rotation matrix as

$$R = \exp_{\mathsf{R}} \eta = \exp\left([\eta \times]\right), \qquad \eta = \log_{\mathsf{R}} R = \begin{pmatrix} (\log R)_{32} \\ (\log R)_{13} \\ (\log R)_{21} \end{pmatrix}, \tag{A.15}$$

where $\log R$ is the matrix logarithm and $\log_{\mathsf{R}}$ and $\exp_{\mathsf{R}}$ are the mappings introduced in (3.37) and (3.39).

A rotation in terms of Euler angles can be expressed as a rotation matrix R as

$$\begin{aligned} R &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{pmatrix}. \end{aligned} \tag{A.16}$$

The rotation matrix R can be converted into Euler angles as

$$\psi = \tan^{-1}\left(\frac{R_{12}}{R_{11}}\right) = \tan^{-1}\left(\frac{2q_1q_2-2q_0q_3}{2q_0^2+2q_1^2-1}\right), \tag{A.17a}$$

$$\theta = -\sin^{-1}\left(R_{13}\right) = -\sin^{-1}\left(2q_1q_3+2q_0q_2\right), \tag{A.17b}$$

$$\phi = \tan^{-1}\left(\frac{R_{23}}{R_{33}}\right) = \tan^{-1}\left(\frac{2q_2q_3-2q_0q_1}{2q_0^2+2q_3^2-1}\right). \tag{A.17c}$$

Using the relations presented in this section, it is possible to convert the parametrizations discussed in §3.2 into each other.

Appendix B

Pose Estimation

In this appendix, we will introduce the necessary components to extend Algorithms 1–4 to pose estimation algorithms using the state space model (3.74).

B.1 Smoothing in an optimization framework

In §4.6, we presented the smoothing optimization problem for pose estimation. To adapt Algorithm 1, the derivatives (4.14a)–(4.14c) in combination with the following derivatives are needed

$$\frac{\partial e_{\mathrm{p,i}}}{\partial p_1^{\mathrm{n}}} = \mathcal{I}_3, \qquad \frac{\partial e_{\mathrm{v,i}}}{\partial v_1^{\mathrm{n}}} = \mathcal{I}_3, \qquad \frac{\partial e_{\mathrm{p},t}}{\partial p_t^{\mathrm{n}}} = -\mathcal{I}_3, \tag{B.1a}$$

$$\frac{\partial e_{\mathrm{p,a},t}}{\partial p_{t+1}^{\mathrm{n}}} = \tfrac{2}{T^2}\mathcal{I}_3, \qquad \frac{\partial e_{\mathrm{p,a},t}}{\partial p_t^{\mathrm{n}}} = -\tfrac{2}{T^2}\mathcal{I}_3,$$

$$\frac{\partial e_{\mathrm{p,a},t}}{\partial v_t^{\mathrm{n}}} = -\tfrac{1}{T}\mathcal{I}_3, \qquad \frac{\partial e_{\mathrm{p,a},t}}{\partial \eta_t^{\mathrm{n}}} = [\tilde{R}_t^{\mathrm{nb}} y_{\mathrm{a},t}\times], \tag{B.1b}$$

$$\frac{\partial e_{\mathrm{v,a},t}}{\partial v_{t+1}^{\mathrm{n}}} = \tfrac{1}{T}\mathcal{I}_3, \qquad \frac{\partial e_{\mathrm{v,a},t}}{\partial v_t^{\mathrm{n}}} = -\tfrac{1}{T}\mathcal{I}_3, \qquad \frac{\partial e_{\mathrm{v,a},t}}{\partial \eta_t^{\mathrm{n}}} = [\tilde{R}_t^{\mathrm{nb}} y_{\mathrm{a},t}\times]. \tag{B.1c}$$

B.2 Filtering in an optimization framework

To obtain position, velocity and orientation estimates in a filtering framework, the optimization problem (4.28) is adapted to

$$\begin{aligned}\hat{x}_t &= \arg\min_{x_t} -\log p(x_t \mid y_{1:t}) \\ &= \arg\min_{x_t} \|e_{\mathrm{f},t}\|_{P_{t|t-1}^{-1}} + \|e_{\mathrm{p},t}\|_{\Sigma_{\mathrm{p}}^{-1}},\end{aligned} \tag{B.2}$$

where $\|e_{\mathrm{p},t}\|_{\Sigma_{\mathrm{p}}^{-1}}$ is the position measurement model also used in the smoothing optimization problem presented in §4.6. Furthermore, $e_{\mathrm{f},t}$ is extended from (4.27) to be

$$e_{\mathrm{f},t} = \begin{pmatrix} p_t^{\mathrm{n}} - p_{t-1}^{\mathrm{n}} - T v_{t-1}^{\mathrm{n}} - \frac{T^2}{2}\left(R_{t-1}^{\mathrm{nb}} y_{\mathrm{a},t-1} + g^{\mathrm{n}}\right) \\ v_t^{\mathrm{n}} - v_{t-1}^{\mathrm{n}} - T\left(R_{t-1}^{\mathrm{nb}} y_{\mathrm{a},t-1} + g^{\mathrm{n}}\right) \\ \eta_t^{\mathrm{n}} - 2\log_{\mathsf{q}}\left(\hat{q}_{t-1}^{\mathrm{nb}} \odot \exp_{\mathsf{q}}(\tfrac{T}{2} y_{\omega,t-1}) \odot \tilde{q}_t^{\mathrm{bn}}\right) \end{pmatrix}, \tag{B.3}$$

where $\frac{\partial e_{\mathrm{f},t}}{\partial x_t} = \mathcal{I}_9$ and $x_t = \begin{pmatrix} p_t^{\mathsf{T}} & v_t^{\mathsf{T}} & (\eta_t^{\mathrm{n}})^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}$.

The covariance matrix $P_{t+1|t}$ is given by $P_{t+1|t} = F_t P_{t|t} F_t^{\mathsf{T}} + G_t Q G_t^{\mathsf{T}}$ with

$$F_t = \begin{pmatrix} \mathcal{I}_3 & T\mathcal{I}_3 & -\frac{T^2}{2}[\hat{R}_t^{\mathrm{nb}} y_{\mathrm{a},t}\times] \\ 0 & \mathcal{I}_3 & -T[\hat{R}_t^{\mathrm{nb}} y_{\mathrm{a},t}\times] \\ 0 & 0 & \mathcal{I}_3 \end{pmatrix}, \qquad G_t = \begin{pmatrix} \mathcal{I}_6 & 0 \\ 0 & T\tilde{R}_{t+1}^{(0)} \end{pmatrix}, \tag{B.4a}$$

$$Q = \begin{pmatrix} \Sigma_{\mathrm{a}} & 0 & 0 \\ 0 & \Sigma_{\mathrm{a}} & 0 \\ 0 & 0 & \Sigma_{\omega} \end{pmatrix}. \tag{B.4b}$$

Similar to the update of the linearization point in (4.31a), we also update the estimates of the position and velocity before starting the optimization algorithm, such that $e_{\mathrm{f},t}$ is equal to zero for iteration $k = 0$.

B.3 Extended Kalman filter with quaternion states

Following the notation in §4.3, the following matrices are needed for implementing the EKF for pose estimation with quaternion states

$$F_t = \begin{pmatrix} \mathcal{I}_3 & T\mathcal{I}_3 & \frac{T^2}{2} \left.\frac{\partial R^{\text{nb}}_{t|t}}{\partial q^{\text{nb}}_{t|t}}\right|_{q^{\text{nb}}_{t|t}=\hat{q}^{\text{nb}}_{t|t}} y_{\text{a},t} \\ 0 & \mathcal{I}_3 & T \left.\frac{\partial R^{\text{nb}}_{t|t}}{\partial q^{\text{nb}}_{t|t}}\right|_{q^{\text{nb}}_{t|t}=\hat{q}^{\text{nb}}_{t|t}} y_{\text{a},t} \\ 0 & 0 & \left(\exp_{\mathsf{q}}(\frac{T}{2} y_{\omega,t})\right)^{\mathsf{R}} \end{pmatrix}, \quad H = \mathcal{I}_3, \quad R = \Sigma_{\text{p}}, \tag{B.5a}$$

$$G_t = \begin{pmatrix} \mathcal{I}_6 & 0 \\ 0 & -\frac{T}{2}\left(\hat{q}^{\text{nb}}_{t|t}\right)^{\mathsf{L}} \frac{\partial \exp_{\mathsf{q}}(e_{\omega,t})}{\partial e_{\omega,t}} \end{pmatrix}, \quad Q = \begin{pmatrix} \Sigma_{\text{a}} & 0 & 0 \\ 0 & \Sigma_{\text{a}} & 0 \\ 0 & 0 & \Sigma_{\omega} \end{pmatrix}. \tag{B.5b}$$

B.4 Extended Kalman filter with orientation deviation states

In the time update of the pose estimation algorithm with orientation deviation states, the linearization point is again directly updated as in (4.49). The position and velocity states are updated according to the dynamic model (3.74a). Furthermore, the matrices Q, H and R from (B.5) are needed for implementing the EKF for pose estimation in combination with

$$F_t = \begin{pmatrix} \mathcal{I}_3 & T\mathcal{I}_3 & -\frac{T^2}{2}[\tilde{R}^{\text{nb}}_{t|t} y_{\text{a},t}\times] \\ 0 & \mathcal{I}_3 & -T[\tilde{R}^{\text{nb}}_{t|t} y_{\text{a},t}\times] \\ 0 & 0 & \mathcal{I}_3 \end{pmatrix}, \qquad G_t = \begin{pmatrix} \mathcal{I}_6 & 0 \\ 0 & T\tilde{R}_{t+1|t} \end{pmatrix}. \tag{B.6}$$

Appendix C

Gyroscope Bias Estimation

In this appendix, we will introduce the necessary components to extend Algorithms 1–4 to estimate an unknown gyroscope bias. Note that the measurement models are independent of the gyroscope bias. The dynamic models and time update equations are adapted as described below.

C.1 Smoothing in an optimization framework

To include a gyroscope bias in Algorithm 1, a prior on the bias is included, leading to an additional term in the objective function as

$$e_{\delta_\omega} = \delta_\omega, \qquad e_{\delta_\omega} \sim \mathcal{N}(0, \Sigma_{\delta_\omega}). \tag{C.1}$$

The derivatives in (4.14) are complemented with

$$\tfrac{\partial e_{\omega,t}}{\partial \delta_\omega} = \mathcal{I}_3, \qquad \tfrac{\partial e_{\delta_\omega}}{\partial \delta_\omega} = \mathcal{I}_3. \tag{C.2}$$

C.2 Filtering in an optimization framework

To include estimation of a gyroscope bias in Algorithm 2, (4.24) needs to take the gyroscope bias into account as

$$\tilde{q}_{t+1}^{\mathrm{nb},(0)} = \hat{q}_t^{\mathrm{nb}} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} - \hat{\delta}_{\omega,t})\right). \tag{C.3}$$

Modeling the dynamics of the gyroscope bias as a random walk, $e_{\mathrm{f},t}$ is extended from (4.27) to

$$e_{\mathrm{f},t} = \begin{pmatrix} \eta_t^{\mathrm{n}} - 2\log_{\mathsf{q}}\left(\hat{q}_{t-1}^{\mathrm{nb}} \odot \exp_{\mathsf{q}}(\tfrac{T}{2} y_{\omega,t-1}) \odot \tilde{q}_t^{\mathrm{bn}}\right) \\ \delta_{\omega,t} - \hat{\delta}_{\omega,t-1} \end{pmatrix}, \tag{C.4}$$

where $\frac{\partial e_{\mathrm{f},t}}{\partial x_t} = \mathcal{I}_6$ and $x_t = \begin{pmatrix} (\eta_t^{\mathrm{n}})^{\mathsf{T}} & \delta_{\omega,t}^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}$.

The covariance matrix $P_{t+1|t}$ is given by $P_{t+1|t} = F_t P_{t|t} F_t^{\mathsf{T}} + G_t Q G_t^{\mathsf{T}}$ with

$$F_t = \begin{pmatrix} \mathcal{I}_3 & -T\tilde{R}_{t+1}^{\mathrm{nb},(0)} \\ 0 & \mathcal{I}_3 \end{pmatrix}, \tag{C.5a}$$

$$G_t = \begin{pmatrix} T\tilde{R}_{t+1}^{\mathrm{nb},(0)} & 0 \\ 0 & \mathcal{I}_3 \end{pmatrix}, \qquad Q = \begin{pmatrix} \Sigma_\omega & 0 \\ 0 & \Sigma_{\delta_{\omega,t}} \end{pmatrix}. \tag{C.5b}$$

Note that a prior on the gyroscope bias at $t = 1$ also needs to be included.

C.3 Extended Kalman filter with quaternion states

To include estimation of an unknown gyroscope bias in Algorithm 3, the matrices for the time update of the EKF need to be adapted to

$$F_t = \begin{pmatrix} \left(\exp_{\mathsf{q}}(\tfrac{T}{2}(y_{\omega,t} - \hat{\delta}_{\omega,t|t}))\right)^{\mathsf{R}} & -\tfrac{T}{2}\left(\hat{q}^{\mathrm{nb}}_{t|t}\right)^{\mathsf{L}} \frac{\partial \exp_{\mathsf{q}}(\delta_{\omega,t})}{\partial \delta_{\omega,t}} \\ 0_{3\times 4} & \mathcal{I}_3 \end{pmatrix}, \tag{C.6a}$$

$$Q = \begin{pmatrix} \Sigma_\omega & 0 \\ 0 & \Sigma_{\delta_{\omega,t}} \end{pmatrix}, \qquad G_t = \begin{pmatrix} -\tfrac{T}{2}\left(\hat{q}^{\mathrm{nb}}_{t|t}\right)^{\mathsf{L}} \frac{\partial \exp_{\mathsf{q}}(e_{\omega,t})}{\partial e_{\omega,t}} & 0 \\ 0 & \mathcal{I}_3 \end{pmatrix}. \tag{C.6b}$$

Note that also for this algorithm a prior on the gyroscope bias needs to be included. Furthermore, the renormalization of the covariance in (4.44) needs to be adjusted to the size of the covariance matrix as

$$J_t = \begin{pmatrix} \frac{1}{\|\tilde{q}^{\mathrm{nb}}_{t|t}\|_2^3} \tilde{q}^{\mathrm{nb}}_{t|t} \left(\tilde{q}^{\mathrm{nb}}_{t|t}\right)^{\mathsf{T}} & 0 \\ 0 & \mathcal{I}_3 \end{pmatrix}. \tag{C.7}$$

C.4 Extended Kalman filter with orientation deviation states

To include estimation of an unknown gyroscope bias in Algorithm 4, the update of the linearization point in the time update takes the estimate of the gyroscope bias into account as

$$\tilde{q}^{\mathrm{nb}}_{t+1|t} = \tilde{q}^{\mathrm{nb}}_{t|t} \odot \exp_{\mathsf{q}}\left(\tfrac{T}{2}(y_{\omega,t} - \hat{\delta}_{\omega,t|t})\right). \tag{C.8}$$

The matrices F_t and G_t for the time update are adapted to

$$F_t = \begin{pmatrix} \mathcal{I}_3 & -T\tilde{R}^{\mathrm{nb}}_{t+1|t} \\ 0 & \mathcal{I}_3 \end{pmatrix}, \qquad G_t = \begin{pmatrix} T\tilde{R}^{\mathrm{nb}}_{t+1|t} & 0 \\ 0 & \mathcal{I}_3 \end{pmatrix}, \tag{C.9}$$

and Q is given in (C.5b). A prior on the gyroscope bias also needs to be included.

Bibliography