Institute for Computer Science XVII
Robotics

Master's thesis

IMU calibration and scale estimation improvement of existing VIO methods

Janis Nicolas Kaltenthaler

10. February 2023

First supervisor: Prof. Dr. Andreas Nüchter
Second supervisor: Prof. Dr.-Ing. Sergio Montenegro
Advisor: Dipl.-Ing. Michael Bleier

Acknowledgments

Special thanks belong to Prof. Dr. Andreas Nüchter, who made it possible for me to work on the topic. Furthermore, I thank Dipl.-Ing. Michael Bleier, who supported me with a lot of effort during the writing of the master thesis. Moreover, I thank my parents for making my studies possible and for proof-reading the thesis.

Abstract

In the field of autonomous motion and mapping, estimating the position and orientation of a moving body is of high significance. This includes in particular the latest information about acceleration, speed, traveled distance and orientation. A wide range of different technologies exists that are applied for this purpose. In case the state determination requires the use of inertial sensors, it is well known that these types of sensors tend to drift with time. This leads to increasing errors in the calculated trajectory, which is unacceptable if precise position information is required such as for mapping. This work presents a method to calibrate the deterministic and stochastic error parameters of inertial sensors measuring linear acceleration and angular velocity. No external equipment is necessary for this procedure, which simplifies its usage. Using the methods presented in this thesis, the error of the acceleration measurement will be reduced by a factor of 30.

Since remaining measurement errors continuously decrease the accuracy of position estimation by IMU, additional positioning sensors are usually used to improve the accuracy of the estimation. In the case of *visual inertial odometry* methods, a camera is used in addition to the inertial sensors to improve the position estimation. To combine the position measurements of several sensors, a transformation (i.e. translation and rotation) between the individual sensors should be known as exact as possible. Methods that use only the data from a camera and an IMU for position estimation, *monocular visual inertial odometry* methods, are especially dependent on a well calibrated IMU and the exact translation between the two sensors to estimate the scale as accurately as possible. The VIO software *VINS-Mono* used in the course of this thesis has disadvantages in estimating these parameters, especially as the translation during initialization is not taken into account.

In the second part of this thesis, methods are presented that improve the initialization of *VINS-Mono*. The translation and rotation data from camera images and IMU measurements are used to accurately determine the translation and rotation between a camera and an IMU. In the same course, the biases of the accelerometer and the gyroscope are also determined, as well as the direction of the gravity vector. The accurate estimation of these parameters allows the exact determination of the visual scale, which leads to a more accurate pose estimation. Using the methods presented in this thesis to initialize *VINS-Mono*, the accuracy of the resulting trajectory is significantly improved. This is especially important for the *UWSensor* project, which motivated the research on this topic. Here the accuracy of the estimated trajectory was improved, which led to a better point cloud result. Furthermore, the use of these methods only requires that there is enough structure in the environment of the camera for the detection of features.

Zusammenfassung

Im Hinblick auf die autonome Bewegung und Kartierung ist die Schätzung der Position und Orientierung eines sich fortbewegenden Systems von großer Bedeutung. Dazu gehören insbesondere die aktuellen Informationen über Beschleunigung, Geschwindigkeit, zurückgelegte Strecke und Orientierung. Es gibt eine Vielzahl von unterschiedlichen Technologien, die zu diesem Zweck eingesetzt werden. Erfordert die Zustandsbestimmung den Einsatz von Inertialsensoren, so ist bekannt, dass diese Art von Sensoren dazu neigen, mit der Zeit zu driften. Dies führt zu zunehmenden Fehlern in der berechneten Trajektorie, was inakzeptabel ist, wenn präzise Positionsinformationen benötigt werden, wie z.B. bei der Kartierung. In dieser Arbeit wird eine Methode zur Kalibrierung der deterministischen und stochastischen Fehlerparameter von Inertialsensoren vorgestellt, die lineare Beschleunigung und Winkelgeschwindigkeit messen. Für dieses Verfahren ist keine externe Ausrüstung erforderlich, was seine Anwendung vereinfacht. Mit den in dieser Arbeit vorgestellten Methoden wird der Fehler der Beschleunigungsmessung um einen Faktor von 30 reduziert.

Da die verbleibenden Messfehler die Genauigkeit der Positionsschätzung durch die IMU kontinuierlich verringern, werden in der Regel zusätzliche Positionssensoren eingesetzt, um die Genauigkeit der Schätzung zu verbessern. Bei den Methoden der visuellen inertialen Odometrie wird zusätzlich zu den Inertialsensoren eine Kamera verwendet, um die Positionsschätzung zu verbessern. Um die Positionsmessungen mehrerer Sensoren zu kombinieren, muss eine Transformation (d.h. Translation und Rotation) zwischen den einzelnen Sensoren so genau wie möglich bekannt sein. Methoden, die nur die Daten einer Kamera und einer IMU zur Positionsschätzung verwenden, *monokulare visuelle Inertial-Odometrie*-Methoden, sind besonders auf eine gut kalibrierte IMU und die exakte Translation zwischen den beiden Sensoren angewiesen, um die visuelle Skalierung so genau wie möglich zu schätzen. Die im Rahmen dieser Arbeit verwendete VIO Software *VINS-Mono* hat Nachteile bei der Schätzung dieser Parameter, insbesondere da die Translation während der Initialisierung nicht berücksichtigt wird.

Im zweiten Teil dieser Arbeit werden Methoden vorgestellt, die die Initialisierung von *VINS-Mono* verbessern. Die Translations- und Rotationsdaten aus Kamerabildern und IMU-Messungen werden verwendet, um die Translation und Rotation zwischen einer Kamera und einer IMU präzise zu bestimmen. Im gleichen Zuge wird auch der Bias des Beschleunigungssensors und des Gyroskops, sowie die Richtung des Gravitationsvektors bestimmt. Die genaue Schätzung dieser Parameter ermöglicht die exakte Bestimmung der visuellen Skalierung, was zu einer genaueren Posenschätzung führt. Mit den in dieser Arbeit vorgestellten Methoden zur Initialisierung von *VINS-Mono* wird die Genauigkeit der resultierenden Trajektorie deutlich verbessert. Dies ist besonders wichtig für das Projekt *UWSensor*, welches die Arbeiten an diesem Thema motiviert hat. Hier konnte die Genauigkeit der geschätzten Trajektorie verbessert werden, was zu einem besseren Punktwolkenergebnis führte. Darüber hinaus setzt die Anwendung dieser Methoden lediglich voraus, dass in der Umgebung der Kamera genügend Struktur für die Erkennung von Merkmalen vorhanden ist.

Contents

List of Figures

Glossary

ACRW Acceleration Random Walk

ADC Analog-to-digital converter

AHRS Attitude and Heading Reference System

AMCL Adaptive Monte Carlo Localization

ANRW Angle Random Walk

CAD Computer-Aided Design

FOG Fibre-Optic Gyroscope

GNSS Global Navigation Satellite System

IMU Inertial Measurement Unit

LM Levenberg–Marquardt

MARG Magnetic, Angular Rate, Gravity

MEMS Microelectromechanical System

PSD Power spectral density

RK4n Runge-Kutta 4^{th} order normalized

ROS Roboter Operating System

ROV Remote Operated Vehicle

RRW Rate Random Walk

VIO Visual Inertial Odometry

VRW Velocity Random Walk

Chapter 1

Introduction

Nowadays, the use of autonomous systems is widespread in many aspects of life. Each moving system relies on its exact position while navigating in its environment. A variety of different sensors and measurement methods exist to acquire motion data, such as acceleration, velocity, position and orientation data.

Imagine a drone that is supposed to move inside a building, e.g. to create a map of the surroundings. In order to create a map and navigate, the drone relies on an accurate estimate of its position, speed and orientation. Since a Global Navigation Satellite System (GNSS) receiver only works accurately when there is sufficient radio reception, i.e. neither underground nor indoors, this is not an option. As odometry is only suitable for a rolling system on a smooth surface, another technology must be used. For this example, it is decided to equip the drone with a camera and an IMU in order to estimate the necessary information.

Before using the measured values of the IMU, hence linear acceleration and angular velocity, the sensors need to be calibrated. This ensures that the measured values correspond as precisely as possible to the real values. For example, if the accelerometer suffers from a small measurement error in the acceleration reading, the necessary double integration results in a position error that cannot be neglected. To reduce this drift *monocular visual inertial odometry* is used to obtain pose information of the drone by combining the IMU readings with the camera images.

At this point, another challenge arises. Any localization algorithm that combines the data of several pose sensors requires the exact pose of the individual sensors to each other. The most trivial method to obtain the rotation and translation between the two sensors is to simply measure by hand the required transformation. Accuracy is limited here since, for example, the positions of the two senors within the housing of the IMU are not exactly known. Better accuracy is achieved by estimating the transformation with the given measured values of the camera and the IMU. For this purpose, various algorithms are available, such as the work of [20]. Here, a checkerboard pattern is used to determine the position and orientation change of the camera from the captured images. Since the rigidly connected IMU is subject to the same motion, the transformation between the two sensors is determined by aligning the trajectories of the camera

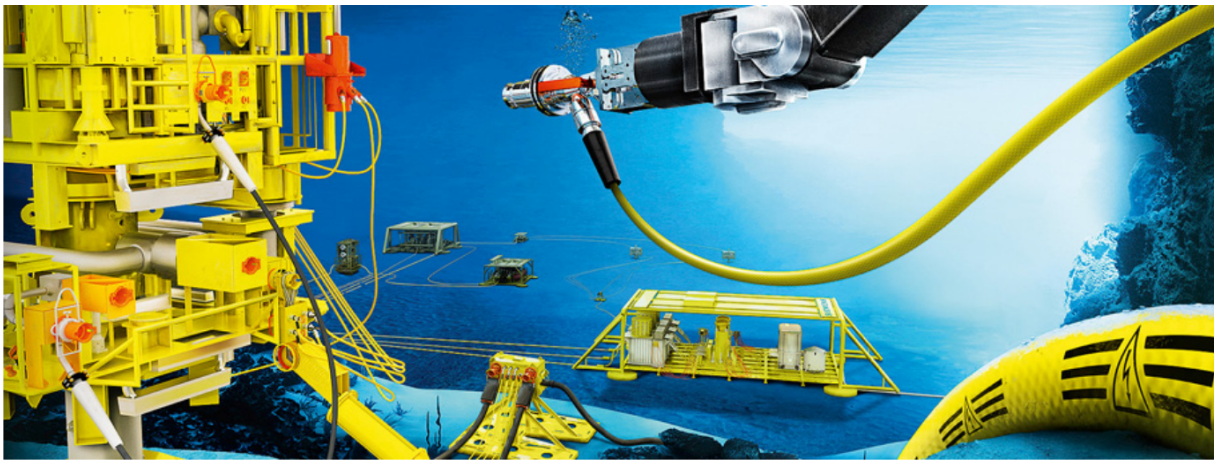

Figure 1.1: This rendering illustrates a possible underwater factory in the future. In this case, it could be a plant that produces crude oil.[1]

and the IMU. As described in [21], the use of external tools, such as checkerboard patterns or april tags, can be omitted here as well. In this case, features are extracted from the camera images to determine the poses of the camera. One advantage of this is that the transformation is always determined, as long as enough features are available in the environment.

Since any *monocular visual inertial odometry* method needs to obtain scale information from the IMU data in order to estimate the position, e.g. of the drone, it is essential to determine the transformation as accurately as possible. An erroneous transformation inevitably leads to a distorted scale and thus to poor localization. This thesis presents methods to calibrate the IMU and to determine the transformation between a camera and an IMU without additional tools to improve the pose estimation.

1.1 Problem Statement

For understanding the problem underlying this work, imagine an underwater factory in the future. Fig. 1.1 shows an example of this, where a factory for the mining of crude oil is depicted. If the necessary maintenance of this factory is to be carried out autonomously in the future, an Remote Operated Vehicle (ROV) is used. This work is based on the project *UWSensor* [2], which develops the sensor technology necessary for the ROV to carry out underwater surveying. A possible design of the sensor platform is shown in Fig. 1.2. For position estimation it is equipped with a color camera and an IMU with a Fibre-Optic Gyroscope (FOG). To build up a map of the environment with the help of the installed 3D scanner technology, the exact position of the sensor must be known at all times. The algorithm used to estimate the position, *VINS-Mono*, is described in paper [27]. As explained earlier, this algorithm has to determine scale information

[1] https://www.ingenieur.de/technik/fachbereiche/verkehr/statoil-siemens-entwickeln-fabriken-oel-gasfoerderung-meeresgrund/

[2] https://www.searenergy.com/portfolio-item/r-d-project-3d-uw-sensor/

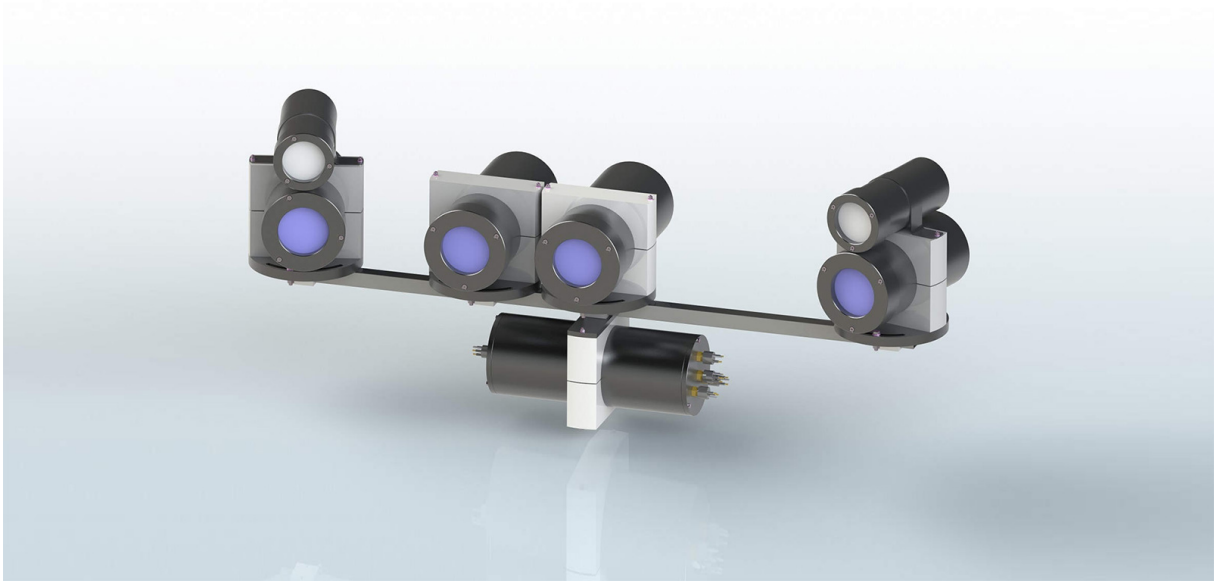

Figure 1.2: Preliminary design of the sensor platform of the *UWSensor* project[2]. The two outer housings each contain a 3D measurement camera and an additional light source. In the center housings, an additional color camera is installed, which is used for the visual odometry, as well as the fringe projector and the control unit of the sensors. One IMU is installed in each of the two lower housings, with the second one serving as a redundancy. The device has a width of about 1m.

from camera and IMU data before the actual position estimation can be performed. Here, the problem has been found that this initialization is not robust, depending on the data set used. Therefore, if the scale is estimated incorrectly in the initialization, the subsequent pose estimation does not work or the results are not satisfactory. This is mainly due to the fact that *VINS-Mono* estimates the rotation between the camera and the IMU, but not the translation. In the case of *UWSensor*, however, this must be taken into account, since the distance between the camera and the IMU is not negligible due to the sensor design.

The purpose of this work is now to improve the initialization of *VINS-Mono*, or of any *monocular visual inertial odometry* algorithm. This includes the calibration of the IMU and the estimation of the translation between camera and IMU in order to determine an accurate scale.

1.2 Contributions

The scientific contribution of this work involves the development of two approaches for the calibration of IMUs and the estimation of extrinsic parameters between a camera and an IMU. In this work, algorithms from [21] for estimating extrinsic parameters were extended for methods to find an optimal weighting of the datasets used. Experiments show that this strategy improves

the accuracy of the estimates. Using this developed software, the initialization of *VINS-Mono* is improved. This has the advantage, especially for the *UWSensor* project, that the trajectory is estimated with a higher accuracy, which also improves the result of the resulting point clouds. Also, by using the software developed in the course of this thesis, the trajectory of a drone flight estimated by *VINS-Mono* is improved.

1.3 Structure of the thesis

In Chapter 2 following the introduction of this thesis, the necessary tools and methods used in the course of this work are described. This includes general mathematical frameworks as well as explanations of the sensors used.

Subsequently, Chapter 3 presents methods to estimate the deterministic and stochastic error parameters necessary to calibrate an IMU, including the implementation of software to this purpose.

The second part of this thesis follows in Chapter 4, which presents methods, the underlying mathematics, and the final implementation of a software that determines the extrinsic calibration parameters between an IMU and a camera without external tools.

In the two Chapters 5 and 6 the results of the two software packages are then evaluated and explained.

In Chapter 8 the topic of this thesis is summarized and evaluated.

Chapter 2

Fundamentals and Background

In this chapter, the mathematical concepts, sensor techniques, software and algorithms used in the course of this work are explained.

2.1 Inertial Measurement Unit (IMU)

An IMU is a combination of several inertial sensors. An inertial sensor can be e.g. an acceleration sensor, a angular rate sensor or a magnetic field sensor. The IMUs, which are used in this work, represent a combination of the mentioned sensors. For each sensor type they have three sensors orthogonally arranged on top of each other, in order to be able to record translational and rotational movements or the magnetic field strength for all three spatial directions [23, p. 4] [3]. In the following, the operating principle of the three types of sensors will be briefly explained:

Accelerometer

The acceleration sensor measures the acceleration acting on it in all three spatial directions. Newton's second law $\boldsymbol{F} = m \cdot \boldsymbol{a}$ forms the basis of the acceleration measurement. The inertial force acting on a sample mass is measured. This is realized in a Microelectrome-

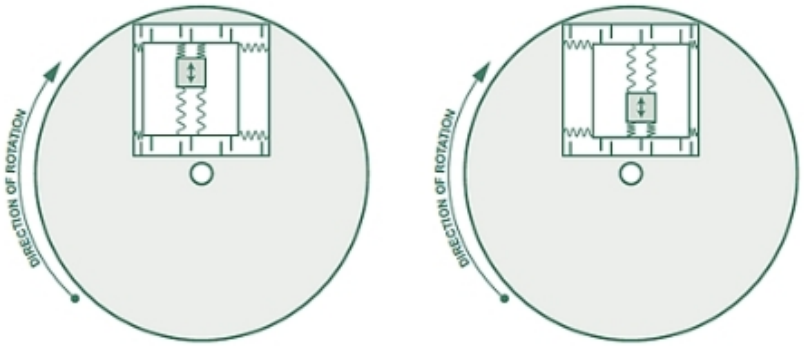

Figure 2.1: Operating principle of the angular rate sensor (source: [5])

chanical System (MEMS), a miniaturized acceleration sensor. The sample mass together with a spring represents a spring-mass system in which both components are made of silicon. To determine the deflection of the mass, the change in capacitance between the sample mass and a fixed reference electrode is measured when an acceleration occurs. With the knowledge of the deflection, the spring constant and the mass, the force acting on the sample mass and from this the acceleration is calculated. The sensor cannot distinguish between static and dynamic acceleration. On Earth, therefore, gravity must be subtracted from the measured values to obtain the dynamic acceleration [23, p. 4ff.] [1].

Gyroscope

The rotation rate sensor measures the rotation speed around all three spatial directions. This sensor is also implemented in a MEMS. The measuring principle is based on the Coriolis effect, whereby the Coriolis acceleration is measured. For each of the three directions of rotation, there is a spring-loaded frame which can move in the direction of rotation. A sample mass is attached within the frame, which oscillates orthogonally to the rotational motion. Fig. 2.1 shows this sensor setup and the measuring principle. If the sample mass moves towards the outside of the rotation, it experiences a Coriolis force opposite to the direction of rotation. If it moves toward the center of rotation, it experiences a Coriolis force in the direction of rotation. These forces move the frame in the direction, or against the direction, of rotation. This movement is recorded via capacitive measuring elements and the rotational speed is calculated from it [23, S. 9ff.] [5]. An alternative technology is a so-called Fibre-Optic Gyroscope (FOG). This uses the Sagnac effect to determine orientation changes. FOG technology is more expensive but more precise than MEMS technology. For a detailed description of this technology, see Article [31].

Magnetic field sensor

The magnetic field sensor measures the magnetic flux density or the magnetic field strength in all three spatial directions. The cardinal direction is determined by the relative orientation of the sensor to the magnetic field. The majority of magnetometers use the Hall effect to determine the magnetic flux density. This is done by passing an electric current through a conductive metal plate. The electrons flow directly through the plate. If the plate is crossed by a magnetic field, the electrons are deflected to one side of the plate, depending on the strength and direction of the magnetic field. Due to the resulting excess of electrons on one of the plate sides, a voltage is measured. This voltage is directly dependent on the strength of the magnetic field [6] [4].

2.2 Allan Variance

Allan Variance is a time domain analysis technique originally developed for characterizing noise and stability in clock systems, which is used for detecting and determining the underlying noise processes of gyroscopes and accelerometers. The Allan Variance analysis method is used in this thesis because it is easier and more reliable to determine the coefficients of the underlying noise processes compared to other techniques [34]. All formulas and descriptions in this section are taken from [12], [18], [35] and [15].

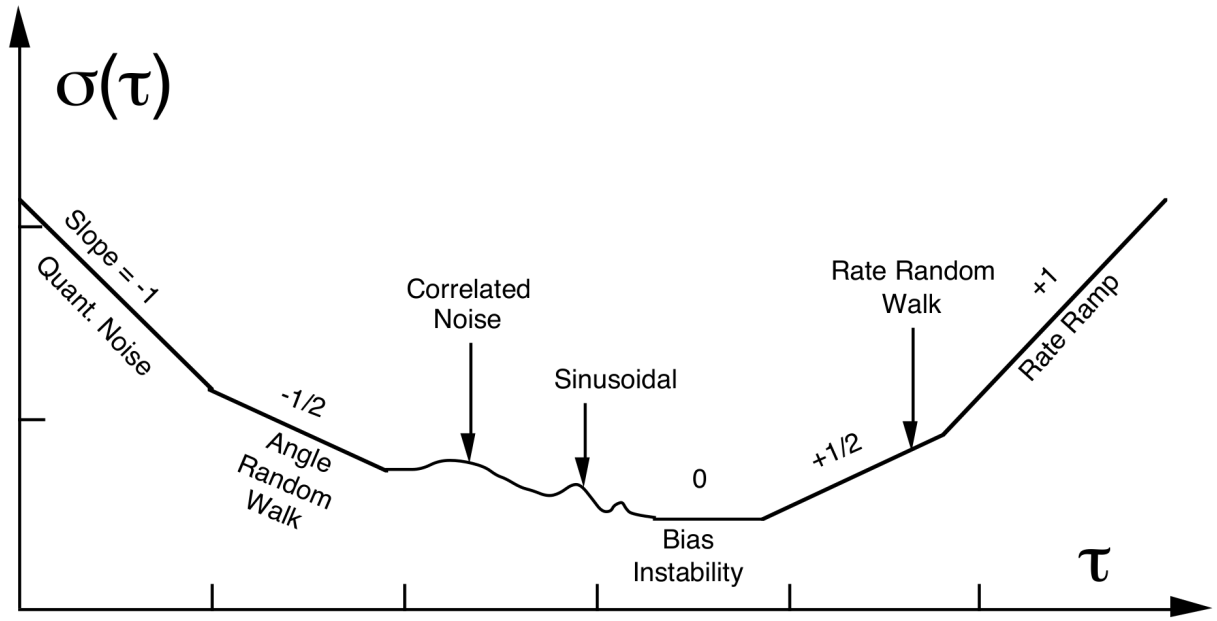

Figure 2.2: Exemplary representation of the Allan deviation $\sigma(\tau)$. The random processes recognizable by their specific slopes in the plot are marked by name. (Source: [12])

Consider a long sequence of N gyroscope or accelerometer measures $\Omega(t)$ with a sample time of t_0. Since the measurements are made at discrete times $t = kt_0$, $k = 1, 2, 3 ..., N$, the notation is simplified as $\Omega_k = \Omega(kt_0)$. These data points need to be divided into clusters of size n with $n < N/2$, where each cluster has a length of $\tau = t_0, 2t_0, ..., nt_0$. As a result, there are a number of $K = N/n$ clusters. The average of a cluster is now given by

$$\overline{\Omega}_k(\tau) = \frac{\theta_{k+n} - \theta_k}{\tau}, \tag{2.1}$$

and

$$\overline{\Omega}_{\text{next}}(\tau) = \frac{\theta_{k+2n} - \theta_{k+n}}{\tau}, \tag{2.2}$$

where the output angle or velocity is given by

$$\theta_k = t_0 \sum_{i=1}^{k} \Omega_i. \tag{2.3}$$

The Allan variance is now defined as

$$\sigma^2(\tau) = \frac{1}{2(K-1)} \sum_{k=1}^{K-1} \Big(\overline{\Omega}_{\text{next}}(\tau) - \overline{\Omega}_k(\tau)\Big)^2 \tag{2.4}$$

$$= \frac{1}{2\tau^2(K-1)} \sum_{k=1}^{K-1} \Big(\theta_{k+2n} - 2\theta_{k+n} + \theta_k\Big)^2. \tag{2.5}$$

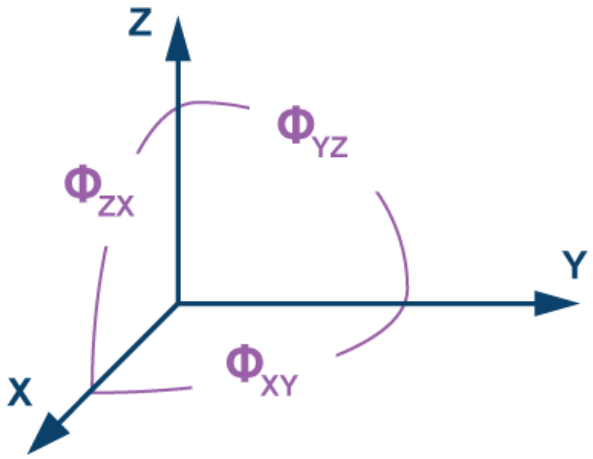

Figure 2.3: This diagram shows an example of the axis to axis misalignment of an accelerometer or gyroscope. Due to assembly inaccuracies, the angles between the individual sensor axes (Φ_{XY}, Φ_{YZ} and Φ_{ZX}) deviate from 90 degrees. (Source: [24])

By plotting the Allan Deviation $\sigma(\tau)$, hence the square root of the Allan Variance, in a log-log plot, the coefficients of the associated random processes are observed straightforwardly. A classic Allan Deviation plot is exemplified in Fig. 2.2. The following unique relationship exists between the Allan Variance and the Power spectral density (PSD) of the intrinsic random processes:

$$\sigma^2(\tau) = 4 \int_0^\infty S_\Omega(f) \cdot \frac{\sin^4(\pi f \tau)}{(\pi f \tau)^2} df. \tag{2.6}$$

This relationship is used to find an expression for the Allan variance $\sigma^2(\tau)$ as a function of cluster length τ for the PSD $S_\Omega(f)$ of a random process $Q(\tau)$. Eq. (2.6) can be interpreted as the proportionality between the Allan variance and the total power output of the random process when passed through a filter with a transfer function of the form $\sin^4(x)/(x)^2$. This particular transfer function is the result of the method used to create and operate on the clusters. A detailed discussion of PSDs is not presented in this thesis, the interested reader can refer to articles [25] and [18] for more information.

Section 3.3 describes how Eq. (2.6) is applied to determine the noise coefficients of the various random processes from Fig. 2.2, for given PSDs, listed in Section 2.3.2.

2.3 IMU Errors and Characteristics

The MEMS accelerometers and gyroscopes used for this work suffer from measurement errors in the uncalibrated case. These measurement errors, which consist of deterministic and stochastic errors, are described in the following sections.

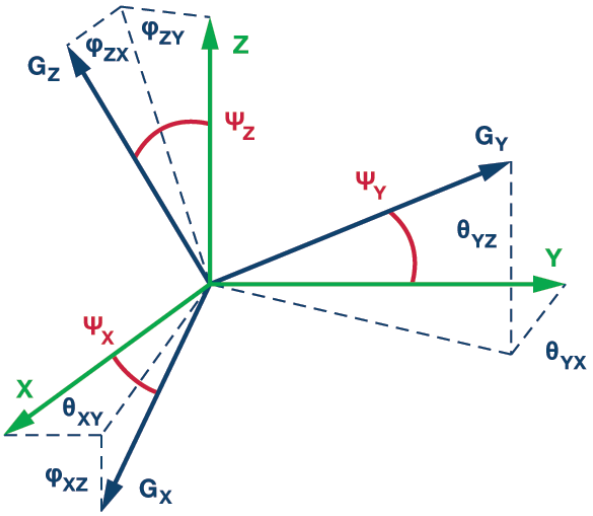

Figure 2.4: This diagram shows an example of an orthogonal three axis accelerometer or gyroscope, with alignment errors. The green orthogonal coordinate system defines a body frame, which is e.g. aligned to the IMU's chassis. The blue orthogonal coordinate system defines the accelerometers frame, respectively the gyroscopes frame. Due to assembly inaccuracies, the misalignment errors (Ψ_X, Ψ_Y and Ψ_Z) between the body frame and the sensor's axes (accelerometer or gyroscope) deviate from 0 degrees. (Source: [24])

2.3.1 Deterministic Errors

Due to the imperfect manufacturing process, IMUs suffer from deterministically quantifiable errors: constant biases, scale-factor errors and misalignment errors. These errors can be calibrated out and the interpretation of them is as follows.

In this thesis, the raw measurements of acceleration are referred to as ${}^{S_a}\hat{\boldsymbol{a}}$ and the raw measurements of angular velocity are referred to as ${}^{S_\omega}\hat{\boldsymbol{\omega}}$.

Misalignments

In the ideal case, the three sensor axes of the accelerometer, respectively the gyroscope, will each form an orthogonal coordinate system and coincides with the body frame $\{S\}$. The body frame can be aligned e.g. to the IMU's chassis.
Due to assembly inaccuracies, both the accelerometer frame $\{S_a\}$ and the gyroscope frame $\{S_\omega\}$ form two distinct (i.e., misaligned), non-orthogonal, frames, as demonstrated in Fig. 2.3.
In addition, the sensor coordinate systems of the accelerometer and the gyroscope differ from the body frame, as demonstrated in Fig. 2.4.
A detailed description of the background to these misalignments can be found in [24].

The purpose is to orthogonalize both sensor coordinate systems on one side and to ensure that both sensor coordinate systems refer to the same reference frame on the other side. Here, the ortogonalized coordinate frame of the accelerometer defines the reference coordinate frame, e.g. the body frame. As described in [22], for small angles, a measurement ${}^{A}\boldsymbol{s}$ in a non-orthogonal

frame A is converted into an orthogonal frame B as follows, where β_{ij} is the rotation angle of the i-th sensor axis around the j-th axis of the body frame:

$$^{B}\boldsymbol{s} = \boldsymbol{T}\,^{A}\boldsymbol{s}\,, \qquad \boldsymbol{T} = \begin{bmatrix} 1 & -\beta_{yz} & \beta_{zy} \\ \beta_{xz} & 1 & -\beta_{zx} \\ -\beta_{xy} & \beta_{yx} & 1 \end{bmatrix}. \tag{2.7}$$

Continuing, the body frame S is defined as follows:

- The x-axis of S_a coincides with that of S, it follows: $\beta_{xz} = \beta_{xy} = 0$
- The y-axis of S_a lies in the plane which is spanned by the x- and y-axis of S, it follows: $\beta_{yx} = 0$.

For the accelerometer Eq. (2.7) is simplified to:

$$^{S}\hat{\boldsymbol{a}} = \boldsymbol{T}^{a}\,^{S_a}\hat{\boldsymbol{a}}\,, \qquad \boldsymbol{T}^{a} = \begin{bmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.8}$$

where β has been substituted with α for the case of the accelerometer.

For the Gyroscope Eq. (2.7) becomes:

$$^{S}\hat{\boldsymbol{\omega}} = \boldsymbol{T}^{\omega}\,^{S_\omega}\hat{\boldsymbol{\omega}}\,, \qquad \boldsymbol{T}^{\omega} = \begin{bmatrix} 1 & -\gamma_{yz} & \gamma_{zy} \\ \gamma_{xz} & 1 & -\gamma_{zx} \\ -\gamma_{xy} & \gamma_{yx} & 1 \end{bmatrix} \tag{2.9}$$

where β has been substituted with γ for the case of the gyroscope.

In summary, Eq. (2.8) and Eq. (2.9) are used to orthogonalize the non-orthogonal sensor coordinate systems of the accelerometer and the gyroscope and rotate them to the same body frame.

Scale Factor Errors

The scaling factors for converting the digital measured value of the sensors into the physical value acceleration, respectively angular velocity, vary for different instances of the same sensors. To minimize the measurement errors the scaling factors for each IMU must be determined. For this purpose, the following scaling matrices for the accelerometer and the gyroscope, respectively, are introduced:

$$\boldsymbol{K}^{a} = \begin{bmatrix} s_x^a & 0 & 0 \\ 0 & s_y^a & 0 \\ 0 & 0 & s_z^a \end{bmatrix}, \qquad \boldsymbol{K}^{\omega} = \begin{bmatrix} s_x^\omega & 0 & 0 \\ 0 & s_y^\omega & 0 \\ 0 & 0 & s_z^\omega \end{bmatrix}. \tag{2.10}$$

Here s_i^a, or s_i^ω, is the scaling factor for the i-th sensor axis of the accelerometer, or gyroscope.

In summary, the individual sensor axes of the accelerometer and the gyroscope can now be scaled with the help of the matrices (2.10), so that the digital measured value of the sensors are converted into the correct physical value.

Constant Bias Error

Both the accelerometers and the gyroscopes are affected by biases which are the offsets in the measurement provided by the inertial sensor. The bias of each sensor is generally composed of two components: one deterministic part called bias offset and a random part [15, p. 21]. The random part of the bias is discussed later in Section 2.3.2. The biases for each IMU must be determined to minimize the measurement errors. For this purpose, we introduce the following bias vectors for the accelerometer and the gyroscope, respectively:

$$\boldsymbol{b}^a = \begin{bmatrix} b_x^a \\ b_y^a \\ b_z^a \end{bmatrix}, \qquad \boldsymbol{b}^\omega = \begin{bmatrix} b_x^\omega \\ b_y^\omega \\ b_z^\omega \end{bmatrix} \tag{2.11}$$

Here b_i^a, or b_i^ω, is the bias for the i-th sensor axis of the accelerometer, or gyroscope.

In summary, the individual sensor axes of the accelerometer and the gyroscope can now be corrected of their bias using the vectors (2.11).

Deterministic Sensor Error Model

The following sensor error models for the accelerometer and gyroscope are determined from the previous sections and allow the corrected measurements for the acceleration (${}^S\boldsymbol{a}$) and angular velocity (${}^S\boldsymbol{\omega}$) in the reference frame S to be determined.

$${}^S\boldsymbol{a} = \boldsymbol{T}^a\boldsymbol{K}^a({}^{S_a}\hat{\boldsymbol{a}} + \boldsymbol{b}^a) \tag{2.12}$$

$${}^S\boldsymbol{\omega} = \boldsymbol{T}^\omega\boldsymbol{K}^\omega({}^{S_\omega}\hat{\boldsymbol{\omega}} + \boldsymbol{b}^\omega) \tag{2.13}$$

2.3.2 Stochastic Errors

After the deterministic errors are removed from the accelerometer and gyroscope measurements, the stochastic errors remain. These errors cannot be removed and must be modeled as a stochastic processes. The prevailing errors are explained below, with all formulas and descriptions in this section taken from [12] if not otherwise indicated.

Bias Instability

The bias stability provides information on how stable the bias of a gyroscope or accelerometer is over a period of time. The rate Power spectral density (PSD) associated with this noise, also known as $1/f$ or pink noise, is

$$S_\Omega(f) = \begin{cases} \frac{B^2}{2\pi}\frac{1}{f} & f \leq f_0 \\ 0 & f > f_0 \end{cases}, \tag{2.14}$$

where B is the bias instability coefficient and f_0 the 3dB cutoff frequency.

Angle Random Walk (ANRW) / Velocity Random Walk (VRW)
Due to thermo-mechanical noise disturbed gyroscopes and accelerometers, the discrete measurements of these sensors are disturbed by a white noise sequence. This noise is composed of a sequence of zero-mean uncorrelated random variables, where each random variable is identically distributed with a finite variance. If this white noise signal is integrated, which is the case when calculating the angle or velocity from the sensor readings, this leads to zero-mean ANRW or VRW errors in the integrated signals. Thereby the standard deviation of the noise increases with the squared root of time. [35]
The associated rate noise PSD is represented by

$$S_\Omega(f) = Q^2, \tag{2.15}$$

where Q is the ANRW or VRW coefficient.

Rate Random Walk (RRW) / Acceleration Random Walk (ACRW)
These two random processes have an unknown origin and the associated rate PSD from this noise, also known as $1/f^2$ or Brownian noise, is

$$S_\Omega(f) = \left(\frac{K}{2\pi}\right)^2 \frac{1}{f^2}, \tag{2.16}$$

where K is the RRW or ACRW coefficient.

Quantization Noise
Due to the quantization in the Analog-to-digital converter (ADC), there is a rounding error between the analog input voltage and the digitized output value. The quantization noise is a model of this quantization error with a rate PSD of

$$S_\Omega(f) = \begin{cases} \frac{4Q^2}{t_0}\sin^2(\pi f t_0) \\ \approx (2\pi f)^2 t_0 Q^2 \quad f < \frac{1}{2t_0} \end{cases}, \tag{2.17}$$

where Q is the quantization noise coefficient.

2.4 Camera Calibration

In the course of this work, the images captured by a camera are used to obtain information about the position of the camera. For this purpose, the positions of the features to be recognized in the image are determined by the positions of the respective pixels. If the radial and tangential distortion parameters of the camera are not known or only with large error, this has a negative effect on the position estimated by the camera images. Therefore, in this work we are dependent on calibration parameters that are as exact as possible, which are assumed to be given in this case, since the calibration of cameras would exceed the scope of this work. For a detailed description of methods for calibrating cameras, reference is made to relevant literature such as [17]. If the reader is interested in performing the camera calibration on his own, the ROS package *camera_calibration* [1] is recommended.

[1] `http://wiki.ros.org/camera_calibration`

2.5 Visual Inertial Odometry (VIO)

As already explained, this thesis deals with VIO methods. So-called Visual Odometry methods focus on the determination of the position and orientation of a moving system equipped with a camera. If an IMU is also used to support the position estimation, it is called a VIO method. This work is limited to the VIO software package *VINS-Mono*. Details about the functionality are described in [27]. A detailed mathematical explanation of VIO methods would exceed the scope of this thesis, so please refer to literature like [32].

2.6 Roboter Operating System (ROS)

2.6.1 Basic Concepts

ROS is a flexible general-purpose framework for a wide variety of autonomous systems. Among other things, it provides the following services:

- Hardware abstraction
- Device driver
- Libraries
- Visualization
- Communication
- Package management

The objective behind the development work started in 2007 under the name *Switchyard* is the simplification of complex and stable tasks to different robot systems. The development of a general-purpose robot software by a single person, or an institution would be effortful. The open-source software ROS has since been developed by a community of scientists, laboratories and institutions. Anyone can contribute their expertise in a particular area. The five basic ideas on which the design philosophy is based are:

Peer-to-Peer
This enables the use of multiple computers in the same robot system. In addition, data can be transferred to external computers for computationally intensive processing.

Tool-based
In favor of low complexity, ROS is implemented in a micro kernel design.

Multi-language
ROS features connections for the programming languages C++, Python and Lisp. In addition, there are modules for JAVA, Haskell and Lua, among others.

Slim
Algorithms are developed as a library independent of concrete hardware. This facilitates porting to other hardware platforms and the re-usability of the algorithms.

Open Source
ROS is under the BSD license and can therefore be used for both commercial and non-commercial projects without restrictions.

[2][11][7][8][9]

2.6.2 Architecture

The main task of a robot operating system is the simultaneous processing of different jobs. The information can be exchanged synchronously or asynchronously depending on the requirements. The individual components of the ROS system are presented below:

Node
A node is a process that performs calculations. It can refer to a sensor, a motor, a processing algorithm or a monitoring algorithm. Several nodes can run in parallel and declare themselves to the master.

Master
The master is the declaration and storage service of the individual nodes and enables them to know each other. It has the following tasks:

- Storage of information about which nodes publish or subscribe to which topics.
- Management of the services provided.
- Provision of services for finding communication partners at run time.

Topics
ROS enables asynchronous information exchange through Topics and synchronous information exchange through Services. A Topic is an information movement based on a Subscribe/Publish system. Nodes send and receive Topic messages.

Messages
A message consists of typecast entries, which can be different basic data types, fields or nestings.

Bags
Bags can be used to store all messages from the ROS system for later use or evaluation.

The ability to divide different computational processes among individual nodes simplifies software development for complex autonomous systems and their modeling [2][10].

2.7 Optimization Algorithms

Since in the course of this work some non-linear least squares problems have to be solved numerically, two possible *C++* software libraries are presented in the following, which carry out this task for this thesis. To ensure that the optimization algorithms work correctly, two libraries are used and compared to each other.

2.7.1 Eigen's Levenberg–Marquardt (LM) Algorithm

The Levenberg-Marquardt algorithm was developed to solve non-linear least squares problems and it combines two numerical minimization algorithms: the gradient descent method and the Gauss-Newton method. A detailed description of the algorithm can be found in paper [19]. Instructions for using the *C++* implementation of Eigen's [2] LM algorithm can be found online [3].

The implementation of the methods presented in this thesis mainly uses this library. However, it has been found that this optimization algorithm sometimes does not converge in the estimation of the translation between the IMU and the camera, which will be explained later. For this reason, the optimization algorithm presented in the next section is also used.

2.7.2 Ceres Solver

Ceres Solver is an open source *C++* library for modeling and solving large, complicated optimization problems. It can be used to solve non-linear least squares problems with bounds constraints and general unconstrained optimization problems. It is a mature, feature rich, and performant library that has been used in production at Google since 2010 [13].

This optimization algorithm is used in the implementation of Chapter 4 to get a comparison to the results of Eigen's Levenberg–Marquardt (LM) algorithm.

[2] `https://eigen.tuxfamily.org/dox/unsupported/classEigen_1_1LevenbergMarquardt.html`
[3] `https://medium.com/@sarvagya.vaish/levenberg-marquardt-optimization-part-2-5a71f7db27a0`

Chapter 3

IMU Calibration

In this chapter the procedure for calibration of IMUs is explained, regarding the mathematical background followed by the implementation of the calibration algorithms. This allows to correct the measured values of the IMU before these are further processed. The underlying sources of error were explained in Section 2.3. As will be shown later, the error of acceleration measurement is reduced by a factor of 30 (cf. Section 5), which reveals the importance of calibration.

3.1 Deterministic Error Parameters Estimation

To minimize the deterministic errors described in Section 2.3.1, the techniques developed by [29] are used. These allow to eliminate the above-mentioned sources of error relating to the IMU without the need for external equipment. In the following, the underlying mathematics are explained.

3.1.1 Estimating Error Parameters of Accelerometer

To calibrate the accelerometer, the matrices $\boldsymbol{T}^a$ and $\boldsymbol{K}^a$ and the vector $\boldsymbol{b}^a$ must be determined, hence the following vector with the unknown parameters:

$$\boldsymbol{\theta}^a = \begin{bmatrix} \alpha_{yz} & \alpha_{zy} & \alpha_{zx} & s_x^a & s_y^a & s_z^a & b_x^a & b_y^a & b_z^a \end{bmatrix}. \tag{3.1}$$

The purpose of the calibration is to estimate the parameters in $\boldsymbol{\theta}^a$ such that the sensor measures $|{}^S\boldsymbol{a}| = |\boldsymbol{g}|$ at a standstill regardless of the orientation to the gravitational vector $\boldsymbol{g}$. To estimate these unknown parameters we use the following cost function:

$$\boldsymbol{L}(\boldsymbol{\theta}^a) = \sum_{k=1}^{M} \Big(|\boldsymbol{g}|^2 - |{}^S\boldsymbol{a}|^2 \Big) = \sum_{k=1}^{M} \Big(|\boldsymbol{g}|^2 - |\boldsymbol{T}^a \boldsymbol{K}^a ({}^{S_a}\hat{\boldsymbol{a}} + \boldsymbol{b}^a)|^2 \Big) \tag{3.2}$$

Where M is the number of acceleration vectors ${}^{S_a}\hat{\boldsymbol{a}}$ measured at different orientations. In order to estimate the parameters precisely, the cost function (3.2) is minimized using the Levenberg–Marquardt (LM) algorithm, explained in Section 2.7.1. At least $M = 9$ data sets must

be provided for this. However, it has been observed that the results are better when about $M = 30$ data sets are used for minimization. A more detailed description of the procedure and implementation follows in Section 3.2.

3.1.2 Estimating Error Parameters of Gyroscope

To calibrate the gyroscope, the matrices $\boldsymbol{T}^{\omega}$ and $\boldsymbol{K}^{\omega}$ and the vector $\boldsymbol{b}^{\omega}$ must be determined. The bias $\boldsymbol{b}^{\omega}$ can be determined by averaging the measurement of the rotational speed ${}^{S_\omega}\hat{\boldsymbol{\omega}}$ at standstill over a suitable time period. Typically, a duration of a few seconds is sufficient. The remaining unknown parameters are combined into the following vector:

$$\boldsymbol{\theta}^{\omega} = \begin{bmatrix} \gamma_{yz} & \gamma_{zy} & \gamma_{xz} & \gamma_{zx} & \gamma_{xy} & \gamma_{yx} & s_x^{\omega} & s_y^{\omega} & s_z^{\omega} \end{bmatrix}. \tag{3.3}$$

The purpose of the calibration is to estimate the parameters in $\boldsymbol{\theta}^{\omega}$ such that the temporal integration of the measured angular velocities corresponds to the same change in orientation as determined by the accelerometer. The following function is defined, which uses a sequence of bias-corrected gyroscope measurements ${}^{S_\omega}\hat{\boldsymbol{\omega}}_k - \boldsymbol{b}^{\omega} = {}^{S_\omega}\bar{\boldsymbol{\omega}}_k$ and an initial orientation to determine the final orientation:

$$\boldsymbol{u}_m^{\omega} = \Psi\left[{}^{S_\omega}\bar{\boldsymbol{\omega}}_k, \boldsymbol{u}_{m-1}^{a}\right] \tag{3.4}$$

Here $\boldsymbol{u}_{m-1}^{a}$ is the initial orientation determined by the accelerometer at the $(m-1)$-th standstill. The operator Ψ represents here an arbitrary integration algorithm, which calculates the final orientation $\boldsymbol{u}_m^{\omega}$ at the m-th standstill with the help of the initial orientation $\boldsymbol{u}_{m-1}^{a}$ and the gyroscope measurements between the $(m-1)$-th and m-th standstill. The orientation is defined with $\boldsymbol{u}$ as the direction of the gravitational acceleration vector. A possible approach for integration is explained in the following sections.

To estimate the unknown parameters $\boldsymbol{\theta}^{\omega}$ in the case of the gyroscope the following cost function is used:

$$\boldsymbol{L}(\boldsymbol{\theta}^{\omega}) = \sum_{m=2}^{M} |\boldsymbol{u}_m^{a} - \boldsymbol{u}_m^{\omega}|^2 \tag{3.5}$$

where M represents the number of standstills and $\boldsymbol{u}_m^{a}$ the final orientation measured by the accelerometer at the m-th standstill. In order to estimate the parameters precisely, the cost function (3.5) is minimized using the LM algorithm, explained in Section 2.7.1. A more detailed description of the procedure and implementation follows in Section 3.2.

3.1.3 Integrating Angular Velocity to get a Quaternion

In this work quaternions are used for the representation of rotations, since they do not lead to ambiguous results and no singularity problems arise. This section will give an overview of how to properly integrate the angular velocities measured by the gyroscope to obtain a quaternion. The following mathematical backgrounds were acquired online [1] [2] and from the approach developed by [33].

[1] `https://www.ashwinnarayan.com/post/how-to-integrate-quaternions/`
[2] `https://mathepedia.de/Quaternionen.html`

Multiplication of two quaternions is defined as follows:

$$\boldsymbol{q}_1 \otimes \boldsymbol{q}_2 = \begin{bmatrix} w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \\ w_1x_2 + w_2x_1 + y_1z_2 - y_2z_1 \\ w_1y_2 + w_2y_1 - x_1z_2 + x_2z_1 \\ w_1z_2 + w_2z_1 + x_1y_2 - x_2y_1 \end{bmatrix}. \tag{3.6}$$

This can be written as a matrix product:

$$\boldsymbol{q}_1 \otimes \boldsymbol{q}_2 = \begin{bmatrix} w_2 & -x_2 & -y_2 & -z_2 \\ x_2 & w_2 & z_2 & -y_2 \\ y_2 & -z_2 & w_2 & x_2 \\ z_2 & y_2 & -x_2 & w_2 \end{bmatrix} \begin{bmatrix} w_1 \\ x_1 \\ y_1 \\ z_1 \end{bmatrix}, \tag{3.7}$$

where a quaternion is represented as a tuple of 4 numbers:

$$\boldsymbol{q} = \begin{bmatrix} w & x & y & z \end{bmatrix}^\mathsf{T} = \begin{bmatrix} w & \boldsymbol{v} \end{bmatrix}^\mathsf{T}. \tag{3.8}$$

Here $S(\boldsymbol{q}) = w$ is called scalar part and $V(\boldsymbol{q}) = \boldsymbol{v}$ is called vector part. A conjugate quaternion is defined as follows:

$$\boldsymbol{q}^* = \begin{bmatrix} w & -x & -y & -z \end{bmatrix}^\mathsf{T} \tag{3.9}$$

and has the property

$$(\boldsymbol{q}_1 \otimes \boldsymbol{q}_2)^* = \boldsymbol{q}_2^* \otimes \boldsymbol{q}_1^* \tag{3.10}$$

Any 3D vector $\boldsymbol{r}$ can be written as a pure quaternion:

$$\boldsymbol{r}_q = \begin{bmatrix} 0 & \boldsymbol{r} \end{bmatrix}^\mathsf{T} = \begin{bmatrix} 0 & r_x & r_y & r_z \end{bmatrix}^\mathsf{T} \tag{3.11}$$

Unit quaternions, i.e. $||\boldsymbol{q}|| = 1$, can be used to rotate a vector ${}^A\boldsymbol{r}$, expressed with respect to the frame A, to a vector ${}^B\boldsymbol{r}$, expressed with respect to the frame B as follows:

$${}^B\boldsymbol{r}_q = {}^B_A\boldsymbol{q} \otimes {}^A\boldsymbol{r}_q \otimes {}^B_A\boldsymbol{q}^*, \tag{3.12}$$

where ${}^B_A\boldsymbol{q}$ represents any arbitrary orientation in the 3D space of a frame A with respect to a frame B.

In addition, zero rotation is defined as follows:

$$\boldsymbol{q}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^\mathsf{T}. \tag{3.13}$$

In the following the relationship between quaternions and angular velocity will be described. The physical law relating the tangential velocity of a circularly moving particle with position vector $\boldsymbol{r}(t)$ and the associated vector $\boldsymbol{\omega}(t)$ of angular velocity is defined as follows:

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{\omega} \times \boldsymbol{r} \tag{3.14}$$

Since a representation in quaternion form is required, Eq. (3.11) is used to obtain the two pure quaternions $\boldsymbol{r}_q$ and $\boldsymbol{\omega}_q$. In the multiplication of two pure quaternions $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ the vector part $V(\boldsymbol{q})$ of the resulting quaternion $\boldsymbol{q}$ corresponds to the cross product of the vector parts of $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$. Furthermore, the scalar part $S(\boldsymbol{q})$ corresponds to the scalar product of the vector parts of $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ except for the sign. Since the angular velocity $\boldsymbol{\omega}$ is perpendicular to the position vector $\boldsymbol{r}$, it follows with:

$$S(\boldsymbol{q}) = -\boldsymbol{q}_a \cdot \boldsymbol{q}_b = -\boldsymbol{\omega}_q \cdot \boldsymbol{r}_q = S\Big(\frac{\mathrm{d}\boldsymbol{r}_q}{\mathrm{d}t}\Big) = 0 \tag{3.15}$$

$$V(\boldsymbol{q}) = \boldsymbol{q}_a \times \boldsymbol{q}_b = \boldsymbol{\omega}_q \times \boldsymbol{r}_q = V\Big(\frac{\mathrm{d}\boldsymbol{r}_q}{\mathrm{d}t}\Big) \tag{3.16}$$

that Eq. (3.14) can also be written in quaternion form:

$$\frac{\mathrm{d}\boldsymbol{r}_q}{\mathrm{d}t} = \boldsymbol{\omega}_q \otimes \boldsymbol{r}_q. \tag{3.17}$$

In addition, the quaternion commutator for the two pure quaternions $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ is defined as follows:

$$[\boldsymbol{q}_a, \boldsymbol{q}_b] = 2(\boldsymbol{q}_a \times \boldsymbol{q}_b) = 2(\boldsymbol{q}_a \otimes \boldsymbol{q}_b). \tag{3.18}$$

Now imagine that the position vector $\boldsymbol{r}_q(t)$ of the moving particle can be described as follows:

$$\boldsymbol{r}_q = \boldsymbol{q} \otimes \boldsymbol{r}_{q,0} \otimes \boldsymbol{q}^* \tag{3.19}$$

$$\Leftrightarrow \boldsymbol{r}_{q,0} = \boldsymbol{q}^* \otimes \boldsymbol{r}_q \otimes \boldsymbol{q} \tag{3.20}$$

$$\frac{\mathrm{d}\boldsymbol{r}_q}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}[\boldsymbol{q} \otimes \boldsymbol{r}_{q,0} \otimes \boldsymbol{q}^*] \tag{3.21}$$

where the constant vector $\boldsymbol{r}_{q,0}$ is rotated to $\boldsymbol{r}_q$ using the quaternion $\boldsymbol{q}$. To solve Eq. (3.21), the product rule can be applied, since it is also valid for unit quaternions [3]:

$$\frac{\mathrm{d}\boldsymbol{r}_q}{\mathrm{d}t} = \dot{\boldsymbol{q}} \otimes \boldsymbol{r}_{q,0} \otimes \boldsymbol{q}^* + \boldsymbol{q} \otimes \boldsymbol{r}_{q,0} \otimes \dot{\boldsymbol{q}}^* \tag{3.22}$$

Since $\boldsymbol{q}$ is a unit quaternion the following relationship can be established:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{q} \otimes \boldsymbol{q}^*) = \frac{\mathrm{d}}{\mathrm{d}t}1 \tag{3.23}$$

$$\dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* + \boldsymbol{q} \otimes \dot{\boldsymbol{q}}^* = 0 \tag{3.24}$$

$$\Leftrightarrow \boldsymbol{q} \otimes \dot{\boldsymbol{q}}^* = -\dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \tag{3.25}$$

$$\Leftrightarrow \boldsymbol{q}^* \otimes \boldsymbol{q} \otimes \dot{\boldsymbol{q}}^* = -\boldsymbol{q}^* \otimes \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \tag{3.26}$$

$$\Leftrightarrow \boldsymbol{q}_0 \otimes \dot{\boldsymbol{q}}^* = -\boldsymbol{q}^* \otimes \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \tag{3.27}$$

$$\Leftrightarrow \dot{\boldsymbol{q}}^* = -\boldsymbol{q}^* \otimes \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \tag{3.28}$$

[3] `https://fgiesen.wordpress.com/2012/08/24/quaternion-differentiation/`

Now the relationships (3.28) and (3.20) are used to substitute $\dot{\boldsymbol{q}}^*$ and $\boldsymbol{r}_{q,0}$ in Eq. (3.22).

$$\frac{\mathrm{d}\boldsymbol{r}_q}{\mathrm{d}t} = \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \otimes \boldsymbol{r}_q \otimes \boldsymbol{q} \otimes \boldsymbol{q}^* - \boldsymbol{q} \otimes \boldsymbol{q}^* \otimes \boldsymbol{r}_q \otimes \boldsymbol{q} \otimes \boldsymbol{q}^* \otimes \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \tag{3.29}$$

$$\Leftrightarrow \frac{\mathrm{d}\boldsymbol{r}_q}{\mathrm{d}t} = \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \otimes \boldsymbol{r}_q \otimes \boldsymbol{q}_0 - \boldsymbol{q}_0 \otimes \boldsymbol{r}_q \otimes \boldsymbol{q}_0 \otimes \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \tag{3.30}$$

$$\Leftrightarrow \frac{\mathrm{d}\boldsymbol{r}_q}{\mathrm{d}t} = \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \otimes \boldsymbol{r}_q - \boldsymbol{r}_q \otimes \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \tag{3.31}$$

From Eq. (3.24), the following relationship can be derived with the help of property (3.10):

$$\dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* + \boldsymbol{q} \otimes \dot{\boldsymbol{q}}^* = 0 \tag{3.32}$$

$$\Leftrightarrow \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* = -\boldsymbol{q} \otimes \dot{\boldsymbol{q}}^* \tag{3.33}$$

$$\Leftrightarrow \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* = -(\dot{\boldsymbol{q}} \otimes \boldsymbol{q}^*)^* \tag{3.34}$$

The expression (3.34) is in the form of $\boldsymbol{q} = -\boldsymbol{q}^*$, which is equivalent to $w = -w$. Consequently, it can be concluded that the scalar part is zero, therefore $S(\dot{\boldsymbol{q}} \otimes \boldsymbol{q}^*) = 0$, and that the product $\dot{\boldsymbol{q}} \otimes \boldsymbol{q}^*$ is a pure quaternion.

With this knowledge, the quaternion commutator from Eq. (3.18) and Eq. (3.17), Eq. (3.31) is simplified as follows:

$$\dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \otimes \boldsymbol{r}_q - \boldsymbol{r}_q \otimes \dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* = \boldsymbol{\omega}_q \otimes \boldsymbol{r}_q \tag{3.35}$$

$$\Leftrightarrow [\dot{\boldsymbol{q}} \otimes \boldsymbol{q}^*, \boldsymbol{r}_q] = \boldsymbol{\omega}_q \otimes \boldsymbol{r}_q \tag{3.36}$$

$$\Leftrightarrow 2\dot{\boldsymbol{q}} \otimes \boldsymbol{q}^* \otimes \boldsymbol{r}_q = \boldsymbol{\omega}_q \otimes \boldsymbol{r}_q \tag{3.37}$$

$$\Leftrightarrow \dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\omega}_q \otimes \boldsymbol{q} \tag{3.38}$$

Here the angular velocity $\boldsymbol{\omega}_q$ refers to the global fixed frame. However, the gyroscope measures the uncorrected angular velocities in its sensor coordinate system S_ω, so the relationship $\boldsymbol{\omega}_q = \boldsymbol{q} \otimes {}^{S_\omega}\hat{\boldsymbol{\omega}}_q \otimes \boldsymbol{q}^*$ can be used to substitute $\boldsymbol{\omega}_q$ in Eq. (3.38):

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes {}^{S_\omega}\hat{\boldsymbol{\omega}}_q \otimes \boldsymbol{q}^* \otimes \boldsymbol{q} \tag{3.39}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes {}^{S_\omega}\hat{\boldsymbol{\omega}}_q \tag{3.40}$$

With the help of Eq. (3.7) the following final equation that allows to calculate the change of the quaternion is obtained:

$$\begin{bmatrix} \dot{w} \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -\hat{\omega}_x & -\hat{\omega}_y & -\hat{\omega}_z \\ \hat{\omega}_x & 0 & \hat{\omega}_z & -\hat{\omega}_y \\ \hat{\omega}_y & -\hat{\omega}_z & 0 & \hat{\omega}_x \\ \hat{\omega}_z & \hat{\omega}_y & -\hat{\omega}_x & 0 \end{bmatrix} \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} \tag{3.41}$$

3.1.4 Runge-Kutta Integration

The gyroscope provides us with discrete measurements of angular velocity at timestamps t_k with a sampling interval Δt. As the authors of [29], the Runge-Kutta 4^{th} order normalized (RK4n) method is used, since this provides sufficient accuracy. A detailed description of the Runge-Kutta procedures can be obtained from [14]. The RK4n algorithm is given as follows:

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \frac{h}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{3.42}$$

$$\boldsymbol{k}_1 = \boldsymbol{f}(\boldsymbol{q}_k, t_k) \tag{3.43}$$

$$\boldsymbol{k}_2 = \boldsymbol{f}\left(\boldsymbol{q}_k + \frac{h}{2}\boldsymbol{k}_1, t_k + \frac{h}{2}\right) \tag{3.44}$$

$$\boldsymbol{k}_3 = \boldsymbol{f}\left(\boldsymbol{q}_k + \frac{h}{2}\boldsymbol{k}_2, t_k + \frac{h}{2}\right) \tag{3.45}$$

$$\boldsymbol{k}_4 = \boldsymbol{f}\left(\boldsymbol{q}_k + h\boldsymbol{k}_3, t_k + h\right) \tag{3.46}$$

Where $\boldsymbol{q}_k$ is the orientation quaternion at timestamp t_k, h the step size and from the quaternion kinematics Eq. (3.40), let

$$\boldsymbol{f}(\boldsymbol{q}, t_k) = \frac{1}{2}\boldsymbol{q} \otimes {}^{S_\omega}\hat{\boldsymbol{\omega}}_{q,k}. \tag{3.47}$$

In our case, $\boldsymbol{f}(\boldsymbol{q}, t_k)$ can only be evaluated at the discrete measuring times t_k, since the angular velocity is only determined or measured there. Accordingly, $h = 2\Delta t$ is set. Hence, the following is obtained from Eqs. (3.42) - (3.46):

$$\boldsymbol{q}_{k+2} = \boldsymbol{q}_k + \frac{\Delta t}{3}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{3.48}$$

$$\boldsymbol{k}_1 = \boldsymbol{f}(\boldsymbol{q}_k, t_k) \tag{3.49}$$

$$\boldsymbol{k}_2 = \boldsymbol{f}\left(\boldsymbol{q}_k + \Delta t\boldsymbol{k}_1, t_k + \Delta t\right) = \boldsymbol{f}\left(\boldsymbol{q}_k + \Delta t\boldsymbol{k}_1, t_{k+1}\right) \tag{3.50}$$

$$\boldsymbol{k}_3 = \boldsymbol{f}\left(\boldsymbol{q}_k + \Delta t\boldsymbol{k}_2, t_k + \Delta t\right) = \boldsymbol{f}\left(\boldsymbol{q}_k + \Delta t\boldsymbol{k}_2, t_{k+1}\right) \tag{3.51}$$

$$\boldsymbol{k}_4 = \boldsymbol{f}\left(\boldsymbol{q}_k + 2\Delta t\boldsymbol{k}_3, t_k + 2\Delta t\right) = \boldsymbol{f}\left(\boldsymbol{q}_k + 2\Delta t\boldsymbol{k}_3, t_{k+2}\right) \tag{3.52}$$

Finally, each recalculated $\boldsymbol{q}_{k+2}$ must be normalized to ensure that the orientation quaternion remains a unit quaternion:

$$\boldsymbol{q}_{k+2} \rightarrow \frac{\boldsymbol{q}_{k+2}}{||\boldsymbol{q}_{k+2}||} \tag{3.53}$$

In this work, the final orientation from Eq. (3.4) is determined as follows. As already mentioned above, the n bias-corrected angular velocities ${}^{S_\omega}\bar{\boldsymbol{\omega}}_k$ measured between two successive standstills need to be numerically integrated to obtain the orientation determined by the gyroscope. Starting with an initial orientation $\boldsymbol{q}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^\mathsf{T}$, which represents the null rotation, the

Eqs. (3.48) - (3.53) are now applied iteratively to obtain the quaternion $\boldsymbol{q}_n$, which describes the rotation since the previous standstill. If n is odd, the last measured value $({}^{S_\omega}\hat{\boldsymbol{\omega}}_n - \boldsymbol{b}^\omega)$ is integrated as follows:

$$\boldsymbol{q}_n = \boldsymbol{q}_{n-1} + \Delta t \cdot \boldsymbol{f}(\boldsymbol{q}_{n-1}, t_n) = \boldsymbol{q}_{n-1} + \frac{\Delta t}{2} \cdot \boldsymbol{q}_{n-1} \otimes \left({}^{S_\omega}\hat{\boldsymbol{\omega}}_n - \boldsymbol{b}^\omega\right)_q \tag{3.54}$$

In the next step, $\boldsymbol{q}_n$ has to be normalized according Eq. (3.53). Finally, the final orientation from Eq. (3.4) is obtained as follows:

$$\boldsymbol{u}^\omega_{q,m} = \boldsymbol{q}_n \otimes \boldsymbol{u}^a_{q,m-1} \otimes \boldsymbol{q}^*_n \tag{3.55}$$

3.2 Implementation Deterministic Error Parameter Estimation

This section presents a possible approach to implement a system that estimates the calibration parameters $(\boldsymbol{T}^a, \boldsymbol{K}^a, \boldsymbol{b}^a)$ of the accelerometer and $(\boldsymbol{T}^\omega, \boldsymbol{K}^\omega, \boldsymbol{b}^\omega)$ of the gyroscope, explained in Section 3.1, using the raw measurements of an IMU.

For this work, the software that performs the calibration was written in *C++*, using ROS as the underlying system (cf. Section 2.6). The advantage of ROS here is that for many IMU models drivers are already available, which send the measured values via topics, which can be used directly by this calibration software. For exact implementation details, please refer to the appendix.

Fig. 3.1 is intended to give a rough overview of the calibration procedures. The flow of the different data and the order of the steps are shown. In addition, algorithms 1 - 4 present some more complex procedures in more detail in the form of pseudo-code. The following is a brief introduction to the procedure.

1. **Accelerometer Calibration Parameter Estimation**
   To estimate the parameters, solely the raw measurement data of the acceleration are used. The procedure, which appears in Fig. 3.1 on the left and is shown in algorithm 2, is divided into the following two steps:
   (a) First, the median of the measured values is determined per standstill phase. Here, algorithm 1 is used to detect the standstills by checking whether the acceleration measurements remain constant over a certain period of time.
   (b) If sufficient medians (≥ 9) have been obtained, the calibration parameters are now estimated using Eq. (3.2).
2. **Correction of Acceleration Measurements**
   Using the estimated calibration parameters, the measurements of the accelerometer are now corrected using Eq. (2.12). The procedure appears in Fig. 3.1 at the center.
3. **Gyroscope Calibration Parameter Estimation**
   To estimate the calibration parameters, the corrected acceleration measurements and the raw measurement data of the angular velocity are used. The procedure, which appears in Fig. 3.1 on the left and is shown in algorithm 4, is divided into the following three steps:

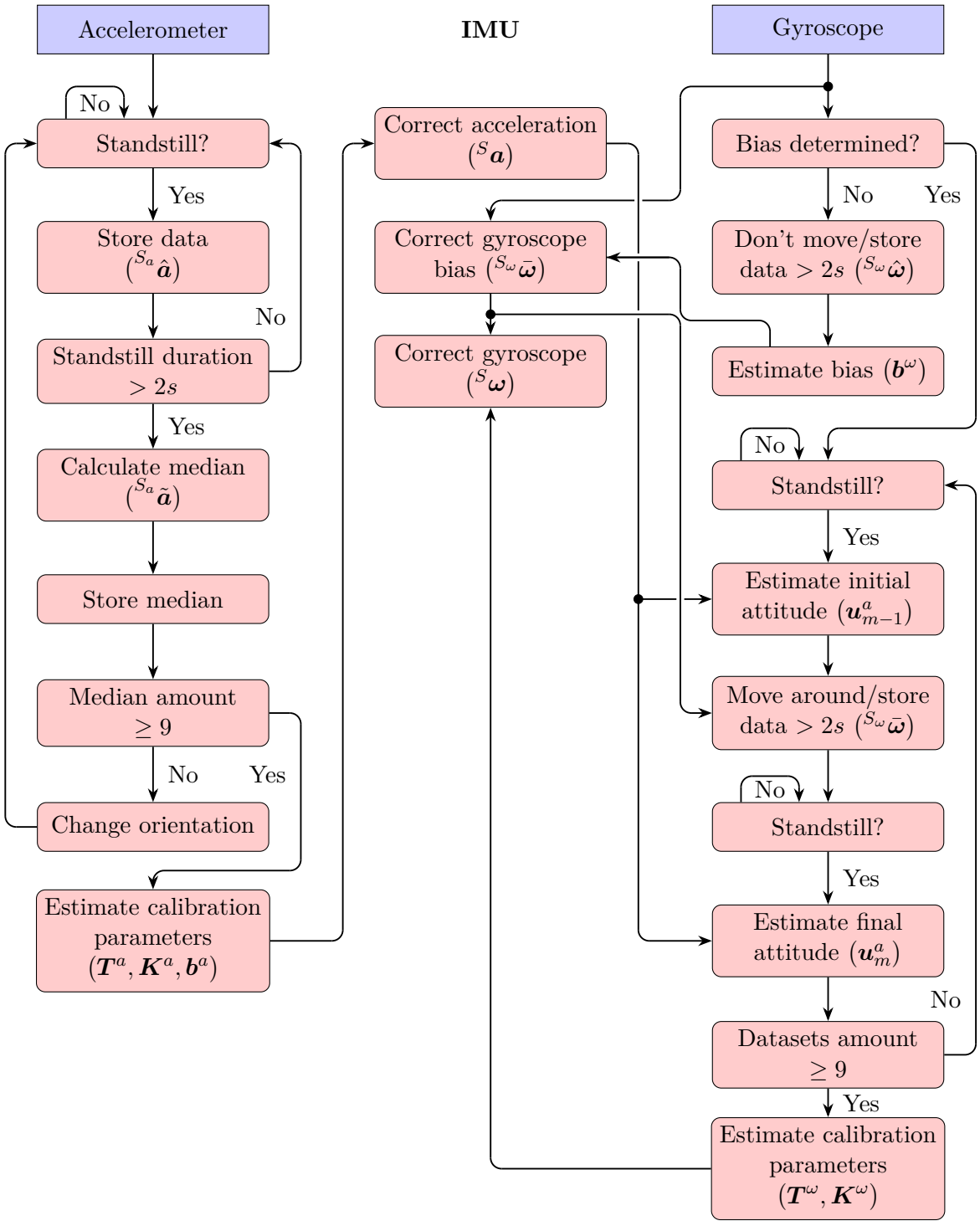

Figure 3.1: This diagram shows the structure of the procedure for calibrating the accelerometer and the gyroscope. The links between the individual blocks indicate the flow of the data and the order of the individual actions.

Algorithm 1: CheckForStillstand().
This algorithm uses a sequence of raw acceleration measurements $\left({}^{S_a}\hat{\boldsymbol{a}}\left(t_{0:N}\right)\right)$ to determine whether the latest measurement belongs to a standstill phase. It checks whether the past measured values of all sensor axes are within a defined tolerance band.

Input: ${}^{S_a}\hat{\boldsymbol{a}}(t_{0:N})$
Output: Returns true if ${}^{S_a}\hat{\boldsymbol{a}}(t_N)$ belongs to a standstill, false otherwise.

$StillstandThreshold \leftarrow 0.2$ // Values must be within this range for detection of standstill.

/* Amount of last measurements to be used for detection of standstill. Depending on the sampling rate and the desired time interval. */

$AmountNewestData \leftarrow 70$

/* Returns the *AmountNewestData* newest acceleration measurements. */

Function `GetNewestData(`*allMeasurments*`):`
  $newestMeasurements \leftarrow 0$
  for $i \leftarrow$ (lenght(*allMeasurments*) - *AmountNewestData*) to lenght(*allMeasurments*) **do**
    newestMeasurements.append(*allMeasurments*[i])
  return *newestMeasurements*

/* Returns the maximum and minimum measured acceleration in *newestMeasurements* for each sensor axis (x,y,z). */

Function `FindMinMaxValues(`*newestMeasurements*`):`
  $minValue \leftarrow 100$
  $maxValue \leftarrow 0$
  for $i \leftarrow 1$ to length(*newestMeasurements*) **do**
    if $newestMeasurements[i] > maxValue$ **then**
      $maxValue = newestMeasurements[i]$
    if $newestMeasurements[i] < minValue$ **then**
      $minValue = newestMeasurements[i]$
  return $minValue$, $maxValue$

/* Returns true if the newest measurement $\left({}^{S_a}\hat{\boldsymbol{a}}\left(t_N\right)\right)$ is within a standstill phase, false otherwise. */

Function `CheckForStillstand(`${}^{S_a}\hat{\boldsymbol{a}}\left(t_{0:N}\right)$`):`
  if length$\left({}^{S_a}\hat{\boldsymbol{a}}\left(t_{0:N}\right)\right) < AmountNewestData$ **then**
    return false // Too few measurements for evaluation.
  $newestMeasurements$ = GetNewestData$\left({}^{S_a}\hat{\boldsymbol{a}}\left(t_{0:N}\right)\right)$
  $minValue, maxValue$ = FindMinMaxValues($newestMeasurements$)
  if $(maxValue - minValue) > StillstandThreshold$ **then**
    return false // Sensor is moving.
  else
    return true // Stillstand detected.

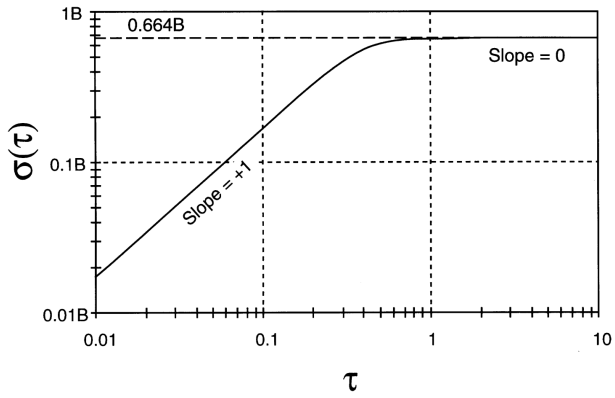

Figure 3.2: Plot for $\sigma_B(\tau)$. The bias instability coefficient B can be determined in the flat region. (Source: [12])

(a) First, the bias of the gyroscope is determined by averaging the measured angular velocity during a standstill.

(b) Now, data sets are generated, each containing the bias corrected measured angular velocities per motion phase together with the initial and final orientation, determined using the acceleration data. This procedure is shown in algorithm 3. Here, algorithm 1 is used to detect the standstills, respectively motion phases.

(c) If sufficient data sets have been obtained, the calibration parameters are now estimated using Eq. (3.5).

4. **Correction of Gyroscope Measurements**
Using the estimated calibration parameters, the measurements of the gyroscope are now corrected using Eq. (2.13). The procedure appears in Fig. 3.1 at the center.

3.3 Stochastic Error Coefficient Estimation

To estimate the stochastic error coefficients described in Section 2.3.2, the Allan Variance analysis technique mentioned in Section 2.2 is used. Note that these error parameters are not evaluated in the course of this work, since they are not used in the final software. For the sake of completeness, however, methods for determining these parameters will be discussed in the following.

Bias Instability

Substituting Eq. (2.14) into Eq. (2.6) and performing the integration yields

$$\sigma_B^2(\tau) = \frac{2B^2}{\pi}\left[\ln 2 - \frac{\sin^3 x}{2x^2}(\sin x + 4x\cos x) + C_i(2x) - C_i(4x)\right] \tag{3.56}$$

$$= \left(\frac{B}{0.6648}\right)^2 \qquad \text{for} \qquad \tau \gg \frac{1}{f_0}, \tag{3.57}$$

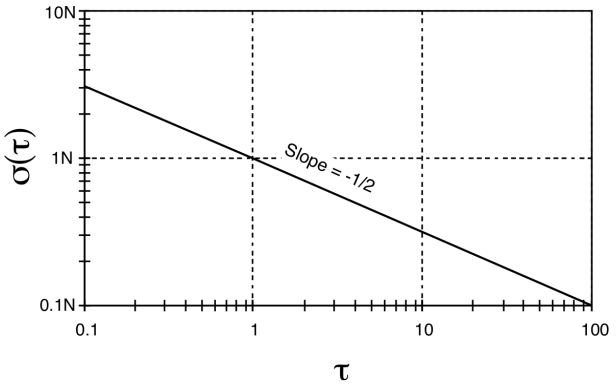

Figure 3.3: Plot for $\sigma_{\mathrm{ANRW}}(\tau)$ or $\sigma_{\mathrm{VRW}}(\tau)$. The ANRW or VRW coefficients Q can be determined at $\tau = 1$. (Source: [12])

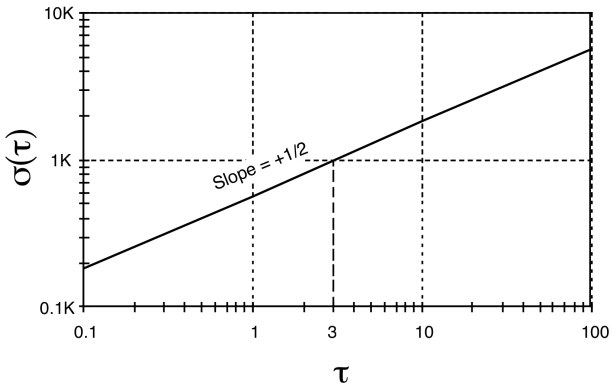

Figure 3.4: Plot for $\sigma_{\mathrm{RRW}}(\tau)$ or $\sigma_{\mathrm{ACRW}}(\tau)$. The RRW or ACRW coefficients K can be determined at $\tau = 3$. (Source: [12])

where $x = \pi f_0 \tau$ and C_i is the cosine-integral function. Fig. 3.2 shows a log-log plot of the square root of Eq. (3.56) vs τ for $f_0 = 1$. As shown by Eq. (3.57), the function reaches a plateau for τ much longer than the inverse cut off frequency. This flat region is now used to estimate the limit of B. [26]

ANRW / VRW

Substituting Eq. (2.15) into Eq. (2.6) and performing the integration yields

$$\sigma^2_{\mathrm{ANRW/VRW}}(\tau) = \frac{Q^2}{\tau}. \tag{3.58}$$

Fig. 3.3 shows a log-log plot of the square root of Eq. (3.58) vs τ, having an associated slope of $-1/2$. The coefficient Q can now be obtained for $\tau = 1$ directly from the plot.

RRW / ACRW

Substituting Eq. (2.16) into Eq. (2.6) and performing the integration yields

$$\sigma^2_{\mathrm{RRW/ACRW}}(\tau) = \frac{K^2 \tau}{3}. \tag{3.59}$$

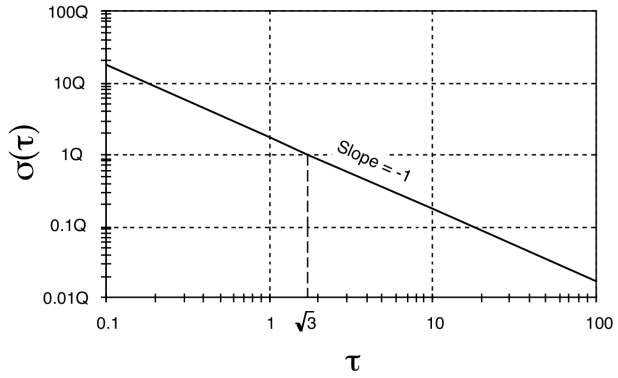

Figure 3.5: Plot for $\sigma_Q(\tau)$. The quantization noise coefficient Q is determined at $\tau = \sqrt{3}$. (Source: [12])

Fig. 3.4 shows a log-log plot of the square root of Eq. (3.59) vs τ, having an associated slope of 1/2. The coefficient K is now obtained for $\tau = 3$ directly from the plot.

Quantization Noise

Substituting Eq. (2.17) into Eq. (2.6) and performing the integration yields

$$\sigma_Q^2(\tau) = \frac{3Q^2}{\tau^2}. \tag{3.60}$$

Fig. 3.5 shows a log-log plot of the square root of Eq. (3.60) vs τ, having an associated slope of -1. The coefficient Q is now obtained for $\tau = \sqrt{3}$ directly from the plot.

3.4 Implementation Stochastic Error Parameter Estimation

As mentioned before, no evaluation of the stochastic error parameters will be done in the course of this work. Interested readers, who are nevertheless interested in a software for the estimation of these parameters, are recommended to use the ROS package *Allan Variance ROS* [4].

[4] `https://github.com/ori-drs/allan_variance_ros`

Algorithm 2: MainAccelerometerCalibration().
This algorithm uses raw accelerometer measurements $\left({}^{S_a}\hat{\boldsymbol{a}}\left(t_k\right)\right)$ to determine the calibration parameters $(\boldsymbol{T}^a, \boldsymbol{K}^a, \boldsymbol{b}^a)$ of the accelerometer. For this purpose, the median of the measured values of each standstill phase is determined.

Input: ${}^{S_a}\hat{\boldsymbol{a}}(t_k)$
Output: $\boldsymbol{T}^a, \boldsymbol{K}^a, \boldsymbol{b}^a$

$StillstandData \leftarrow 0$ // Stores acceleration data used for calculation of median.

/* Stores medians of acceleration data acquired during standstills, used for calibration parameter estimation. */

$Medians \leftarrow 0$

/* Stores acceleration measurements if they belong to a standstill phase, calculates median of measurements from previous standstill phase if orientation is changed. */

Function `ProcessData(`*measurements*`):`
  /* Method *CheckForStillstand* is explained in algorithm 1. */
  if CheckForStillstand(*measurements*) **then** // Store data for parameter estimation.
    $StillstandData$.append$\left({}^{S_a}\hat{\boldsymbol{a}}\left(t_k\right)\right)$
  else if length($StillstandData$) $>$ 500 **then**// Standstill phase was long enough.
    $Medians$.append(CalculateMedian($StillstandData$))
    $StillstandData = 0$
  else // Standstill phase was too short.
    $StillstandData = 0$

/* Returns the calibration parameters calculated using Eq. (3.2). */

Function `PerformCalibration(`*medians*`):`
  EstimateParameters($\boldsymbol{T}^a, \boldsymbol{K}^a, \boldsymbol{b}^a$, *medians*)
  return $\boldsymbol{T}^a, \boldsymbol{K}^a, \boldsymbol{b}^a$

/* Main function of calibration of the accelerometer. Triggers the calculation of calibration parameters once enough data has been collected. */

Function `MainAccelerometerCalibration:`
  $i \leftarrow 0$
  $measurements \leftarrow 0$ // Stores accelerometer data used for standstill detection.
  while length($Medians$) < 9 **do**
    if *isNewMeasurementArrived* **then**
      $measurements[i] = {}^{S_a}\hat{\boldsymbol{a}}(t_k)$
      $i = i + 1$
      ProcessData($measurements$)
  return PerformCalibration($Medians$)

Algorithm 3: ProcessData().
This algorithm creates data sets which are used for the calibration of the gyroscope. The angular velocities of each motion phase are stored together with the corresponding initial and final orientation.

Input: ${}^{S_\omega}\hat{\boldsymbol{\omega}}(t_k), {}^{S}\boldsymbol{a}(t_k), \boldsymbol{b}^{\omega}$
Output: Returns true if a new dataset for gyroscope calibration was stored, false otherwise.

```
1  OrientationData ← 0 // Stores acceleration data used for calculation of orientation.
2  GyroData ← 0 // Stores gyroscope data between successive standstills.
3  InitialOrientation ← 0 // Stores initial attitude before movement.
4  FinalOrientation ← 0 // Stores final attitude after movement.
5  Function ProcessData(accelerometerMeasurements, GyroDatasets):
       /* Method CheckForStillstand is explained in algorithm 1.                */
6      if CheckForStillstand(accelerometerMeasurements) then
           /* Store acceleration data for attitude estimation.                   */
7          OrientationData.append(S a (t_k))
8      else if length(OrientationData) > 500 then
           /* Enough data for attitude estimation was stored.                    */
9          if isInitialOrientationSet then
10             FinalOrientation = CalculateOrientation(OrientationData)
11             if length(GyroData) > 500 then
                   /* Enough gyroscope data was stored between successive standstills. */
12                 GyroDatasets.append(Initial/FinalOrientation, GyroData)
13             InitialOrientation = FinalOrientation
14             FinalOrientation = 0
15             GyroData = 0
16         else
17             InitialOrientation = CalculateOrientation(OrientationData)
18         OrientationData = 0
19         GyroData.append(S_ω ω̂ (t_k) − b^ω)
20     else if isInitialOrientationSet and OrientationData == 0 then
           /* Store gyroscope data between successive standstills.               */
21         GyroData.append(S_ω ω̂ (t_k) − b^ω)
22     else
           /* Too less data to estimate final or initial attitude.               */
23         OrientationData = 0
24         InitialOrientation = 0
25         GyroData = 0
```

Algorithm 4: MainGyroscopeCalibration().
This algorithm uses the angular velocity raw measurements $\left({}^{S_\omega}\hat{\boldsymbol{\omega}}\left(t_k\right)\right)$ to determine the calibration parameters of the gyroscope. First the bias ($\boldsymbol{b}^\omega$) is determined and then the remaining parameters ($\boldsymbol{T}^\omega, \boldsymbol{K}^\omega$).

Input: ${}^{S_\omega}\hat{\boldsymbol{\omega}}(t_k), {}^{S}\boldsymbol{a}(t_k)$
Output: $\boldsymbol{T}^\omega, \boldsymbol{K}^\omega, \boldsymbol{b}^\omega$

/* Stores data sets consisting of measured angular velocities and corresponding initial and final orientation. */
$GyroDatasets \leftarrow 0$
/* Returns the calibration parameters calculated using Eq. (3.5). */
Function `PerformCalibration(`*GyroDatasets*`):`
  EstimateParameters($\boldsymbol{T}^\omega, \boldsymbol{K}^\omega$, *GyroDatasets*)
  return $\boldsymbol{T}^\omega, \boldsymbol{K}^\omega$
/* Main function of calibration of the gyroscope. Triggers the calculation of calibration parameters once enough data has been collected and the bias of gyroscope is estimated. */
Function `MainGyroscopeCalibration:`
  $i \leftarrow 0$
  $gyroscopeMeasurements \leftarrow 0$ // Stores gyroscope data used for bias estimation.
  $accelerometerMeasurements \leftarrow 0$ // Stores accelerometer data used for standstill detection.
  $\boldsymbol{b}^\omega \leftarrow 0$ // Stores estimated bias of gyroscope.
  while length(*GyroDatasets*) < 9 **do**
    if *isNewMeasurementArrived* **then**
      if *isBiasEstimated* **then**
        $accelerometerMeasurements[i] = {}^{S}\boldsymbol{a}(t_k)$
        /* Method *ProcessData* is explained in algorithm 3. */
        ProcessData(*accelerometerMeasurements*, *GyroDatasets*)
        $i = i + 1$
      else
        $gyroscopeMeasurements[i] = {}^{S_\omega}\hat{\boldsymbol{\omega}}_k(t_k)$
        $i = i + 1$
        if length(*gyroscopeMeasurements*) > 500 **then**
          /* Calculates the bias of gyroscope by averaging the measurement. */
          $\boldsymbol{b}^\omega =$ EstimateBias(*gyroscopeMeasurements*)
          $i \leftarrow 0$
  return PerformCalibration(*GyroDatasets*)

Chapter 4

Camera-IMU Extrinsic Calibration

This chapter explains the procedure for determining the extrinsic parameters for calibrating a system consisting of a camera and an IMU. Therefore, the estimation of the visual scale, the rotation and translation between camera and IMU, as well as the gravity vector is described. The mathematical background is discussed, followed by a possible implementation of the calibration algorithms. Unless otherwise stated, the following methods for the estimation of the mentioned parameters are taken from the work [21].

4.1 IMU Pre-Integration

This section describes the preprocessing of IMU data, since later the non bias corrected IMU data ($\tilde{\boldsymbol{a}}$, $\tilde{\boldsymbol{\omega}}$) will be used to propagate position ($\boldsymbol{p}$), velocity ($\boldsymbol{v}$) and orientation ($\boldsymbol{q}$) states during the time interval $[t_k, t_{k+1}]$ between two consecutive frames $\{b_k\}$ and $\{b_{k+1}\}$. As follows the states are propagated in the world frame $\{w\}$:

$$\boldsymbol{p}^w_{b_{k+1}} = \boldsymbol{p}^w_{b_k} + \boldsymbol{v}^w_{b_k}\Delta t_k + \int\int_{t\in[t_k,t_{k+1}]} \Big(\boldsymbol{R}^w_t(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t} - \boldsymbol{n}_a) - \boldsymbol{g}^w\Big)\, dt^2 \tag{4.1}$$

$$\boldsymbol{v}^w_{b_{k+1}} = \boldsymbol{v}^w_{b_k} + \int_{t\in[t_k,t_{k+1}]} \Big(\boldsymbol{R}^w_t(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t} - \boldsymbol{n}_a) - \boldsymbol{g}^w\Big)\, dt \tag{4.2}$$

$$\boldsymbol{q}^w_{b_{k+1}} = \boldsymbol{q}^w_{b_k} \otimes \int_{t\in[t_k,t_{k+1}]} \Big(\frac{1}{2}\boldsymbol{\Omega}(\tilde{\boldsymbol{\omega}}_t - \boldsymbol{b}_{\omega_t} - \boldsymbol{n}_\omega)\boldsymbol{q}^{b_k}_t\Big)\, dt, \tag{4.3}$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}, \lfloor\boldsymbol{\omega}\rfloor_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{4.4}$$

Δt_k is the duration of the time interval $[t_k, t_{k+1}]$. $\boldsymbol{R}^w_t$ defines the rotation from the IMU frame to $\{w\}$ at time t. $\boldsymbol{q}^{b_k}_t$ defines the rotation, represented as quaternion, from the IMU frame to $\{b_k\}$ at time t. Note that the additive noise terms, which affect accelerometer measurements

($\boldsymbol{n}_a$) and gyroscope measurements ($\boldsymbol{n}_\omega$), are unknown and treated as zero in the following. The accelerometer bias ($\boldsymbol{b}_{a_t}$) and the gyroscope bias ($\boldsymbol{b}_{\omega_t}$) are modeled as random walk, whose derivatives are assumed to be Gaussian:

$$\begin{aligned}\dot{\boldsymbol{b}}_{a_t} &= \boldsymbol{n}_{b_a}\\ \dot{\boldsymbol{b}}_{\omega_t} &= \boldsymbol{n}_{b_\omega},\end{aligned}$$

where $\boldsymbol{n}_{b_a} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}_{b_a}^2)$ and $\boldsymbol{n}_{b_\omega} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}_{b_\omega}^2)$.

In summary, using Eqs. (4.1) - (4.3), the states ($\boldsymbol{p}^w_{b_{k+1}}$,$\boldsymbol{v}^w_{b_{k+1}}$,$\boldsymbol{q}^w_{b_{k+1}}$) can be estimated using the measurements of the IMU and the previous states ($\boldsymbol{p}^w_{b_k}$,$\boldsymbol{v}^w_{b_k}$,$\boldsymbol{q}^w_{b_k}$). Later, these states are used to formulate a minimization problem. For this reason, the IMU measurements are pre-integrated to save computing power required to find the solution. Furthermore, since only the states at the time of image acquisition are relevant, memory is saved, as not all IMU measurements have to be stored. In order to calculate the integrals in advance, the states are now considered in the body frame $\{b_k\}$ by applying the rotation $\boldsymbol{R}^{b_k}_w$, respectively $\boldsymbol{q}^{b_k}_w$, as follows:

$$\begin{aligned}
\boldsymbol{R}^{b_k}_w \boldsymbol{p}^w_{b_{k+1}} &= \boldsymbol{R}^{b_k}_w (\boldsymbol{p}^w_{b_k} + \boldsymbol{v}^w_{b_k}\Delta t_k) + \int\int_{t\in[t_k,t_{k+1}]} \boldsymbol{R}^{b_k}_w \left(\boldsymbol{R}^w_t(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t}) - \boldsymbol{g}^w\right) dt^2\\
\Leftrightarrow \boldsymbol{p}^{b_k}_{b_{k+1}} &= \boldsymbol{p}^{b_k}_{b_k} + \boldsymbol{v}^{b_k}_{b_k}\Delta t_k + \int\int_{t\in[t_k,t_{k+1}]} \left(\boldsymbol{R}^{b_k}_t(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t}) - \boldsymbol{g}^{b_k}\right) dt^2\\
\Leftrightarrow \boldsymbol{p}^{b_k}_{b_{k+1}} &= \boldsymbol{p}^{b_k}_{b_k} + \boldsymbol{v}^{b_k}_{b_k}\Delta t_k - \frac{1}{2}\boldsymbol{g}^{b_k}\Delta t_k^2 + \int\int_{t\in[t_k,t_{k+1}]} \left(\boldsymbol{R}^{b_k}_t(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t})\right) dt^2\\
\boxed{\Leftrightarrow \boldsymbol{p}^{b_k}_{b_{k+1}} &= \boldsymbol{p}^{b_k}_{b_k} + \boldsymbol{v}^{b_k}_{b_k}\Delta t_k - \frac{1}{2}\boldsymbol{g}^{b_k}\Delta t_k^2 + \boldsymbol{\alpha}^{b_k}_{b_{k+1}}} \qquad (4.5)\\
\boldsymbol{R}^{b_k}_w \boldsymbol{v}^w_{b_{k+1}} &= \boldsymbol{R}^{b_k}_w \boldsymbol{v}^w_{b_k} + \int_{t\in[t_k,t_{k+1}]} \boldsymbol{R}^{b_k}_w \left(\boldsymbol{R}^w_t(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t}) - \boldsymbol{g}^w\right) dt\\
\Leftrightarrow \boldsymbol{v}^{b_k}_{b_{k+1}} &= \boldsymbol{v}^{b_k}_{b_k} + \int_{t\in[t_k,t_{k+1}]} \left(\boldsymbol{R}^{b_k}_t(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t}) - \boldsymbol{g}^{b_k}\right) dt\\
\Leftrightarrow \boldsymbol{v}^{b_k}_{b_{k+1}} &= \boldsymbol{v}^{b_k}_{b_k} - \boldsymbol{g}^{b_k}\Delta t_k + \int_{t\in[t_k,t_{k+1}]} \left(\boldsymbol{R}^{b_k}_t(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t})\right) dt\\
\boxed{\Leftrightarrow \boldsymbol{v}^{b_k}_{b_{k+1}} &= \boldsymbol{v}^{b_k}_{b_k} - \boldsymbol{g}^{b_k}\Delta t_k + \boldsymbol{\beta}^{b_k}_{b_{k+1}}} \qquad (4.6)\\
\boldsymbol{q}^{b_k}_w \otimes \boldsymbol{q}^w_{b_{k+1}} &= \boldsymbol{q}^{b_k}_w \otimes \boldsymbol{q}^w_{b_k} \otimes \int_{t\in[t_k,t_{k+1}]} \left(\frac{1}{2}\boldsymbol{\Omega}(\tilde{\boldsymbol{\omega}}_t - \boldsymbol{b}_{\omega_t})\boldsymbol{q}^{b_k}_t\right) dt\\
\Leftrightarrow \boldsymbol{q}^{b_k}_{b_{k+1}} &= \boldsymbol{q}^{b_k}_{b_k} \otimes \int_{t\in[t_k,t_{k+1}]} \left(\frac{1}{2}\boldsymbol{\Omega}(\tilde{\boldsymbol{\omega}}_t - \boldsymbol{b}_{\omega_t})\boldsymbol{q}^{b_k}_t\right) dt\\
\boxed{\Leftrightarrow \boldsymbol{q}^{b_k}_{b_{k+1}} &= \boldsymbol{\gamma}^{b_k}_{b_{k+1}}} \qquad (4.7)
\end{aligned}$$

Following these transformations, it is possible to calculate the integrals, hence the integration of the raw IMU readings, in advance, since they are now independent of the rotation $\boldsymbol{R}^w_t$ and the gravity vector $\boldsymbol{g}^w$. Pre-integration terms follow, which are calculated using solely the IMU

measurements:

$$\boldsymbol{\alpha}_{b_{k+1}}^{b_k} = \int\int_{t\in[t_k,t_{k+1}]} \left(\boldsymbol{R}_t^{b_k}(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t})\right) dt^2 \tag{4.8}$$

$$\boldsymbol{\beta}_{b_{k+1}}^{b_k} = \int_{t\in[t_k,t_{k+1}]} \left(\boldsymbol{R}_t^{b_k}(\tilde{\boldsymbol{a}}_t - \boldsymbol{b}_{a_t})\right) dt \tag{4.9}$$

$$\boldsymbol{\gamma}_{b_{k+1}}^{b_k} = \int_{t\in[t_k,t_{k+1}]} \left(\frac{1}{2}\boldsymbol{\Omega}(\tilde{\boldsymbol{\omega}}_t - \boldsymbol{b}_{\omega_t})\boldsymbol{\gamma}_t^{b_k}\right) dt \tag{4.10}$$

Since we obtain the IMU measurements at discrete time points, we apply a numerical integration method in the following. In this case we apply the Euler method to estimate ($\hat{\cdot}$) the values of the pre-integration terms (4.8) - (4.10) as follows:

$$\hat{\boldsymbol{\alpha}}_{i+1}^{b_k} = \hat{\boldsymbol{\alpha}}_i^{b_k} + \hat{\boldsymbol{\beta}}_i^{b_k}\delta t + \frac{1}{2}\boldsymbol{R}(\hat{\boldsymbol{\gamma}}_i^{b_k})(\tilde{\boldsymbol{a}}_i - \boldsymbol{b}_{a_i})\delta t^2 \tag{4.11}$$

$$\hat{\boldsymbol{\beta}}_{i+1}^{b_k} = \hat{\boldsymbol{\beta}}_i^{b_k} + \boldsymbol{R}(\hat{\boldsymbol{\gamma}}_i^{b_k})(\tilde{\boldsymbol{a}}_i - \boldsymbol{b}_{a_i})\delta t \tag{4.12}$$

$$\hat{\boldsymbol{\gamma}}_{i+1}^{b_k} = \hat{\boldsymbol{\gamma}}_i^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\tilde{\boldsymbol{\omega}}_i - \boldsymbol{b}_{\omega_i})\delta t \end{bmatrix}, \tag{4.13}$$

where i is the discrete instant of time corresponding to a IMU measurement within the time interval $[t_k, t_{k+1}]$. Furthermore, the operator $\boldsymbol{R}(\cdot)$ converts an orientation quaternion into a rotation matrix and δt defines the time interval between two consecutive IMU measurements i and $i+1$.

In summary, the estimated pre-integrations ($\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}$,$\hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}$,$\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}$) between two consecutive frames $\{b_k\}$ and $\{b_{k+1}\}$ are calculated directly from the output of the IMU, which saves computation time and memory.

Later, the pre-integration results should be correctable using the estimated bias of the gyroscope, respectively the accelerometer, avoiding to recalculate the pre-integration (Eqs. (4.11) - (4.13)). In the next section, a procedure for this purpose is explained.

4.1.1 Rectify Pre-Integrated Results

If a new estimate of the bias is given, the non bias corrected pre-integrations ($\tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}, \tilde{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}, \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}$) are recalculated with the help of the new bias estimate. Since this would result in a large computational effort, they can alternatively be corrected for a given bias update $\boldsymbol{b} \leftarrow \bar{\boldsymbol{b}} + \delta\boldsymbol{b}$, where $\bar{\boldsymbol{b}}$ represents the previously estimated bias and $\delta\boldsymbol{b}$ the change, respectively the update. In the following $\delta\boldsymbol{b}_a$ denotes the bias update for the accelerometer and $\delta\boldsymbol{b}_\omega$ denotes the bias update for the gyroscope. In addition, the error terms $\delta\boldsymbol{\alpha}^{b_k}$ (position), $\delta\boldsymbol{\beta}^{b_k}$ (velocity) and $\delta\boldsymbol{\theta}^{b_k}$ (orientation) are introduced. The error term for the orientation is defined as a perturbation as follows:

$$\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \approx \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta}_t^{b_k} \end{bmatrix}. \tag{4.14}$$

Now the continuous-time linearized dynamics of error terms of Eqs. (4.8) - (4.10) can be derived:

$$\begin{bmatrix} \delta\dot{\boldsymbol{\alpha}}_t^{b_k} \\ \delta\dot{\boldsymbol{\beta}}_t^{b_k} \\ \delta\dot{\boldsymbol{\theta}}_t^{b_k} \\ \delta\dot{\boldsymbol{b}}_{a_t} \\ \delta\dot{\boldsymbol{b}}_{\omega_t} \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{I} & 0 & 0 & 0 \\ 0 & 0 & -\boldsymbol{R}_t^{b_k}\lfloor\tilde{\boldsymbol{\alpha}}_t - \boldsymbol{b}_{a_t}\rfloor_\times & -\boldsymbol{R}_t^{b_k} & 0 \\ 0 & 0 & -\lfloor\tilde{\boldsymbol{\omega}}_t - \boldsymbol{b}_{\omega_t}\rfloor_\times & 0 & -\boldsymbol{I} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\alpha}_t^{b_k} \\ \delta\boldsymbol{\beta}_t^{b_k} \\ \delta\boldsymbol{\theta}_t^{b_k} \\ \delta\boldsymbol{b}_{a_t} \\ \delta\boldsymbol{b}_{\omega_t} \end{bmatrix} \tag{4.15}$$

$$= \boldsymbol{F}_t \delta\boldsymbol{z}_t^{b_k} \tag{4.16}$$

Next, the first-order Jacobian matrix $\boldsymbol{J}_{b_{k+1}}$ of $\delta\boldsymbol{z}_{b_{k+1}}^{b_k}$ with respect to $\delta\boldsymbol{z}_{b_k}^{b_k}$ is computed recursively as follows:

$$\boldsymbol{J}_{t+\delta t} = (\boldsymbol{I} + \boldsymbol{F}_t \delta t)\boldsymbol{J}_t, \quad t \in [k, k+1]. \tag{4.17}$$

Now $\boldsymbol{J}_{b_{k+1}}$ is used to correct the pre-integration terms for a bias update $\delta\boldsymbol{b}_k$ as follows:

$$\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \approx \bar{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} + \boldsymbol{J}_{b_{a_{k+1}}}^{\alpha}\delta\boldsymbol{b}_{a_k} = \tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} + \boldsymbol{J}_{b_{a_{k+1}}}^{\alpha}\delta\boldsymbol{b}_{a_k} + \boldsymbol{J}_{b_{\omega_{k+1}}}^{\alpha}\delta\boldsymbol{b}_{\omega_k} \tag{4.18}$$

$$\hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \approx \bar{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} + \boldsymbol{J}_{b_{a_{k+1}}}^{\beta}\delta\boldsymbol{b}_{a_k} = \tilde{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} + \boldsymbol{J}_{b_{a_{k+1}}}^{\beta}\delta\boldsymbol{b}_{a_k} + \boldsymbol{J}_{b_{\omega_{k+1}}}^{\beta}\delta\boldsymbol{b}_{\omega_k} \tag{4.19}$$

$$\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \approx \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{J}_{b_{\omega_{k+1}}}^{\gamma}\delta\boldsymbol{b}_{\omega_k} \end{bmatrix}. \tag{4.20}$$

Here $\boldsymbol{J}_{b_{a_{k+1}}}^{\alpha}$ is the sub-block 3×3 matrix in $\boldsymbol{J}_{b_{k+1}}$, where the starting row is equivalent to the position of $\delta\boldsymbol{\alpha}_t^{b_k}$ in Eq. (4.15) (here 1) and the starting column is equivalent to the position of $\delta\boldsymbol{b}_{a_t}$ in Eq. (4.15) (here 10). The positions of $\boldsymbol{J}_{b_{\omega_{k+1}}}^{\alpha}$, $\boldsymbol{J}_{b_{a_{k+1}}}^{\beta}$, $\boldsymbol{J}_{b_{\omega_{k+1}}}^{\beta}$ and $\boldsymbol{J}_{b_{\omega_{k+1}}}^{\gamma}$ in $\boldsymbol{J}_{b_{k+1}}$ can be determined in the same way.

In summary, the Eqs. (4.18) - (4.20) can now be used to correct the pre-integrations when the bias changes, instead of recalculating them completely.

4.2 Gyroscope Bias and Rotation Estimation

This section describes how to use the IMU pre-integrations and the up-to-scale determined camera poses to find the rotation between the camera and the IMU. This is then used to determine the bias of the gyroscope. Using the estimate of the bias, the rotation between the camera and IMU is then refined and thus the estimate of the bias. This process is repeated until the values of the estimates remain stable.

4.2.1 Extrinsic Orientation Estimation

To estimate the rotation ($\boldsymbol{R}_c^b$) between the camera frame $\{c\}$ and the IMU frame $\{b\}$, the fact that both the camera and the IMU are subject to the same rotation during a movement is used.

Thus, the following notations are defined:

$$\boldsymbol{R}_{c_k}^{c_{k+1}} = \boldsymbol{R}_w^{c_{k+1}} \boldsymbol{R}_{c_k}^w \tag{4.21}$$

$$\boldsymbol{R}_{b_k}^{b_{k+1}} = \boldsymbol{R}_w^{b_{k+1}} \boldsymbol{R}_{b_k}^w, \tag{4.22}$$

where $\boldsymbol{R}_{c_k}^{c_{k+1}}$ is the rotation from camera frame $\{c_k\}$ to the subsequent frame $\{c_{k+1}\}$, obtained from camera data, and $\boldsymbol{R}_{b_k}^{b_{k+1}}$ is the rotation from IMU frame $\{b_k\}$ to the subsequent frame $\{b_{k+1}\}$, obtained from IMU data. As follows, we use $\boldsymbol{R}_c^b$ to make the Eqs. (4.21) and (4.22) represent the same rotation, namely from camera frame $\{c_k\}$ to IMU frame $\{b_{k+1}\}$:

$$\begin{aligned} \boldsymbol{R}_c^b \boldsymbol{R}_{c_k}^{c_{k+1}} &= \boldsymbol{R}_c^b \boldsymbol{R}_w^{c_{k+1}} \boldsymbol{R}_{c_k}^w \\ \boxed{\Leftrightarrow \boldsymbol{R}_{c_k}^{b_{k+1}} &= \boldsymbol{R}_c^b \boldsymbol{R}_w^{c_{k+1}} \boldsymbol{R}_{c_k}^w} \end{aligned} \tag{4.23}$$

$$\begin{aligned} \boldsymbol{R}_{b_k}^{b_{k+1}} \boldsymbol{R}_c^b &= \boldsymbol{R}_w^{b_{k+1}} \boldsymbol{R}_{b_k}^w \boldsymbol{R}_c^b \\ \boxed{\Leftrightarrow \boldsymbol{R}_{c_k}^{b_{k+1}} &= \boldsymbol{R}_w^{b_{k+1}} \boldsymbol{R}_{b_k}^w \boldsymbol{R}_c^b}. \end{aligned} \tag{4.24}$$

Now the Eqs. (4.23) and (4.24) are equated as follows:

$$\boldsymbol{R}_c^b \boldsymbol{R}_w^{c_{k+1}} \boldsymbol{R}_{c_k}^w = \boldsymbol{R}_w^{b_{k+1}} \boldsymbol{R}_{b_k}^w \boldsymbol{R}_c^b. \tag{4.25}$$

Since there is no knowledge about the world frame $\{w\}$ yet, the Eqs. (4.21) and (4.22) are used as follows:

$$\boldsymbol{R}_c^b \boldsymbol{R}_{c_k}^{c_{k+1}} = \boldsymbol{R}_{b_k}^{b_{k+1}} \boldsymbol{R}_c^b. \tag{4.26}$$

Note that $\boldsymbol{R}_{c_k}^{c_{k+1}}$ is calculated based on only the camera data and $\boldsymbol{R}_{b_k}^{b_{k+1}}$ is calculated using only IMU data, since both rotations represent relative rotation increments and are therefore independent of $\{w\}$. Now Eq. (4.26) is transferred into quaternion notation as follows:

$$\begin{aligned} \boldsymbol{q}_c^b \otimes \boldsymbol{q}_{c_k}^{c_{k+1}} &= \boldsymbol{q}_{b_k}^{b_{k+1}} \otimes \boldsymbol{q}_c^b \\ \Leftrightarrow \boldsymbol{q}_c^b \otimes \boldsymbol{q}_{c_k}^{c_{k+1}} - \boldsymbol{q}_{b_k}^{b_{k+1}} \otimes \boldsymbol{q}_c^b &= \boldsymbol{0} \\ \Leftrightarrow \left(\left[\boldsymbol{q}_{c_k}^{c_{k+1}} \right]_R - \left[\boldsymbol{q}_{b_k}^{b_{k+1}} \right]_L \right) \cdot \boldsymbol{q}_c^b &= \boldsymbol{0} \end{aligned} \tag{4.27}$$

Since the poses in the camera frame are given as $\boldsymbol{q}_{c_k}^{c_0}$ with respect to the frame $\{c_0\}$ of the first image, Eq. (4.27) changes as follows:

$$\left(\left[\boldsymbol{q}_{c_0}^{c_{k+1}} \otimes \boldsymbol{q}_{c_k}^{c_0} \right]_R - \left[\boldsymbol{q}_{b_k}^{b_{k+1}} \right]_L \right) \cdot \boldsymbol{q}_c^b = \boldsymbol{0} \tag{4.28}$$

$$\Leftrightarrow \left(\left[\left(\boldsymbol{q}_{c_{k+1}}^{c_0} \right)^{-1} \otimes \boldsymbol{q}_{c_k}^{c_0} \right]_R - \left[\boldsymbol{q}_{b_k}^{b_{k+1}} \right]_L \right) \cdot \boldsymbol{q}_c^b = \boldsymbol{0} \tag{4.29}$$

Furthermore, the rotational increment $\boldsymbol{q}_{b_k}^{b_{k+1}}$ is already given as gyroscope bias corrected (presumed that was already estimated) pre-integration ($\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}$), thus the equation is now transformed as follows:

$$\boxed{\left(\left[\boldsymbol{q}_{c_{k+1}}^{c_0}{}^{-1} \otimes \boldsymbol{q}_{c_k}^{c_0} \right]_R - \left[\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \right]_L \right) \cdot \boldsymbol{q}_c^b = \boldsymbol{Q}_{k,k+1} \cdot \boldsymbol{q}_c^b = \boldsymbol{0}} \tag{4.30}$$

If N datesets are given, based on Eq. (4.30) the following system of equations is formed:

$$\begin{bmatrix} w_{1,2} \cdot \boldsymbol{Q}_{1,2} \\ w_{2,3} \cdot \boldsymbol{Q}_{2,3} \\ \vdots \\ w_{N-1,N} \cdot \boldsymbol{Q}_{N-1,N} \end{bmatrix} \cdot \boldsymbol{q}_c^b = \boldsymbol{A} \cdot \boldsymbol{q}_c^b = \boldsymbol{0}. \tag{4.31}$$

At this point, the weight $w_{k,k+1}$ is introduced. This allows incorrect data sets to have less influence on the estimation of $\boldsymbol{q}_c^b$. The calculation of the weights will be discussed later. To lower the DOFs of $\boldsymbol{q}_c^b$ in the minimization problem below, the following parameterization, described in [30] is used:

$$\boldsymbol{q}_c^b = \begin{bmatrix} \cos\left(\frac{v}{2}\right) & \sin\left(\frac{v}{2}\right) \cdot \frac{u_1}{v} & \sin\left(\frac{v}{2}\right) \cdot \frac{u_2}{v} & \sin\left(\frac{v}{2}\right) \cdot \frac{u_3}{v} \end{bmatrix}^T, \tag{4.32}$$

$$\boldsymbol{u} = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T, \tag{4.33}$$

$$v = \sqrt{{u_1}^2 + {u_2}^2 + {u_3}^2}, \tag{4.34}$$

where $\boldsymbol{u}$ is the rotation axis vector and v is the norm of $\boldsymbol{u}$. Note that during optimization, there is no need to normalize the estimated quaternion, because the norm of Eq. (4.32) is always 1. Using Eq.(4.31), the following minimization problem is established, with the solution being an estimate of $\boldsymbol{u}$, respectively $\boldsymbol{q}_c^b$.

$$\min_{\boldsymbol{u}} \left\{ \left|\left| \boldsymbol{A} \cdot \boldsymbol{q}_c^b \right|\right| \right\} \tag{4.35}$$

An iterative strategy is applied here, where after each minimization, using the estimated $\hat{\boldsymbol{q}}_c^b$, the weights in Eq.(4.31) are calculated as follows:

$$\boldsymbol{e}_{k,k+1} = \boldsymbol{Q}_{k,k+1} \cdot \hat{\boldsymbol{q}}_c^b \tag{4.36}$$

$$w_{k,k+1} = \exp(-||\boldsymbol{e}_{k,k+1}|| \cdot K_0) \tag{4.37}$$

As a result, we obtain an estimate of the rotation $\hat{\boldsymbol{q}}_c^b$, respectively $\hat{\boldsymbol{R}}_c^b$, between camera frame $\{c\}$ and IMU frame $\{b\}$. This allows the estimation of the bias of the gyroscope as described in the next section.

4.2.2 Gyroscope Bias Estimation

Using the previously estimated rotation $\hat{\boldsymbol{q}}_c^b$, the poses determined by the camera is now rotated into the frame of the IMU as follows:

$$\boldsymbol{q}_{b_k}^{c_0} = \boldsymbol{q}_{c_k}^{c_0} \otimes \left(\hat{\boldsymbol{q}}_c^b\right)^{-1} \tag{4.38}$$

$$\boldsymbol{q}_{b_{k+1}}^{c_0} = \boldsymbol{q}_{c_{k+1}}^{c_0} \otimes \left(\hat{\boldsymbol{q}}_c^b\right)^{-1} \tag{4.39}$$

$$\Leftrightarrow \boldsymbol{q}_{b_k}^{b_{k+1}} = \left(\boldsymbol{q}_{b_{k+1}}^{c_0}\right)^{-1} \otimes \boldsymbol{q}_{b_k}^{c_0} \tag{4.40}$$

Since the bias corrected rotational increment $\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}$ (Eq. (4.20)) represents the same rotation as the rotational increment of the camera ($\boldsymbol{q}_{b_k}^{b_{k+1}}$), the following relation is established:

$$\boldsymbol{q}_{b_k}^{b_{k+1}} = \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{J}_{b_{\omega_{k+1}}}^{\gamma}\delta\boldsymbol{b}_{\omega} \end{bmatrix} \tag{4.41}$$

$$\Leftrightarrow \boldsymbol{q}_{b_k}^{b_{k+1}} = \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \tag{4.42}$$

$$\Leftrightarrow \boldsymbol{q}_{b_k}^{b_{k+1}} - \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} = \boldsymbol{0}. \tag{4.43}$$

Here $\delta\boldsymbol{b}_{\omega_k} = \delta\boldsymbol{b}_{\omega}$ is assumed, hence that the bias remains constant during initialization. With the help of Eq. (4.43), the following minimization problem is established:

$$\min_{\delta\boldsymbol{b}_{\omega}} \sum_{k=0}^{N-1} \left|\left| \left(\boldsymbol{q}_{b_k}^{b_{k+1}} - \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\right) w_{k,k+1} \right|\right|^2, \tag{4.44}$$

where N is the amount of datasets. An iterative strategy is applied here, where after each minimization, using the estimated $\delta\hat{\boldsymbol{b}}_{\omega}$, the weights in Eq.(4.44) are calculated as follows:

$$\boldsymbol{e}_{k,k+1} = \boldsymbol{q}_{b_k}^{b_{k+1}} - \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{J}_{b_{\omega_{k+1}}}^{\gamma}\delta\hat{\boldsymbol{b}}_{\omega} \end{bmatrix} \tag{4.45}$$

$$w_{k,k+1} = \exp(-||\boldsymbol{e}_{k,k+1}|| \cdot K_1) \tag{4.46}$$

If there is no prior estimate of the bias ($\bar{\boldsymbol{b}}_{\omega}$),

$$\boldsymbol{b}_{\omega} = \bar{\boldsymbol{b}}_{\omega} + \delta\boldsymbol{b}_{\omega} = \boldsymbol{0} + \delta\boldsymbol{b}_{\omega} = \delta\boldsymbol{b}_{\omega} \tag{4.47}$$

holds, otherwise

$$\boldsymbol{b}_{\omega} = \bar{\boldsymbol{b}}_{\omega} + \delta\boldsymbol{b}_{\omega} \tag{4.48}$$

applies. Using the estimated bias $\boldsymbol{b}_{\omega}$, the pre-integration terms $\left(\tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}, \tilde{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}, \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\right)$ are re-propagated, to obtain the gyroscope bias corrected pre-integrations $\left(\bar{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}, \bar{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}, \bar{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\right)$.

4.3 Scale, Gravity and Translation Estimation

This section deals with a method to estimate the scale s, the gravity $\boldsymbol{g}^{c_0}$ and the translation $\boldsymbol{p}_c^b$ between camera and IMU. Here note that an estimation of the bias of the gyroscope $\boldsymbol{b}_{\omega}$ and the rotation $\boldsymbol{q}_c^b$ between camera and IMU is given as explained in the previous sections. In the following, the fact is used that the position increment registered by the IMU must correspond to the position increment registered by the camera.

The up-to-scale determined poses ($\bar{\boldsymbol{p}}_{c_k}^{c_0}$) are transformed from the camera into the IMU frame with respect to $\{c_0\}$:

$$\boldsymbol{p}_{b_k}^{c_0} = s\bar{\boldsymbol{p}}_{b_k}^{c_0} = s\bar{\boldsymbol{p}}_{c_k}^{c_0} - \boldsymbol{R}_{b_k}^{c_0}\boldsymbol{p}_c^b \tag{4.49}$$

$$\boldsymbol{p}_{b_{k+1}}^{c_0} = s\bar{\boldsymbol{p}}_{b_{k+1}}^{c_0} = s\bar{\boldsymbol{p}}_{c_{k+1}}^{c_0} - \boldsymbol{R}_{b_{k+1}}^{c_0}\boldsymbol{p}_c^b, \tag{4.50}$$

where $\boldsymbol{R}_{b_k}^{c_0}$ is the rotation calculated with Eq. (4.38) in rotation matrix form and s the scaling parameter that aligns the visual structure to the metric scale. Continuing now the reference frame of Eq. (4.5) is changed from $\{b_k\}$ to $\{c_0\}$ as follows:

$$\boldsymbol{R}_{b_k}^{c_0}\boldsymbol{p}_{b_{k+1}}^{b_k} = \boldsymbol{R}_{b_k}^{c_0}\left(\boldsymbol{p}_{b_k}^{b_k} + \boldsymbol{v}_{b_k}^{b_k}\Delta t_k - \frac{1}{2}\boldsymbol{g}^{b_k}\Delta t_k^2 + \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}\right) \tag{4.51}$$

$$\Leftrightarrow \boldsymbol{p}_{b_{k+1}}^{c_0} = \boldsymbol{p}_{b_k}^{c_0} + \boldsymbol{v}_{b_k}^{c_0}\Delta t_k - \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_k^2 + \boldsymbol{R}_{b_k}^{c_0}\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}. \tag{4.52}$$

Here $\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}$ is the pre-integration $\tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}$ corrected for the bias of the gyroscope and the bias of the accelerometer (presumed that was already estimated). With the help of Eq. (4.49) and Eq. (4.50) the position terms in Eq. (4.52) are replaced and a relation for the velocity is obtained as follows:

$$s\bar{\boldsymbol{p}}_{c_{k+1}}^{c_0} - \boldsymbol{R}_{b_{k+1}}^{c_0}\boldsymbol{p}_c^b = s\bar{\boldsymbol{p}}_{c_k}^{c_0} - \boldsymbol{R}_{b_k}^{c_0}\boldsymbol{p}_c^b + \boldsymbol{v}_{b_k}^{c_0}\Delta t_k - \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_k^2 + \boldsymbol{R}_{b_k}^{c_0}\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \tag{4.53}$$

$$\Leftrightarrow \boldsymbol{v}_{b_k}^{c_0}\Delta t_k = s\bar{\boldsymbol{p}}_{c_{k+1}}^{c_0} - \boldsymbol{R}_{b_{k+1}}^{c_0}\boldsymbol{p}_c^b - s\bar{\boldsymbol{p}}_{c_k}^{c_0} + \boldsymbol{R}_{b_k}^{c_0}\boldsymbol{p}_c^b + \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_k^2 - \boldsymbol{R}_{b_k}^{c_0}\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \tag{4.54}$$

$$\Leftrightarrow \boldsymbol{v}_{b_k}^{c_0}\Delta t_k = \left(\bar{\boldsymbol{p}}_{c_{k+1}}^{c_0} - \bar{\boldsymbol{p}}_{c_k}^{c_0}\right)s + \left(\boldsymbol{R}_{b_k}^{c_0} - \boldsymbol{R}_{b_{k+1}}^{c_0}\right)\boldsymbol{p}_c^b + \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_k^2 - \boldsymbol{R}_{b_k}^{c_0}\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \tag{4.55}$$

$$\boxed{\Leftrightarrow \boldsymbol{v}_{b_k}^{c_0} = \left(\bar{\boldsymbol{p}}_{c_{k+1}}^{c_0} - \bar{\boldsymbol{p}}_{c_k}^{c_0}\right)\frac{s}{\Delta t_k} + \left(\boldsymbol{R}_{b_k}^{c_0} - \boldsymbol{R}_{b_{k+1}}^{c_0}\right)\frac{\boldsymbol{p}_c^b}{\Delta t_k} + \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_k - \frac{1}{\Delta t_k}\boldsymbol{R}_{b_k}^{c_0}\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}} \tag{4.56}$$

Using this equation, the velocity of frame $\{b_k\}$, hence the velocity of the IMU at the moment when the k-th image is captured, is determined with reference to the $\{c_0\}$ frame. Now the reference frame of Eq. (4.6) is also changed to $\{c_0\}$:

$$\boldsymbol{R}_{b_k}^{c_0}\boldsymbol{v}_{b_{k+1}}^{b_k} = \boldsymbol{R}_{b_k}^{c_0}\left(\boldsymbol{v}_{b_k}^{b_k} - \boldsymbol{g}^{b_k}\Delta t_k + \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}\right) \tag{4.57}$$

$$\Leftrightarrow \boldsymbol{v}_{b_{k+1}}^{c_0} = \boldsymbol{v}_{b_k}^{c_0} - \boldsymbol{g}^{c_0}\Delta t_k + \boldsymbol{R}_{b_k}^{c_0}\hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \tag{4.58}$$

Here $\hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}$ is the pre-integration $\tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}$ corrected for the bias of the gyroscope and the bias of the accelerometer (presumed that was already estimated). Now, using Eq. (4.56), the velocity

terms in Eq. (4.58) are eliminated as follows, setting $k=0$, $k+1=1$, and $k+2=2$ for clarity:

$$
\begin{aligned}
&(\bar{\boldsymbol{p}}_{c_2}^{c_0} - \bar{\boldsymbol{p}}_{c_1}^{c_0})\,\frac{s}{\Delta t_1} + \left(\boldsymbol{R}_{b_1}^{c_0} - \boldsymbol{R}_{b_2}^{c_0}\right)\frac{\boldsymbol{p}_c^b}{\Delta t_1} + \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_1 - \frac{1}{\Delta t_1}\boldsymbol{R}_{b_1}^{c_0}\hat{\boldsymbol{\alpha}}_{b_2}^{b_1} \\
&= (\bar{\boldsymbol{p}}_{c_1}^{c_0} - \bar{\boldsymbol{p}}_{c_0}^{c_0})\,\frac{s}{\Delta t_0} + \left(\boldsymbol{R}_{b_0}^{c_0} - \boldsymbol{R}_{b_1}^{c_0}\right)\frac{\boldsymbol{p}_c^b}{\Delta t_0} + \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_0 - \frac{1}{\Delta t_0}\boldsymbol{R}_{b_0}^{c_0}\hat{\boldsymbol{\alpha}}_{b_1}^{b_0} \\
&- \boldsymbol{g}^{c_0}\Delta t_0 + \boldsymbol{R}_{b_0}^{c_0}\hat{\boldsymbol{\beta}}_{b_1}^{b_0} \\
\Leftrightarrow &\frac{1}{\Delta t_0}\boldsymbol{R}_{b_0}^{c_0}\hat{\boldsymbol{\alpha}}_{b_1}^{b_0} - \frac{1}{\Delta t_1}\boldsymbol{R}_{b_1}^{c_0}\hat{\boldsymbol{\alpha}}_{b_2}^{b_1} - \boldsymbol{R}_{b_0}^{c_0}\hat{\boldsymbol{\beta}}_{b_1}^{b_0} \\
&= (\bar{\boldsymbol{p}}_{c_1}^{c_0} - \bar{\boldsymbol{p}}_{c_0}^{c_0})\,\frac{s}{\Delta t_0} - (\bar{\boldsymbol{p}}_{c_2}^{c_0} - \bar{\boldsymbol{p}}_{c_1}^{c_0})\,\frac{s}{\Delta t_1} \\
&+ \left(\boldsymbol{R}_{b_0}^{c_0} - \boldsymbol{R}_{b_1}^{c_0}\right)\frac{\boldsymbol{p}_c^b}{\Delta t_0} - \left(\boldsymbol{R}_{b_1}^{c_0} - \boldsymbol{R}_{b_2}^{c_0}\right)\frac{\boldsymbol{p}_c^b}{\Delta t_1} \\
&+ \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_0 - \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_1 - \boldsymbol{g}^{c_0}\Delta t_0 \\
\Leftrightarrow &\frac{1}{\Delta t_0}\boldsymbol{R}_{b_0}^{c_0}\hat{\boldsymbol{\alpha}}_{b_1}^{b_0} - \frac{1}{\Delta t_1}\boldsymbol{R}_{b_1}^{c_0}\hat{\boldsymbol{\alpha}}_{b_2}^{b_1} - \boldsymbol{R}_{b_0}^{c_0}\hat{\boldsymbol{\beta}}_{b_1}^{b_0} \\
&= \left((\bar{\boldsymbol{p}}_{c_1}^{c_0} - \bar{\boldsymbol{p}}_{c_0}^{c_0})\,\frac{1}{\Delta t_0} - (\bar{\boldsymbol{p}}_{c_2}^{c_0} - \bar{\boldsymbol{p}}_{c_1}^{c_0})\,\frac{1}{\Delta t_1}\right) s \\
&+ \left(\left(\boldsymbol{R}_{b_0}^{c_0} - \boldsymbol{R}_{b_1}^{c_0}\right)\frac{1}{\Delta t_0} - \left(\boldsymbol{R}_{b_1}^{c_0} - \boldsymbol{R}_{b_2}^{c_0}\right)\frac{1}{\Delta t_1}\right)\boldsymbol{p}_c^b \\
&- \frac{1}{2}\left(\Delta t_0 + \Delta t_1\right)\boldsymbol{g}^{c_0} \\
\Leftrightarrow &\Delta t_1 \boldsymbol{R}_{b_0}^{c_0}\hat{\boldsymbol{\alpha}}_{b_1}^{b_0} - \Delta t_0 \boldsymbol{R}_{b_1}^{c_0}\hat{\boldsymbol{\alpha}}_{b_2}^{b_1} - \Delta t_0 \Delta t_1 \boldsymbol{R}_{b_0}^{c_0}\hat{\boldsymbol{\beta}}_{b_1}^{b_0} \\
&= \left((\bar{\boldsymbol{p}}_{c_1}^{c_0} - \bar{\boldsymbol{p}}_{c_0}^{c_0})\,\Delta t_1 - (\bar{\boldsymbol{p}}_{c_2}^{c_0} - \bar{\boldsymbol{p}}_{c_1}^{c_0})\,\Delta t_0\right) s \\
&+ \left(\left(\boldsymbol{R}_{b_0}^{c_0} - \boldsymbol{R}_{b_1}^{c_0}\right)\Delta t_1 - \left(\boldsymbol{R}_{b_1}^{c_0} - \boldsymbol{R}_{b_2}^{c_0}\right)\Delta t_0\right)\boldsymbol{p}_c^b \\
&- \frac{1}{2}\left(\Delta t_0^2 \Delta t_1 + \Delta t_0 \Delta t_1^2\right)\boldsymbol{g}^{c_0}.
\end{aligned}
\tag{4.59}
$$

The following system of equations is now obtained:

$$
\left(\begin{bmatrix} \boldsymbol{\lambda}(k) & \boldsymbol{\zeta}(k) & \boldsymbol{\eta}(k) \end{bmatrix}\boldsymbol{\chi} - \boldsymbol{\iota}(k)\right) w_k = \boldsymbol{0}, \tag{4.60}
$$

with:

$$
\boldsymbol{\chi} = \begin{bmatrix} s & \boldsymbol{g}^{c_0} & \boldsymbol{p}_b^c \end{bmatrix}^T \tag{4.61}
$$

$$
\boldsymbol{\lambda}(k) = \left(\left(\bar{\boldsymbol{p}}_{c_{k+1}}^{c_0} - \bar{\boldsymbol{p}}_{c_k}^{c_0}\right)\Delta t_{k+1} - \left(\bar{\boldsymbol{p}}_{c_{k+2}}^{c_0} - \bar{\boldsymbol{p}}_{c_{k+1}}^{c_0}\right)\Delta t_k\right) \tag{4.62}
$$

$$
\boldsymbol{\zeta}(k) = -\frac{1}{2}\left(\Delta t_k^2 \Delta t_{k+1} + \Delta t_k \Delta t_{k+1}^2\right)\boldsymbol{I}_{3\times 3} \tag{4.63}
$$

$$
\boldsymbol{\eta}(k) = \left(\left(\boldsymbol{R}_{b_k}^{c_0} - \boldsymbol{R}_{b_{k+1}}^{c_0}\right)\Delta t_{k+1} - \left(\boldsymbol{R}_{b_{k+1}}^{c_0} - \boldsymbol{R}_{b_{k+2}}^{c_0}\right)\Delta t_k\right) \tag{4.64}
$$

$$
\boldsymbol{\iota}(k) = \Delta t_{k+1}\boldsymbol{R}_{b_k}^{c_0}\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} - \Delta t_k \boldsymbol{R}_{b_{k+1}}^{c_0}\hat{\boldsymbol{\alpha}}_{b_{k+2}}^{b_{k+1}} - \Delta t_k \Delta t_{k+1}\boldsymbol{R}_{b_k}^{c_0}\hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \tag{4.65}
$$

At this point, the weight w_k is introduced. This allows incorrect data sets to have less influence on the estimations of s, $\boldsymbol{g}^{c_0}$ and $\boldsymbol{p}_b^c$. Using Eq.(4.60), the following minimization problem is established, with the solution being an estimate of $\boldsymbol{\chi}$:

$$\min_{\boldsymbol{\chi}} = \sum_{k=0}^{N-2} \left|\left|\left(\begin{bmatrix} \boldsymbol{\lambda}(k) & \boldsymbol{\zeta}(k) & \boldsymbol{\eta}(k) \end{bmatrix} \boldsymbol{\chi} - \boldsymbol{\iota}(k)\right) w_k\right|\right|^2 \tag{4.66}$$

An iterative strategy is applied here, where after each minimization, using the estimate $\hat{\boldsymbol{\chi}}$, the weights in Eq. (4.60) are calculated as follows:

$$\boldsymbol{e}_k = \hat{s} \cdot \boldsymbol{\lambda}(k) + \boldsymbol{\zeta}(k)\hat{\boldsymbol{g}}^{c_0} + \boldsymbol{\eta}(k)\hat{\boldsymbol{p}}_c^b - \boldsymbol{\iota}(k) \tag{4.67}$$

$$w_k = \exp(-||\boldsymbol{e}_k|| \cdot K_2) \tag{4.68}$$

As a result, estimates are now obtained for the scale s, the gravity vector $\boldsymbol{g}^{c_0}$, and the translation between the camera and the IMU $\boldsymbol{p}_c^b$. These allow to determine the bias of the accelerometer and to optimize the estimates of s and $\boldsymbol{p}_c^b$ as described in the next section.

4.4 Refinement and Accelerometer Bias Estimation

This section describes how to use the previous estimates of scale, gravity vector and translation between camera and IMU to determine the bias of the accelerometer and also to optimize the estimates of scale and translation. At this point the fact is used that the magnitude of the gravitational vector is usually known. Now first of all the gravity vector is defined in the world frame $\{w\}$:

$$\boldsymbol{G}^w = \begin{bmatrix} 0 & 0 & G \end{bmatrix}, \tag{4.69}$$

assuming that $G = 9.807\frac{\mathrm{m}}{\mathrm{s}^2}$. Using the already estimated $\hat{\boldsymbol{g}}^{c_0}$, the rotation between the frames $\{w\}$ and $\{c_0\}$ can be obtained as follows. To achieve this, first the unit vectors of the two gravity vectors are determined:

$$\tilde{\boldsymbol{g}}^{c_0} = \frac{\hat{\boldsymbol{g}}^{c_0}}{||\hat{\boldsymbol{g}}^{c_0}||} \tag{4.70}$$

$$\tilde{\boldsymbol{G}}^w = \frac{\boldsymbol{G}^w}{||\boldsymbol{G}^w||} \tag{4.71}$$

Further the rotation axis ($\boldsymbol{v}$) and the rotation angle between θ the two vectors are determined by

$$\boldsymbol{v} = \frac{\tilde{\boldsymbol{g}}^{c_0} \times \tilde{\boldsymbol{G}}^w}{\left|\left|\tilde{\boldsymbol{g}}^{c_0} \times \tilde{\boldsymbol{G}}^w\right|\right|} \tag{4.72}$$

$$\theta = \operatorname{atan2}\left(\left|\left|\tilde{\boldsymbol{g}}^{c_0} \times \tilde{\boldsymbol{G}}^w\right|\right|, \tilde{\boldsymbol{g}}^{c_0} \cdot \tilde{\boldsymbol{G}}^w\right), \tag{4.73}$$

where ($\times$) denotes the cross product and ($\cdot$) denotes the dot product. Using Rodrigues' rotation formula the final rotation matrix is obtained:

$$\boldsymbol{R}_w^{c_0} = \boldsymbol{I} + \sin(\theta)\,[\boldsymbol{v}]_\times + (1 - \cos(\theta))\,[\boldsymbol{v}]_\times^2\,. \tag{4.74}$$

At this point it should be noted that the gravity vector $\hat{\boldsymbol{g}}^{c_0}$ is slightly distorted, since it was estimated under the assumption that the bias of the accelerometer is zero. Therefore, the rotation is extended by a perturbation ($\delta\boldsymbol{\theta}$) as follows:

$$\delta\boldsymbol{\theta} = \begin{bmatrix} \delta\boldsymbol{\theta}_{xy}^T & 0 \end{bmatrix}^T \tag{4.75}$$

$$\delta\boldsymbol{\theta}_{xy} = \begin{bmatrix} \delta\theta_x & \delta\theta_y \end{bmatrix}^T \tag{4.76}$$

$$\boldsymbol{G}^{c_0} = \boldsymbol{R}_w^{c_0} \mathrm{Exp}\left(\delta\boldsymbol{\theta}\right) \boldsymbol{G}^w \approx \boldsymbol{R}_w^{c_0} \boldsymbol{G}^w - \boldsymbol{R}_w^{c_0} \left[\boldsymbol{G}^w\right]_\times \delta\boldsymbol{\theta}, \tag{4.77}$$

where the first-order approximation $\mathrm{Exp}\left(\delta\boldsymbol{\theta}\right) \approx \boldsymbol{I} + \left[\delta\boldsymbol{\theta}\right]_\times$ is applied. Substituting Eq. (4.77) into Eq. (4.56) and taking into account the effect of the accelerometer bias (Eq. (4.18)), it follows:

$$\begin{aligned} \boldsymbol{v}_{b_k}^{c_0} &= \left(\bar{\boldsymbol{p}}_{c_{k+1}}^{c_0} - \bar{\boldsymbol{p}}_{c_k}^{c_0}\right) \frac{s}{\Delta t_k} + \left(\boldsymbol{R}_{b_k}^{c_0} - \boldsymbol{R}_{b_{k+1}}^{c_0}\right) \frac{\boldsymbol{p}_c^b}{\Delta t_k} + \frac{1}{2}\boldsymbol{g}^{c_0}\Delta t_k - \frac{1}{\Delta t_k}\boldsymbol{R}_{b_k}^{c_0}\bar{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \Leftrightarrow \boldsymbol{v}_{b_k}^{c_0} &= \left(\bar{\boldsymbol{p}}_{c_{k+1}}^{c_0} - \bar{\boldsymbol{p}}_{c_k}^{c_0}\right) \frac{s}{\Delta t_k} + \left(\boldsymbol{R}_{b_k}^{c_0} - \boldsymbol{R}_{b_{k+1}}^{c_0}\right) \frac{\boldsymbol{p}_c^b}{\Delta t_k} \\ &\quad + \frac{1}{2}\left(\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w - \boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times \delta\boldsymbol{\theta}\right)\Delta t_k - \frac{1}{\Delta t_k}\boldsymbol{R}_{b_k}^{c_0}\left(\bar{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} + \boldsymbol{J}_{b_{a_{k+1}}}^{\alpha}\delta\boldsymbol{b}_a\right) \end{aligned} \tag{4.78}$$

The same procedure is applied to Eq. (4.58):

$$\begin{aligned} \boldsymbol{v}_{b_{k+1}}^{c_0} &= \boldsymbol{v}_{b_k}^{c_0} - \boldsymbol{g}^{c_0}\Delta t_k + \boldsymbol{R}_{b_k}^{c_0}\bar{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ \Leftrightarrow \boldsymbol{v}_{b_{k+1}}^{c_0} &= \boldsymbol{v}_{b_k}^{c_0} - \left(\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w - \boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times \delta\boldsymbol{\theta}\right)\Delta t_k + \boldsymbol{R}_{b_k}^{c_0}\left(\bar{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} + \boldsymbol{J}_{b_{a_{k+1}}}^{\beta}\delta\boldsymbol{b}_a\right) \end{aligned} \tag{4.79}$$

Here $\delta\boldsymbol{b}_{a_k} = \delta\boldsymbol{b}_a$ is assumed, hence that the bias remains constant during initialization. Now, using Eq. (4.78), the velocity terms in Eq. (4.79) are eliminated as follows, setting $k = 0$,

$k+1=1$, and $k+2=2$ for clarity:

$$
\begin{aligned}
&(\bar{\boldsymbol{p}}_{c_2}^{c_0}-\bar{\boldsymbol{p}}_{c_1}^{c_0})\frac{s}{\Delta t_1}+\left(\boldsymbol{R}_{b_1}^{c_0}-\boldsymbol{R}_{b_2}^{c_0}\right)\frac{\boldsymbol{p}_c^b}{\Delta t_1}+\frac{1}{2}\left(\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w-\boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times\delta\boldsymbol{\theta}\right)\Delta t_1\\
&-\frac{1}{\Delta t_1}\boldsymbol{R}_{b_1}^{c_0}\left(\bar{\boldsymbol{\alpha}}_{b_2}^{b_1}+\boldsymbol{J}_{b_{a_2}}^{\alpha}\delta\boldsymbol{b}_a\right)\\
&=(\bar{\boldsymbol{p}}_{c_1}^{c_0}-\bar{\boldsymbol{p}}_{c_0}^{c_0})\frac{s}{\Delta t_0}+\left(\boldsymbol{R}_{b_0}^{c_0}-\boldsymbol{R}_{b_1}^{c_0}\right)\frac{\boldsymbol{p}_c^b}{\Delta t_0}+\frac{1}{2}\left(\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w-\boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times\delta\boldsymbol{\theta}\right)\Delta t_0\\
&-\frac{1}{\Delta t_0}\boldsymbol{R}_{b_0}^{c_0}\left(\bar{\boldsymbol{\alpha}}_{b_1}^{b_0}+\boldsymbol{J}_{b_{a_1}}^{\alpha}\delta\boldsymbol{b}_a\right)-\left(\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w-\boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times\delta\boldsymbol{\theta}\right)\Delta t_0\\
&+\boldsymbol{R}_{b_0}^{c_0}\left(\bar{\boldsymbol{\beta}}_{b_1}^{b_0}+\boldsymbol{J}_{b_{a_1}}^{\beta}\delta\boldsymbol{b}_a\right)\\
\Leftrightarrow&\frac{1}{\Delta t_0}\boldsymbol{R}_{b_0}^{c_0}\bar{\boldsymbol{\alpha}}_{b_1}^{b_0}-\frac{1}{\Delta t_1}\boldsymbol{R}_{b_1}^{c_0}\bar{\boldsymbol{\alpha}}_{b_2}^{b_1}-\boldsymbol{R}_{b_0}^{c_0}\bar{\boldsymbol{\beta}}_{b_1}^{b_0}\\
&+\frac{1}{2}\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w\Delta t_1-\frac{1}{2}\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w\Delta t_0+\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w\Delta t_0\\
&=(\bar{\boldsymbol{p}}_{c_1}^{c_0}-\bar{\boldsymbol{p}}_{c_0}^{c_0})\frac{s}{\Delta t_0}-(\bar{\boldsymbol{p}}_{c_2}^{c_0}-\bar{\boldsymbol{p}}_{c_1}^{c_0})\frac{s}{\Delta t_1}\\
&+\frac{1}{2}\boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times\delta\boldsymbol{\theta}\Delta t_1-\frac{1}{2}\boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times\delta\boldsymbol{\theta}\Delta t_0+\boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times\delta\boldsymbol{\theta}\Delta t_0\\
&+\frac{1}{\Delta t_1}\boldsymbol{R}_{b_1}^{c_0}\boldsymbol{J}_{b_{a_2}}^{\alpha}\delta\boldsymbol{b}_a-\frac{1}{\Delta t_0}\boldsymbol{R}_{b_0}^{c_0}\boldsymbol{J}_{b_{a_1}}^{\alpha}\delta\boldsymbol{b}_a+\boldsymbol{R}_{b_0}^{c_0}\boldsymbol{J}_{b_{a_1}}^{\beta}\delta\boldsymbol{b}_a\\
&+\left(\boldsymbol{R}_{b_0}^{c_0}-\boldsymbol{R}_{b_1}^{c_0}\right)\frac{\boldsymbol{p}_c^b}{\Delta t_0}-\left(\boldsymbol{R}_{b_1}^{c_0}-\boldsymbol{R}_{b_2}^{c_0}\right)\frac{\boldsymbol{p}_c^b}{\Delta t_1}\\
\Leftrightarrow&\frac{1}{\Delta t_0}\boldsymbol{R}_{b_0}^{c_0}\bar{\boldsymbol{\alpha}}_{b_1}^{b_0}-\frac{1}{\Delta t_1}\boldsymbol{R}_{b_1}^{c_0}\bar{\boldsymbol{\alpha}}_{b_2}^{b_1}-\boldsymbol{R}_{b_0}^{c_0}\bar{\boldsymbol{\beta}}_{b_1}^{b_0}+\frac{1}{2}\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w\left(\Delta t_0+\Delta t_1\right)\\
&=\left((\bar{\boldsymbol{p}}_{c_1}^{c_0}-\bar{\boldsymbol{p}}_{c_0}^{c_0})\frac{1}{\Delta t_0}-(\bar{\boldsymbol{p}}_{c_2}^{c_0}-\bar{\boldsymbol{p}}_{c_1}^{c_0})\frac{1}{\Delta t_1}\right)s\\
&+\left(\frac{1}{2}\boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times\left(\Delta t_0+\Delta t_1\right)\right)\delta\boldsymbol{\theta}\\
&+\left(\frac{1}{\Delta t_1}\boldsymbol{R}_{b_1}^{c_0}\boldsymbol{J}_{b_{a_2}}^{\alpha}-\frac{1}{\Delta t_0}\boldsymbol{R}_{b_0}^{c_0}\boldsymbol{J}_{b_{a_1}}^{\alpha}+\boldsymbol{R}_{b_0}^{c_0}\boldsymbol{J}_{b_{a_1}}^{\beta}\right)\delta\boldsymbol{b}_a\\
&+\left(\left(\boldsymbol{R}_{b_0}^{c_0}-\boldsymbol{R}_{b_1}^{c_0}\right)\frac{1}{\Delta t_0}-\left(\boldsymbol{R}_{b_1}^{c_0}-\boldsymbol{R}_{b_2}^{c_0}\right)\frac{1}{\Delta t_1}\right)\boldsymbol{p}_c^b\\
\Leftrightarrow&\Delta t_1\boldsymbol{R}_{b_0}^{c_0}\left(\bar{\boldsymbol{\alpha}}_{b_1}^{b_0}-\Delta t_0\bar{\boldsymbol{\beta}}_{b_1}^{b_0}\right)-\Delta t_0\boldsymbol{R}_{b_1}^{c_0}\bar{\boldsymbol{\alpha}}_{b_2}^{b_1}+\frac{1}{2}\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w\left(\Delta t_0^2\Delta t_1+\Delta t_0\Delta t_1^2\right)\\
&=\left((\bar{\boldsymbol{p}}_{c_1}^{c_0}-\bar{\boldsymbol{p}}_{c_0}^{c_0})\Delta t_1-(\bar{\boldsymbol{p}}_{c_2}^{c_0}-\bar{\boldsymbol{p}}_{c_1}^{c_0})\Delta t_0\right)s\\
&+\left(\frac{1}{2}\boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times\left(\Delta t_0^2\Delta t_1+\Delta t_0\Delta t_1^2\right)\right)\delta\boldsymbol{\theta}\\
&+\left(\Delta t_0\boldsymbol{R}_{b_1}^{c_0}\boldsymbol{J}_{b_{a_2}}^{\alpha}+\Delta t_1\boldsymbol{R}_{b_0}^{c_0}\left(\Delta t_0\boldsymbol{J}_{b_{a_1}}^{\beta}-\boldsymbol{J}_{b_{a_1}}^{\alpha}\right)\right)\delta\boldsymbol{b}_a\\
&+\left(\left(\boldsymbol{R}_{b_0}^{c_0}-\boldsymbol{R}_{b_1}^{c_0}\right)\Delta t_1-\left(\boldsymbol{R}_{b_1}^{c_0}-\boldsymbol{R}_{b_2}^{c_0}\right)\Delta t_0\right)\boldsymbol{p}_c^b
\end{aligned}
$$

The following system of equations is now obtained:

$$\Big(\Big[\begin{array}{cccc} \boldsymbol{\lambda}(k) & \boldsymbol{\phi}(k) & \boldsymbol{\kappa}(k) & \boldsymbol{\xi}(k) \end{array}\Big]\boldsymbol{\chi} - \boldsymbol{\psi}(k)\Big)\, w_k = \mathbf{0}, \tag{4.80}$$

with:

$$\begin{aligned}
\boldsymbol{\chi} &= \Big[\begin{array}{cccc} s & \delta\boldsymbol{\theta}_{xy} & \delta\boldsymbol{b}_a & \boldsymbol{p}_c^b \end{array}\Big]^T \\
\boldsymbol{\phi}(k) &= \left[\frac{1}{2}\boldsymbol{R}_w^{c_0}\left[\boldsymbol{G}^w\right]_\times\left(\Delta t_k^2\Delta t_{k+1} + \Delta t_k\Delta t_{k+1}^2\right)\right]_{(:,1:2)} \\
\boldsymbol{\kappa}(k) &= \Delta t_k \boldsymbol{R}_{b_{k+1}}^{c_0}\boldsymbol{J}_{b_{a_{k+2}}}^{\alpha} + \Delta t_{k+1}\boldsymbol{R}_{b_k}^{c_0}\left(\Delta t_k \boldsymbol{J}_{b_{a_{k+1}}}^{\beta} - \boldsymbol{J}_{b_{a_{k+1}}}^{\alpha}\right) \\
\boldsymbol{\xi}(k) &= \left(\boldsymbol{R}_{b_k}^{c_0} - \boldsymbol{R}_{b_{k+1}}^{c_0}\right)\Delta t_{k+1} - \left(\boldsymbol{R}_{b_{k+1}}^{c_0} - \boldsymbol{R}_{b_{k+2}}^{c_0}\right)\Delta t_k \\
\boldsymbol{\psi}(k) &= \Delta t_{k+1}\boldsymbol{R}_{b_k}^{c_0}\left(\bar{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} - \Delta t_k\bar{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}\right) - \Delta t_k\boldsymbol{R}_{b_{k+1}}^{c_0}\bar{\boldsymbol{\alpha}}_{b_{k+2}}^{b_{k+1}} \\
&\quad + \frac{1}{2}\boldsymbol{R}_w^{c_0}\boldsymbol{G}^w\left(\Delta t_k^2\Delta t_{k+1} + \Delta t_k\Delta t_{k+1}^2\right)
\end{aligned}$$

where $[\cdot]_{(:,1:2)}$ means the first two columns of the matrix. At this point, the weight w_k is introduced. This allows incorrect data sets to have less influence on the estimations of s, $\delta\boldsymbol{\theta}_{xy}$, $\delta\boldsymbol{b}_a$ and $\boldsymbol{p}_b^c$. Using Eq.(4.80), the following minimization problem is established, with the solution being an estimate of $\boldsymbol{\chi}$:

$$\min_{\boldsymbol{\chi}} = \sum_{k=0}^{N-2}\left|\left|\Big(\Big[\begin{array}{cccc} \boldsymbol{\lambda}(k) & \boldsymbol{\phi}(k) & \boldsymbol{\kappa}(k) & \boldsymbol{\xi}(k) \end{array}\Big]\boldsymbol{\chi} - \boldsymbol{\psi}(k)\Big)\, w_k\right|\right|^2 \tag{4.81}$$

An iterative strategy is applied here, where after each minimization, using the estimate $\hat{\boldsymbol{\chi}}$, the weights in Eq. (4.80) are calculated as follows:

$$\boldsymbol{e}_k = \hat{s}\cdot\boldsymbol{\lambda}(k) + \boldsymbol{\phi}(k)\delta\hat{\boldsymbol{\theta}}_{xy} + \boldsymbol{\kappa}(k)\delta\hat{\boldsymbol{b}}_a + \boldsymbol{\xi}(k)\boldsymbol{p}_c^b - \boldsymbol{\psi}(k) \tag{4.82}$$

$$w_k = \exp\left(-||\boldsymbol{e}_k||\cdot K_3\right) \tag{4.83}$$

Implementation details and the convergence criteria will be discussed later. As a result, estimates are now obtained for the scale s, the perturbation $\delta\boldsymbol{\theta}_{xy}$, the bias of the accelerometer and the translation between the camera and the IMU $\boldsymbol{p}_c^b$.

4.5 Implementation Gyroscope Bias and Extrinsic Rotation Estimation

This section presents an approach to implement a system that estimates the extrinsic rotation between a camera and an IMU $\left(\boldsymbol{q}_c^b\right)$ and the bias of a gyroscope $(\boldsymbol{b}_\omega)$, explained in Section 4.2, using the raw measurements of the gyroscope and the rotation data obtained by a camera.

For this work, the software that performs the estimation was written in *C++*, using ROS as the underlying system. The advantage of ROS here is that this software can be directly used

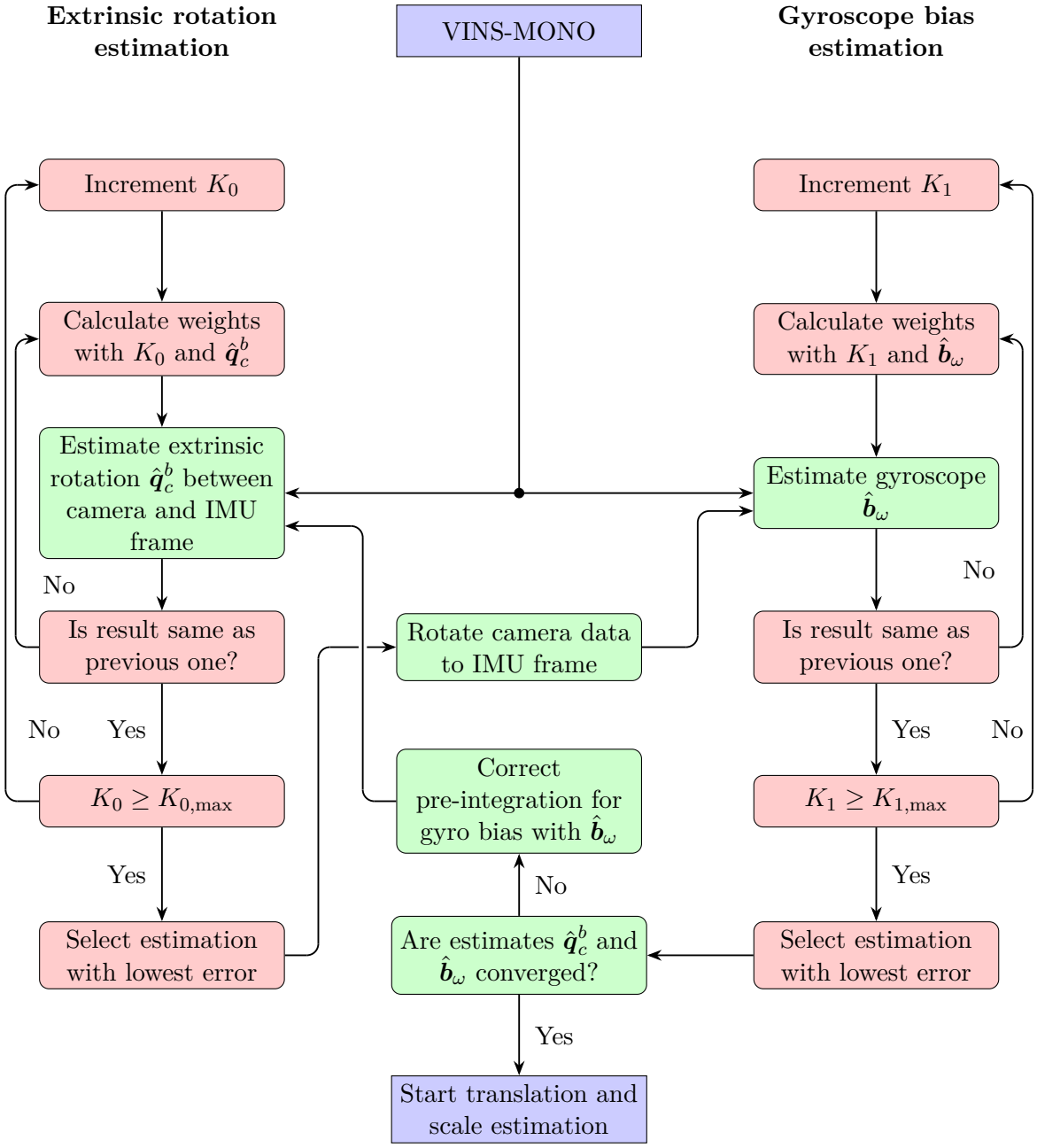

Figure 4.1: This diagram shows the structure of the procedure for estimation of the extrinsic rotation between camera and IMU and the bias of the gyroscope. The link between the individual blocks indicate the flow of the data and the order of the individual actions. Blocks shown in green represent the necessary steps to be performed if the weighting strategy is omitted.

Algorithm 5: PerformRotationEstimation().
This algorithm uses a sequence of gyroscope bias corrected pre-integrations $\left(\hat{\gamma}_{b_{k+1}}^{b_k}(T_{0:K})\right)$ and rotations from camera data $\left(\boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K})\right)$ to estimate the rotation between the camera and the IMU. The fact that both the camera and the IMU are subject to the same rotation during a movement is used. Note that the data sequences have already been cleaned of erroneous data sets.

Input: $\hat{\gamma}_{b_{k+1}}^{b_k}(T_{0:K})$, $\boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K})$

Output: Returns the estimate $\hat{\boldsymbol{q}}_c^b$ with the optimal weight factor K_0.

/* Performs the estimation of rotation for a fixed K_0. */
Function `PerformMinimizations`(K_0, $\boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K})$, $\hat{\gamma}_{b_{k+1}}^{b_k}(T_{0:K})$)**:**
  $\hat{\boldsymbol{q}}_c^b \leftarrow 0$ // Stores the latest estimated rotation from camera to IMU as quaternion.
  $relativeErrorChange \leftarrow 1$ // Stores the relative change of the error between two minimizations.
  $weights \leftarrow 1$ // Stores the weight per data set, where all weights are initially 1.
  $error \leftarrow 0$
  $previousError \leftarrow 1$
  /* Calculates $\hat{\boldsymbol{q}}_c^b$ until the weights converge. */
  while $relativeErrorChange > 0.01$ **do**
    /* Calculates the rotation between camera and IMU according to the minimization function (Eq. (4.35)) and returns its norm after minimization. */
    $error = \text{EstimateRotation}\left(\boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K}), \hat{\gamma}_{b_{k+1}}^{b_k}(T_{0:K}), weights, \hat{\boldsymbol{q}}_c^b\right)$
    $relativeErrorChange = |previousError - error| / previousError \cdot 100$
    $previousError = error$
    /* Calculates the weights of the corresponding data respectively measurements according to Eq. (4.37). */
    $weight = \text{CalculateWeight}\left(\boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K}), \hat{\gamma}_{b_{k+1}}^{b_k}(T_{0:K}), K_0, \hat{\boldsymbol{q}}_c^b\right)$
  return $\hat{\boldsymbol{q}}_c^b$ // Returns last estimate of rotation.
/* Main function of estimation of rotation between camera and IMU. */
Function `PerformRotationEstimation`($\hat{\gamma}_{b_{k+1}}^{b_k}(T_{0:K})$, $\boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K})$)**:**
  $\hat{\boldsymbol{q}}_c^b \leftarrow 0$ // Stores the latest estimated rotation from camera to IMU as quaternion.
  $error \leftarrow 0$
  $results \leftarrow 0$
  $K_0 \leftarrow 0$ // Initially set to zero, all weights are 1 in the first run.
  while $K_0 < K_{0,max}$ **do**
    $\hat{\boldsymbol{q}}_c^b = \text{PerformMinimizations}\left(K_0, \boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K}), \hat{\gamma}_{b_{k+1}}^{b_k}(T_{0:K})\right)$
    /* Calculates error of estimation according to Eq. (4.36). */
    $error = \text{CalculateError}\left(\boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K}), \hat{\gamma}_{b_{k+1}}^{b_k}(T_{0:K}), \hat{\boldsymbol{q}}_c^b\right)$
    $results.\text{append}(\hat{\boldsymbol{q}}_c^b, error)$
    $K_0 = K_0 + 1$
  /* Returns estimation with the lowest resulting error from all results. */
  return SearchEstimateWithMinError($results$)

Algorithm 6: PerformGyroBiasEstimation().
This algorithm uses a sequence of non gyroscope bias corrected pre-integrations $\left(\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K})\right)$ and rotations from camera data $\left(\boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K})\right)$ to estimate the bias of the gyroscope. The fact that both the camera and the IMU are supposed to measure the same rotation during a movement is used. Note that the data sequences have already been cleaned of erroneous data sets and the camera rotations are already given in the IMU frame.

Input: $\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K})$, $\boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K})$

Output: Returns the estimate $\hat{\boldsymbol{b}}_\omega$ with the optimal weight factor K_1.

/* Performs the estimation of rotation for a fixed K_1. */
Function `PerformMinimizations`(K_1, $\boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K})$, $\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K})$)**:**
  $\hat{\boldsymbol{b}}_\omega \leftarrow 0$ // Stores the latest estimated bias of the gyroscope.
  $relativeErrorChange \leftarrow 1$ // Stores the relative change of the error between two minimizations.
  $weights \leftarrow 1$ // Stores the weight per data set, where all weights are initially 1.
  $error \leftarrow 0$
  $previousError \leftarrow 0$
  /* Calculates $\hat{\boldsymbol{q}}_c^b$ until the weights converge. */
  while $relativeErrorChange > 0.01$ **do**
    /* Calculates the bias of the gyroscope according to the minimization function (Eq. (4.44)) and returns its norm after minimization. */
    $error = \text{EstimateBias}\left(\boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K}), \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K}), weights, \hat{\boldsymbol{b}}_\omega\right)$
    $relativeErrorChange = |previousError - error| / previousError \cdot 100$
    $previousError = error$
    /* Calculates the weights of the corresponding data respectively measurements according to Eq. (4.46). */
    $weight = \text{CalculateWeight}\left(\boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K}), \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K}), K_1, \hat{\boldsymbol{b}}_\omega\right)$
  return $\hat{\boldsymbol{b}}_\omega$ // Returns last estimate of bias.
/* Main function of estimation of gyroscope bias. */
Function `PerformGyroBiasEstimation`($\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K})$, $\boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K})$)**:**
  $\hat{\boldsymbol{b}}_\omega \leftarrow 0$ // Stores the latest estimated bias of the gyroscope.
  $error \leftarrow 0$
  $results \leftarrow 0$
  $K_1 \leftarrow 0$ // Initially set to zero, all weights are 1 in the first run.
  while $K_1 < K_{1,max}$ **do**
    $\hat{\boldsymbol{b}}_\omega = \text{PerformMinimizations}\left(K_1, \boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K}), \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K})\right)$
    /* Calculates error of estimation according to Eq. (4.45). */
    $error = \text{CalculateError}\left(\boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K}), \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K}), \hat{\boldsymbol{b}}_\omega\right)$
    $results$.append($\hat{\boldsymbol{b}}_\omega$, $error$)
    $K_1 = K_1 + 1$
  /* Returns estimation with the lowest resulting error from all results. */
  return SearchEstimateWithMinError($results$)

Algorithm 7: MainEstimateRotation().
This algorithm uses a sequence of non bias corrected pre-integrations $\left(\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\left(t_{0:K}\right)\right)$ and rotations from camera data $\left(\boldsymbol{q}_{c_k}^{c_{k+1}}\left(t_{0:K}\right)\right)$ to repeatedly estimate the rotation between camera and IMU and the bias of the gyroscope until the estimated values remain stable. Erroneous data sets are sorted out before the parameters are estimated.

Input: $\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(t_{0:K})$, $\boldsymbol{q}_{c_k}^{c_{k+1}}(t_{0:K})$
Output: $\hat{\boldsymbol{q}}_c^b$, $\hat{\boldsymbol{b}}_\omega$

Function `MainEstimateRotation`:
- $\hat{\boldsymbol{q}}_c^b \leftarrow 0$ // Stores the latest estimated rotation from camera to IMU as quaternion.
- $\hat{\boldsymbol{b}}_\omega \leftarrow 0$ // Stores the latest estimated bias of the gyroscope.
- $errorRotation \leftarrow 0$, $errorBias \leftarrow 0$
- $relativeErrorRotationChange \leftarrow 1$
- $previousErrorRotation \leftarrow 1$
- $relativeErrorBiasChange \leftarrow 1$
- $previousErrorBias \leftarrow 1$
- $\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K}) = \tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K})$
- **while** $relativeErrorRotationChange > 0.002$ **or** $relativeErrorBiasChange > 0.002$ **do**
  - /* Method *PerformRotationEstimation* is explained in algorithm 5. */
  - $\hat{\boldsymbol{q}}_c^b = \text{PerformRotationEstimation}\left(\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\left(T_{0:K}\right), \boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K})\right)$
  - /* Rotates camera determined rotations to IMU frame according to Eq. 4.40. */
  - $\boldsymbol{q}_{b_k}^{b_{k+1}}\left(T_{0:K}\right) = \text{RotateToIMUFrame}\left(\boldsymbol{q}_{c_k}^{c_{k+1}}\left(T_{0:K}\right), \hat{\boldsymbol{q}}_c^b\right)$
  - /* Method *PerformGyroBiasEstimation* is explained in algorithm 6. */
  - $\hat{\boldsymbol{b}}_\omega = \text{PerformGyroBiasEstimation}\left(\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K}), \boldsymbol{q}_{b_k}^{b_{k+1}}\left(T_{0:K}\right)\right)$
  - /* Correct the pre-integrations for gyroscope bias according to Eq. (4.20). */
  - $\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K}) = \text{CorrectForBias}\left(\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\left(T_{0:K}\right)\right)$
  - /* Calculates error of estimation according to Eq. (4.36). */
  - $errorRotation = \text{CalculateError}\left(\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\left(T_{0:K}\right), \boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K}), \hat{\boldsymbol{q}}_c^b\right)$
  - $relativeErrorRotationChange = \left|previousErrorRotation - errorRotation\right| / previousErrorRotation \cdot 100$
  - $previousErrorRotation = errorRotation$
  - /* Calculates error of estimation according to Eq. (4.45). */
  - $errorBias = \text{CalculateError}\left(\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K}), \boldsymbol{q}_{b_k}^{b_{k+1}}\left(T_{0:K}\right), \hat{\boldsymbol{b}}_\omega\right)$
  - $relativeErrorBiasChange = \left|previousErrorBias - errorBias\right| / previousErrorBias \cdot 100$
  - $previousErrorBias = errorBias$
- **return** $\hat{\boldsymbol{q}}_c^b$, $\hat{\boldsymbol{b}}_\omega$

by providing it the pre-integrated gyroscope measurements together with the rotations obtained with camera images. Here *VINS-Mono* (cf. Section 2.5) was used to pre-integrate the measurements and to obtain rotation information from camera images. For exact implementation details, please refer to the appendix.

Fig. 4.1 is intended to give a rough overview of the estimation procedures. The flow of the different data and the order of the steps can be seen. In addition, algorithms 5 - 7 present some more complex procedures in more detail in form of pseudo-code. The intention of the procedure is to find optimal values for the weighting parameters K_0 and K_1. Therefore, it is iteratively searched for K_0 and K_1 such that the datasets are weighted in a manner that the errors of the resulting estimates are minimized.

The following is a brief introduction to the procedure, where steps (2) - (8) are shown in detail in algorithm 7.

1. **Pre-Integrate Gyro Data and obtain Rotation Data from Camera Images**
   The camera acquired images are used to estimate the rotation between selected keyframes $\left(\boldsymbol{q}_{c_k}^{c_{k+1}}\left(t_{0:K}\right)\right)$. Next the gyroscope measurements are pre-integrated according to Eq. (4.13) between successive keyframes $\left(\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\left(t_{0:K}\right)\right)$, where the gyroscope bias is assumed to be zero. The result is a rotation estimate by the camera and one by the IMU, which are supposed to represent the same rotation in the respective coordinate system. This fact is used in the following to estimate the rotation between camera and IMU and the bias of the gyroscope. Note that the two calculations mentioned above are performed by *VINS-Mono*, as shown as the same named block in Fig. 4.1.

2. **Correct Pre-Integration for Gyroscope Bias**
   In this step, the pre-integrated gyroscope measurements are corrected for the bias of the gyroscope, according to Eq. (4.20). This is described in Fig. 4.1 in the middle. Note that this step can only be performed if an estimate of the bias already exists. The bias corrected data is now given as $\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\left(T_{0:K}\right)$.

3. **Estimate $\mathbf{K_{0,max}}$**
   To estimate an upper bound for K_0, Eq. 4.37 is rearranged as follows:

   $$K_{0,\max} = -\frac{\log(w_{\min})}{||\boldsymbol{e}_{\max}||}. \tag{4.84}$$

   Here $\boldsymbol{e}_{\max}$ is the maximum error that occurs in the data sets used, calculated with Eq. 4.36. The minimum weight that this data set is to get is given by $w_{\min}$. It has been found that $w_{\min} = 0.001$ is sufficient.

4. **Estimate Extrinsic Rotation**
   In this step, the bias corrected pre-integrations $\left(\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\left(T_{0:K}\right)\right)$ and the rotations obtained by camera $\left(\boldsymbol{q}_{c_k}^{c_{k+1}}\left(T_{0:K}\right)\right)$ are used to estimate the rotation between camera and IMU. Note that initially $K_0 = 0$ applies, thus all weights in the first run are 1. The procedure, which appears in Fig. 4.1 on the left and is shown in algorithm 5, is divided into the following five steps:

(a) Calculate weights with K_0 and $\hat{\boldsymbol{q}}_c^b$ according to Eq. (4.37).

(b) Estimate rotation $\hat{\boldsymbol{q}}_c^b$ according to Eq. (4.35).

(c) Repeat from step (a) until the estimate remains stable.

(d) Increment K_0 and repeat from step (a) until $K_0 \geq K_{0,\max}$.

(e) Select estimation $\hat{\boldsymbol{q}}_c^b(K_0)$ with lowest error (cf. Eq. (4.36)).

5. **Rotate Camera Data to IMU Frame**
In this step, the data from the camera $\left(\boldsymbol{q}_{c_k}^{c_{k+1}}(T_{0:K})\right)$ is rotated into the coordinate system of the IMU $\{b\}$ using the last estimate $\hat{\boldsymbol{q}}_c^b$ according to Eq. (4.40), resulting in $\left(\boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K})\right)$. This can be seen in Fig. 4.1 in the middle.

6. **Estimate $\mathbf{K_{1,max}}$**
To estimate an upper bound for K_1, Eq. 4.46 is rearranged as follows:

$$K_{1,\max} = -\frac{\log(w_{\min})}{||\boldsymbol{e}_{\max}||}. \tag{4.85}$$

Here $\boldsymbol{e}_{\max}$ is the maximum error that occurs in the data sets used, calculated with Eq. 4.45. The minimum weight that this data set is to get is given by $w_{\min}$. It has been found that $w_{\min} = 0.001$ is sufficient.

7. **Estimate Gyroscope Bias**
In this step, the non bias corrected pre-integrations $\left(\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}(T_{0:K})\right)$ and the rotations obtained by camera $\left(\boldsymbol{q}_{b_k}^{b_{k+1}}(T_{0:K})\right)$ are used to estimate the bias of the gyroscope. Note that initially $K_1 = 0$ applies, thus all weights in the first run are 1. The procedure, which appears in Fig. 4.1 on the right and is shown in algorithm 6, is divided into the following five steps:

(a) Calculate weights with K_1 and $\hat{\boldsymbol{b}}_\omega$ according to Eq. (4.37).

(b) Estimate bias $\hat{\boldsymbol{b}}_\omega$ according to Eq (4.44).

(c) Repeat from step (a) until the estimate remains stable.

(d) Increment K_1 and repeat from step (a) until $K_1 \geq K_{1,\max}$.

(e) Select estimation $\hat{\boldsymbol{b}}_\omega(K_1)$ with lowest error (cf. Eq. (4.45)).

8. **Repeat from step 3 until the estimates remain stable**

4.6 Implementation Scale, Gravity, Translation and Accelerometer Bias Estimation

This section presents an approach to implement a system that estimates the visual scale (s), a perturbation ($\delta\boldsymbol{\theta}_{xy}$), the gravity ($\boldsymbol{g}^{c_0}$), the bias of the accelerometer ($\delta\boldsymbol{b}_a$) and the translation

Algorithm 8: PerformScaleEstimation().
This algorithm uses a sequence of gyroscope and accelerometer bias corrected pre-integrations $\left(\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{c_0}(t_{0:N}), \hat{\boldsymbol{\beta}}_{b_{k+1}}^{c_0}(t_{0:N})\right)$, rotation and translation from camera data $\left(\bar{\boldsymbol{p}}_{c_k}^{c_0}(t_{0:N}), \boldsymbol{R}_{b_k}^{c_0}(t_{0:N})\right)$ and the corresponding durations $(\Delta t_k(t_{0:N}))$ of the time intervals $[t_k, t_{k+1}]$ to estimate scale, gravity and translation between camera and IMU frame. Note here that the bias of the accelerometer has not yet been estimated in the first run of this algorithm and is assumed to be zero.

Input: $\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{c_0}(t_{0:N})$, $\hat{\boldsymbol{\beta}}_{b_{k+1}}^{c_0}(t_{0:N})$, $\bar{\boldsymbol{p}}_{c_k}^{c_0}(t_{0:N})$, $\boldsymbol{R}_{b_k}^{c_0}(t_{0:N})$, $\Delta t_k(t_{0:N})$
Output: Returns estimates $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$ and $\hat{\boldsymbol{p}}_b^c$ with the optimal weight factor K_2.

/* Performs the estimation of scale, gravity and translation for a fixed K_2. */
Function `PerformMinimizations`(K_2, **Input**)**:**
  $\hat{s} \leftarrow 0$, $\hat{\boldsymbol{g}}^{c_0} \leftarrow 0$, $\hat{\boldsymbol{p}}_b^c \leftarrow 0$ // Stores the latest estimates.
  $relativeErrorChange \leftarrow 1$ // Stores the relative change of the error between two minimizations.
  $weights \leftarrow 1$ // Stores the weight per data set, where all weights are initially 1.
  $error \leftarrow 0$
  $previousError \leftarrow 1$
  /* Calculates $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$ and $\hat{\boldsymbol{p}}_b^c$ until the weights converge. */
  while $relativeErrorChange > 0.01$ **do**
    /* Calculates scale, gravity and translation according to the minimization function (Eq. (4.66)) and returns its norm after minimization. */
    $error =$ EstimateScale(**Input**, $weights$, $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$, $\hat{\boldsymbol{p}}_b^c$)
    $relativeErrorChange = |previousError - error| / previousError \cdot 100$
    $previousError = error$
    /* Calculates the weights of the corresponding data respectively measurements according to Eq. (4.68). */
    $weight =$ CalculateWeight(**Input**, K_2, $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$, $\hat{\boldsymbol{p}}_b^c$)
  return $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$, $\hat{\boldsymbol{p}}_b^c$ // Returns last estimates.
/* Main function of estimation of scale, gravity and translation. */
Function `PerformScaleEstimation`(**Input**)**:**
  $\hat{s} \leftarrow 0$, $\hat{\boldsymbol{g}}^{c_0} \leftarrow 0$, $\hat{\boldsymbol{p}}_b^c \leftarrow 0$ // Stores the latest estimates.
  $error \leftarrow 0$
  $results \leftarrow 0$
  $K_2 \leftarrow 0$ // Initially set to zero, all weights are 1 in the first run.
  while $K_2 < K_{2,max}$ **do**
    $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$, $\hat{\boldsymbol{p}}_b^c =$ PerformMinimizations(K_2, **Input**)
    /* Calculates error of estimation according to Eq. (4.67). */
    $error =$ CalculateError(**Input**, $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$, $\hat{\boldsymbol{p}}_b^c$)
    $results$.append($\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$, $\hat{\boldsymbol{p}}_b^c$, $error$)
    $K_2 = K_2 + 1$
  /* Returns estimation with the lowest resulting error from all results. */
  return SearchEstimateWithMinError($results$)

Algorithm 9: PerformRefinement().
This algorithm uses a sequence of gyroscope bias corrected pre-integrations $\left(\bar{\boldsymbol{\alpha}}^{c_0}_{b_{k+1}}(t_{0:N}), \bar{\boldsymbol{\beta}}^{c_0}_{b_{k+1}}(t_{0:N})\right)$, rotation and translation from camera data $\left(\bar{\boldsymbol{p}}^{c_0}_{c_k}(t_{0:N}), \boldsymbol{R}^{c_0}_{b_k}(t_{0:N})\right)$, the corresponding durations $(\Delta t_k(t_{0:N}))$ of the time intervals $[t_k, t_{k+1}]$ and the previously estimated gravity $(\hat{\boldsymbol{g}}^{c_0})$ to estimate scale, perturbation, accelerometer bias and translation between camera and IMU frame.

Input: $\bar{\boldsymbol{\alpha}}^{c_0}_{b_{k+1}}(t_{0:N})$, $\bar{\boldsymbol{\beta}}^{c_0}_{b_{k+1}}(t_{0:N})$, $\bar{\boldsymbol{p}}^{c_0}_{c_k}(t_{0:N})$, $\boldsymbol{R}^{c_0}_{b_k}(t_{0:N})$, $\Delta t_k(t_{0:N})$, $\hat{\boldsymbol{g}}^{c_0}$

Output: Returns estimates $\hat{s}$, $\delta\hat{\boldsymbol{\theta}}_{xy}$, $\delta\hat{\boldsymbol{b}}_a$ and $\hat{\boldsymbol{p}}^c_b$ with the optimal weight factor K_3.

/* Performs the estimation of scale, perturbation, accelerometer bias and translation for a fixed K_3. */
Function `PerformMinimizations`(K_3, **Input**)**:**
  $\hat{s} \leftarrow 0$, $\delta\hat{\boldsymbol{\theta}}_{xy} \leftarrow 0$, $\delta\hat{\boldsymbol{b}}_a \leftarrow 0$, $\hat{\boldsymbol{p}}^b_c \leftarrow 0$ // Stores the latest estimates.
  $relativeErrorChange \leftarrow 1$ // Stores the relative change of the error between two minimizations.
  $weights \leftarrow 1$ // Stores the weight per data set, where all weights are initially 1.
  $error \leftarrow 0$
  $previousError \leftarrow 1$
  /* Calculates $\hat{s}$, $\delta\hat{\boldsymbol{\theta}}_{xy}$, $\delta\hat{\boldsymbol{b}}_a$ and $\hat{\boldsymbol{p}}^c_b$ until the weights converge. */
  while $relativeErrorChange > 0.01$ **do**
    /* Calculates scale, perturbation, accelerometer bias and translation according to the minimization function (Eq. (4.81)) and returns its norm after minimization. */
    $error = \text{EstimateScale}\left(\textbf{Input}, weights, \hat{s}, \delta\hat{\boldsymbol{\theta}}_{xy}, \delta\hat{\boldsymbol{b}}_a, \hat{\boldsymbol{p}}^c_b\right)$
    $relativeErrorChange = |previousError - error| \,/ previousError \cdot 100$
    $previousError = error$
    /* Calculates the weights of the corresponding data respectively measurements according to Eq. (4.83). */
    $weight = \text{CalculateWeight}\left(\textbf{Input}, K_3, \hat{s}, \delta\hat{\boldsymbol{\theta}}_{xy}, \delta\hat{\boldsymbol{b}}_a, \hat{\boldsymbol{p}}^c_b\right)$
  return $\hat{s}$, $\delta\hat{\boldsymbol{\theta}}_{xy}$, $\delta\hat{\boldsymbol{b}}_a$, $\hat{\boldsymbol{p}}^c_b$ // Returns last estimates.
/* Main function of estimation of scale, perturbation, accelerometer bias and translation. */
Function `PerformRefinement`(**Input**)**:**
  $\hat{s} \leftarrow 0$, $\delta\hat{\boldsymbol{\theta}}_{xy} \leftarrow 0$, $\delta\hat{\boldsymbol{b}}_a \leftarrow 0$, $\hat{\boldsymbol{p}}^b_c \leftarrow 0$ // Stores the latest estimates.
  $error \leftarrow 0$
  $results \leftarrow 0$
  $K_3 \leftarrow 0$ // Initially set to zero, all weights are 1 in the first run.
  while $K_3 < K_{3,max}$ **do**
    $\hat{s}$, $\delta\hat{\boldsymbol{\theta}}_{xy}$, $\delta\hat{\boldsymbol{b}}_a$, $\hat{\boldsymbol{p}}^c_b = \text{PerformMinimizations}(K_3, \textbf{Input})$
    /* Calculates error of estimation according to Eq. (4.82). */
    $error = \text{CalculateError}\left(\textbf{Input}, \hat{s}, \delta\hat{\boldsymbol{\theta}}_{xy}, \delta\hat{\boldsymbol{b}}_a, \hat{\boldsymbol{p}}^c_b\right)$
    $results$.append($\hat{s}$, $\delta\hat{\boldsymbol{\theta}}_{xy}$, $\delta\hat{\boldsymbol{b}}_a$, $\hat{\boldsymbol{p}}^c_b$, $error$)
    $K_3 = K_3 + 1$
  /* Returns estimation with the lowest resulting error from all results. */
  return SearchEstimateWithMinError($results$)

Algorithm 10: MainEstimateScaleAndRefinement().
This algorithm uses a sequence of non bias corrected pre-integrations $\left(\tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}(t_{0:N}), \tilde{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}(t_{0:N})\right)$, rotation and translation from camera data $\left(\bar{\boldsymbol{p}}_{c_k}^{c_0}(t_{0:N}), \boldsymbol{R}_{c_k}^{c_0}(t_{0:N})\right)$ and the corresponding durations $(\Delta t_k(t_{0:N}))$ of the time intervals $[t_k, t_{k+1}]$ to repeatedly estimate scale, gravity, perturbation, accelerometer bias and translation between camera and IMU frame until the estimated values remain stable. The previously estimated rotation between camera and IMU $(\hat{\boldsymbol{q}}_b^c)$ and the bias of the gyroscope $\left(\hat{\boldsymbol{b}}_\omega\right)$ are used to rotate the pre-integrations into the camera frame, respectively to correct them for the bias of the gyroscope.

Input: $\tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}(t_{0:N})$, $\tilde{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}(t_{0:N})$, $\bar{\boldsymbol{p}}_{c_k}^{c_0}(t_{0:N})$, $\boldsymbol{R}_{c_k}^{c_0}(t_{0:N})$, $\Delta t_k(t_{0:N})$, $\hat{\boldsymbol{q}}_b^c$, $\hat{\boldsymbol{b}}_\omega$
Output: $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$, $\delta\hat{\boldsymbol{\theta}}_{xy}$, $\hat{\boldsymbol{b}}_a$, $\hat{\boldsymbol{p}}_b^c$

Function `MainEstimateScaleAndRefinement`:
- $\hat{s} \leftarrow 0$, $\hat{\boldsymbol{g}}^{c_0} \leftarrow 0$, $\delta\hat{\boldsymbol{\theta}}_{xy} \leftarrow 0$, $\delta\hat{\boldsymbol{b}}_a \leftarrow 0$, $\hat{\boldsymbol{p}}_c^b \leftarrow 0$ // Stores the latest estimates.
- $errorScale \leftarrow 0$, $errorRefine \leftarrow 0$
- $relativeErrorScaleChange \leftarrow 1$, $relativeErrorRefineChange \leftarrow 1$
- $previousErrorScale \leftarrow 1$, $previousErrorRefine \leftarrow 1$
- $correctedRotatedInput =$ rotateAndCorrectGyroBias(**Input**)
- $correctedAccBiasInput = correctedRotatedInput$
- **while** $relativeErrorScaleChange > 0.002$ **or** $relativeErrorRefineChange > 0.002$ **do**
  - /* Method *PerformScaleEstimation* is explained in algorithm 8. */
  - $\hat{s}, \hat{\boldsymbol{g}}^{c_0}, \hat{\boldsymbol{p}}_b^c =$ PerformScaleEstimation$(correctedAccBiasInput)$
  - /* Calculates error of estimation according to Eq. (4.67). */
  - $errorScale =$ CalculateError$(correctedAccBiasInput, \hat{s}, \hat{\boldsymbol{g}}^{c_0}, \hat{\boldsymbol{p}}_b^c)$
  - /* Method *PerformRefinement* is explained in algorithm 9. */
  - $\hat{s}, \delta\hat{\boldsymbol{\theta}}_{xy}, \delta\hat{\boldsymbol{b}}_a, \hat{\boldsymbol{p}}_b^c =$ PerformRefinement$(correctedRotatedInput, \hat{\boldsymbol{g}}^{c_0})$
  - /* Calculates error of estimation according to Eq. (4.82). */
  - $errorRefine =$ CalculateError$\left(correctedRotatedInput, \hat{s}, \delta\hat{\boldsymbol{\theta}}_{xy}, \delta\hat{\boldsymbol{b}}_a, \hat{\boldsymbol{p}}_b^c\right)$
  - /* Correct the pre-integrations for accelerometer bias according to Eqs. (4.18) and (4.19). */
  - $correctedAccBiasInput =$ CorrectForBias$\left(correctedRotatedInput, \delta\hat{\boldsymbol{b}}_a\right)$
  - $relativeErrorScaleChange = |previousErrorScale - errorScale| / previousErrorScale \cdot 100$
  - $previousErrorScale = errorScale$
  - $relativeErrorRefineChange = |previousErrorRefine - errorRefine| / previousErrorRefine \cdot 100$
  - $previousErrorRefine = errorRefine$
- **return** $\hat{s}, \hat{\boldsymbol{g}}^{c_0}, \delta\hat{\boldsymbol{\theta}}_{xy}, \hat{\boldsymbol{b}}_a, \hat{\boldsymbol{p}}_b^c$

between camera and IMU frame ($\boldsymbol{p}_b^c$), explained in Sections 4.3 and 4.4. Thereby it uses the gyroscope bias corrected measurements of an IMU and the rotations and up-to-scale poses obtained by a camera. Note that all calculates are carried out in the $\{c_0\}$ frame, by rotating the data with the previously estimated rotation $\left(\hat{\boldsymbol{q}}_c^b\right)$.

For this work, the software that performs the estimation was written in *C++*, using ROS as the underlying system. The advantage of ROS is that this software can be directly used by providing it the pre-integrated gyroscope and accelerometer measurements together with the rotations and up-to-scale poses obtained with camera images via topics. Here *VINS-Mono* was used to pre-integrate the measurements and to obtain rotation and pose information from camera images. For exact implementation details, please refer to the appendix.

Fig. 4.2 is intended to give a rough overview of the estimation procedures. The flow of the different data and the order of the steps can be seen. In addition, algorithms 8 - 10 present some more complex procedures in more detail in form of pseudo-code. The intention of the procedure is to find optimal values for the weighting parameters K_2 and K_3. Therefore, it is iteratively searched for K_2 and K_3 such that the datasets are weighted in a manner that the errors of the resulting estimates are minimized.

The following is a brief introduction to the procedure, where steps (2) - (9) are shown in detail in algorithm 10.

1. **Pre-Integrate IMU Data and obtain Pose Data from Camera Images**
   The camera acquired images are used to estimate the pose of selected keyframes $\left(\boldsymbol{R}_{c_k}^{c_0}\left(t_{0:N}\right), \bar{\boldsymbol{p}}_{c_k}^{c_0}\left(t_{0:N}\right)\right)$. Next the IMU measurements are pre-integrated according to Eqs. 4.11 and 4.12 between successive keyframes $\left(\tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}\left(t_{0:N}\right), \tilde{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}\left(t_{0:N}\right)\right)$, where both bias are assumed to be zero. Basically, the poses of the IMU and the camera are now given at the times of the keyframes. With the help of this, the desired parameters can now be estimated. Note that the two calculations mentioned above are performed by *VINS-MONO*, as shown as the same named block in Fig. 4.2.

2. **Correct Pre-Integrations for Gyroscope Bias**
   In this step, the pre-integrations are corrected for the bias of the gyroscope, according to Eqs. (4.18) and (4.19). This can be seen in Fig. 4.2 in the middle. Note that this step uses the previously estimated gyroscope bias $\hat{\boldsymbol{b}}_\omega$ (cf. Section 4.5). The bias corrected data is now given as $\bar{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}\left(t_{0:N}\right)$ and $\bar{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}\left(t_{0:N}\right)$.

3. **Rotate Data to Camera Frame**
   In this step, the data is rotated to represent the same motion defined in the same coordinate system. According to Eq. (4.40) the camera obtained rotations are transformed to get the rotation of the IMU with reference to the $\{c_0\}$ frame $\left(\boldsymbol{R}_{b_k}^{c_0}\left(t_{0:N}\right)\right)$ using the previously estimated extrinsic rotation $\hat{\boldsymbol{q}}_c^b$ (cf. Section 4.5). Using these transformed rotations the reference frame of the pre-integrations is changed to $\{c_0\}$ (cf. Eqs.(4.52) and (4.58)). That gives the position pre integration $\left(\bar{\boldsymbol{\alpha}}_{b_{k+1}}^{c_0}\left(t_{0:N}\right)\right)$ and the velocity pre integration $\left(\bar{\boldsymbol{\beta}}_{b_{k+1}}^{c_0}\left(t_{0:N}\right)\right)$ with reference to the camera frame. This step can be seen in Fig. 4.2.

4. **Correct Pre-Integrations for Accelerometer Bias**
   In this step, the pre-integrations are corrected for the bias of the accelerometer, according to Eqs. (4.18) and (4.19). This can be seen in Fig. 4.2 in the middle. Note that this step can only be performed if an estimate of the bias already exists. The accelerometer bias corrected data is now given as $\hat{\boldsymbol{\alpha}}^{c_0}_{b_{k+1}}(t_{0:N})$ and $\hat{\boldsymbol{\beta}}^{c_0}_{b_{k+1}}(t_{0:N})$.

5. **Estimate $\mathbf{K_{2,max}}$**
   To estimate an upper bound for K_2, Eq. 4.68 is rearranged as follows:

$$K_{2,\max} = -\frac{\log(w_{\min})}{||\boldsymbol{e}_{\max}||}. \tag{4.86}$$

   Here $\boldsymbol{e}_{\max}$ is the maximum error that occurs in the data sets used, calculated with Eq. 4.67. The minimum weight that this data set is to get is given by $w_{\min}$. It has been found that $w_{\min} = 0.001$ is sufficient.

6. **Estimate Scale**
   In this step, the accelerometer and gyroscope bias corrected pre-integrations $\left(\hat{\boldsymbol{\alpha}}^{c_0}_{b_{k+1}}(t_{0:N}), \hat{\boldsymbol{\beta}}^{c_0}_{b_{k+1}}(t_{0:N})\right)$ and pose informations obtained by camera $\left(\boldsymbol{R}^{c_0}_{b_k}(t_{0:N}), \bar{\boldsymbol{p}}^{c_0}_{c_k}(t_{0:N})\right)$ are used to estimate the visual scale, the gravity and the translation between camera and IMU. Note that initially $K_2 = 0$ applies, thus all weights in the first run are 1. The procedure, which appears in Fig. 4.2 on the left and is shown in algorithm 8, is divided into the following five steps:

   (a) Calculate weights with K_2, $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$ and $\hat{\boldsymbol{p}}^c_b$ according to Eq. (4.68).

   (b) Estimate $\hat{s}$, $\hat{\boldsymbol{g}}^{c_0}$ and $\hat{\boldsymbol{p}}^c_b$ according to Eq. (4.66).

   (c) Repeat from step (a) until estimates remain stable.

   (d) Increment K_2 and repeat from step (a) until $K_2 \geq K_{2,\max}$.

   (e) Select estimations $\hat{s}(K_2)$, $\hat{\boldsymbol{g}}^{c_0}(K_2)$ and $\hat{\boldsymbol{p}}^c_b(K_2)$ with lowest error (cf. Eq. (4.67)).

7. **Estimate $\mathbf{K_{3,max}}$**
   To estimate an upper bound for K_3, Eq. 4.83 is rearranged as follows:

$$K_{3,\max} = -\frac{\log(w_{\min})}{||\boldsymbol{e}_{\max}||}. \tag{4.87}$$

   Here $\boldsymbol{e}_{\max}$ is the maximum error that occurs in the data sets used, calculated with Eq. 4.82. The minimum weight that this data set is to get is given by $w_{\min}$. It has been found that $w_{\min} = 0.001$ is sufficient.

8. **Estimate Accelerometer Bias**
   In this step, the non accelerometer bias corrected pre-integrations $\left(\bar{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}}(t_{0:N}), \bar{\boldsymbol{\beta}}^{b_k}_{b_{k+1}}(t_{0:N})\right)$, pose informations obtained by camera $\left(\boldsymbol{R}^{c_0}_{b_k}(t_{0:N}), \bar{\boldsymbol{p}}^{c_0}_{c_k}(t_{0:N})\right)$ and the previously estimated gravity $(\hat{\boldsymbol{g}}^{c_0})$ are used to estimate the visual scale, the perturbation, the accelerometer bias and the translation between camera and IMU. Note that

in this case the bias of the accelerometer is taken into account in the estimation and the results for scale and translation are expected to have a lower error compared to the estimation in step (5). Consider that initially $K_3 = 0$ applies, thus all weights in the first run are 1. The procedure, which appears in Fig. 4.2 on the right and is shown in algorithm 9, is divided into the following five steps:

(a) Calculate weights with K_3, $\hat{s}$, $\delta\hat{\boldsymbol{\theta}}_{xy}$, $\delta\hat{\boldsymbol{b}}_a$ and $\hat{\boldsymbol{p}}_b^c$ according to Eq. (4.83).

(b) Estimate $\hat{s}$, $\delta\hat{\boldsymbol{\theta}}_{xy}$, $\delta\hat{\boldsymbol{b}}_a$ and $\hat{\boldsymbol{p}}_b^c$ according to Eq. (4.81).

(c) Repeat from step (a) until estimates remain stable.

(d) Increment K_3 and repeat from step (a) until $K_3 \geq K_{3,\max}$.

(e) Select estimations $\hat{s}\,(K_3)$, $\delta\hat{\boldsymbol{\theta}}_{xy}\,(K_3)$, $\delta\hat{\boldsymbol{b}}_a\,(K_3)$ and $\hat{\boldsymbol{p}}_b^c\,(K_3)$ with lowest error (cf. Eq. (4.82)).

9. **Repeat from step 4 until the estimates remain stable**

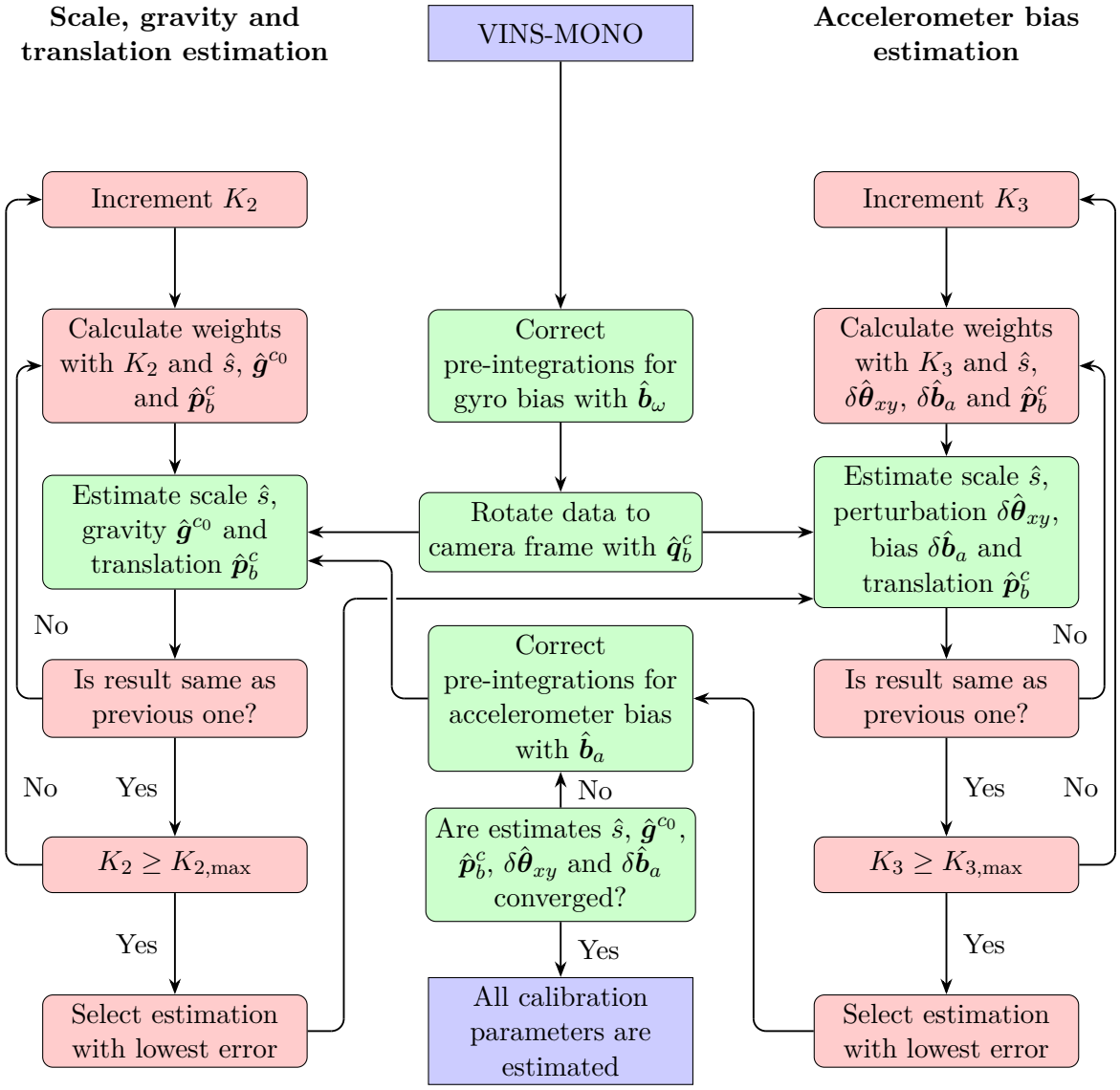

Figure 4.2: This diagram shows the structure of the procedure for estimation of the scale, gravity, translation and accelerometer bias. The link between the individual blocks indicate the flow of the data and the order of the individual actions. Blocks shown in green represent the necessary steps to be performed if the weighting strategy is omitted.

Chapter 5

Evaluation of IMU Calibration

The purpose of this chapter is to demonstrate the performance of the system described in Chapter 3. Thereby the calibration of an accelerometer and a gyroscope is investigated. For experiments within this chapter a low-cost IMU manufactured by *Phidgets* is used. Its type definition is stated as PhidgetSpatial Precision 3/3/3 High Resolution [1].

5.1 Evaluation of Accelerometer Calibration

In this section, the results of calibration of the accelerometer are described. The underlying procedure is shown in Fig. 3.1 on the left side.

In the course of the calibration of the accelerometer, a dataset with a length of about $133s$ was recorded, whereby the orientation of the IMU was changed 15 times by hand. Care was taken to achieve a large variation in orientation. Looking at Fig. 5.1, the red points represent the norm of the measured gravity vector during the standstill phases, the green ones after the calibration. Here it is noticeable that the calibration has an enormous effect on the accuracy of the measurement. If one examines additionally table 5.1, which expresses the results in numerical values, it is noticeable that by the calibration almost the desired value of $9.81007\frac{\mathrm{m}}{\mathrm{s}^2}$ is reached on average. Also the error of the gravity measurement is lowered by the calibration around the factor 30.

[1] `https://www.phidgets.com/?catid=10&pcid=8&prodid=1158`

Source	**Gravity norm**	**Error**
Without calibration	9.7731	0.4463
With calibration	9.8068	0.01518

Table 5.1: Shows the results of the calibration of the accelerometer in the form of the measured norm of gravity in $\frac{\mathrm{m}}{\mathrm{s}^2}$. Here the data of Fig. 5.1 were used, where the norm corresponds to the arithmetic mean and the error to the standard deviation.

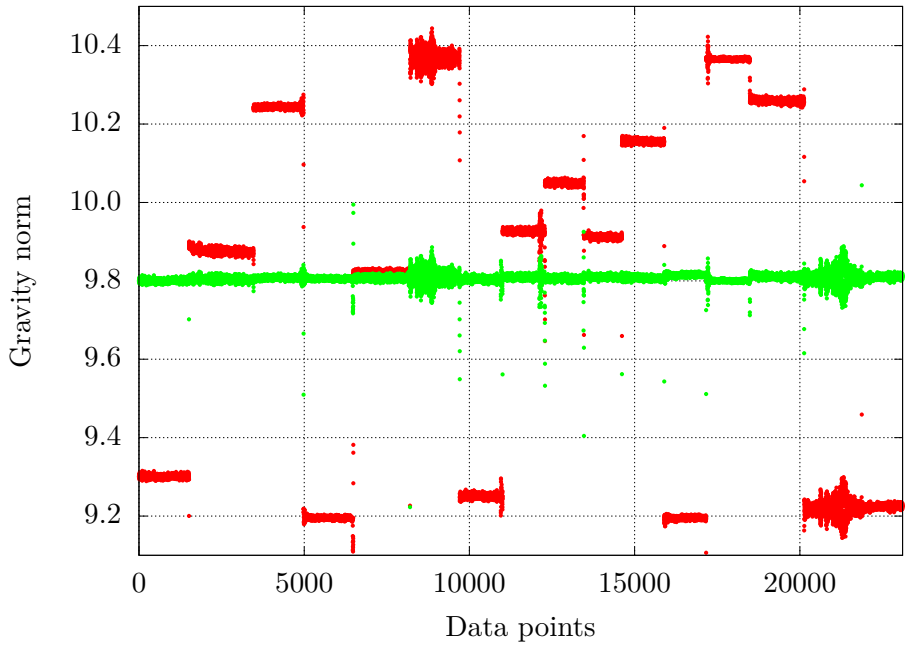

Figure 5.1: The plotted graph shows the results of the calibration of the accelerometer. The red points represent the measured gravity in $\frac{\mathrm{m}}{\mathrm{s}^2}$ before and the green points the measured gravity after the calibration. During the recording of the data the IMU was brought into different orientations. The data which was recorded during the orientation change was cut out for the sake of clarity. The results shown correspond to the procedure on the left side of Fig. 3.1.

In this case, the following calibration parameters are obtained:

$$\boldsymbol{K}^a = \begin{bmatrix} 0.9948850748 & 0 & 0 \\ 0 & 0.9948459340 & 0 \\ 0 & 0 & 1.0026721030 \end{bmatrix}$$

$$\boldsymbol{T}^a = \begin{bmatrix} 1 & -0.0061402733 & 0.0040833257 \\ 0 & 1 & 0.0026169734 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{b}^a = \begin{bmatrix} -0.1490165237 \\ -0.0525612944 \\ -0.5891789281 \end{bmatrix}$$

5.2 Evaluation of Gyroscope Calibration

In this section, the results of calibration of the gyroscope are described. The underlying procedure is shown in Fig. 3.1 on the right side.

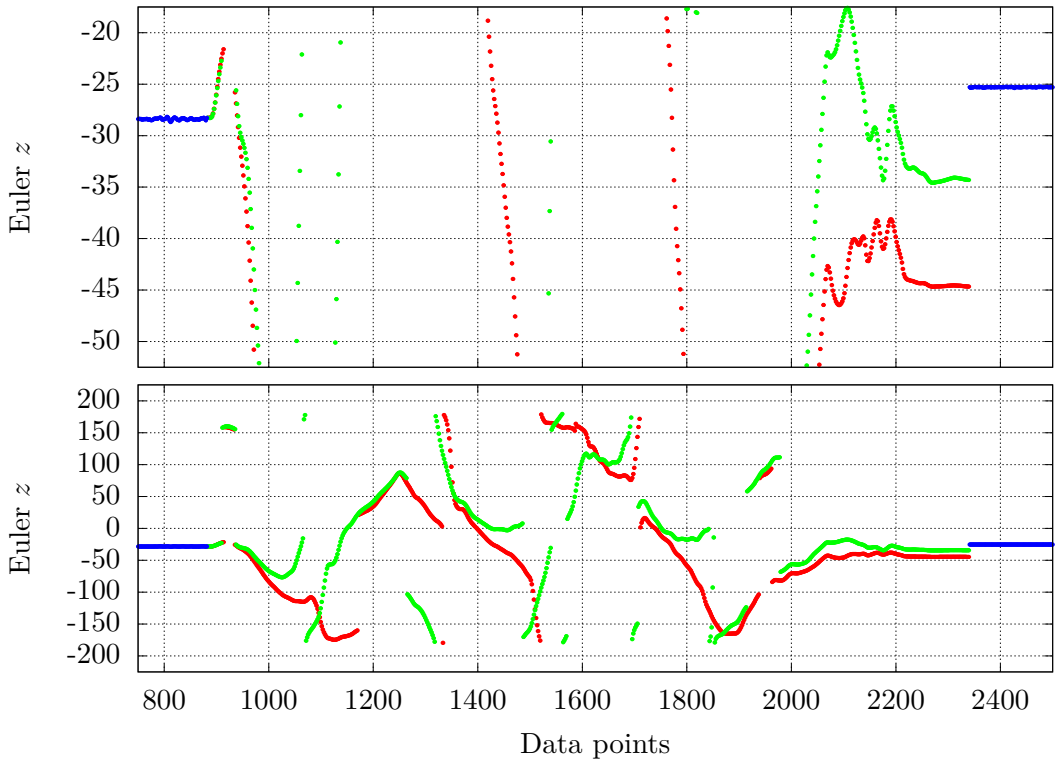

Figure 5.2: The plotted graphs show a section of the data set which is already shown in Fig. 5.1. Here, the orientation with respect to the z-axis of the IMU, represented as Euler angle in °, is shown. For the sake of clarity, only one orientation change between two successive standstills is shown here. The blue dots represent the orientation determined by the accelerometer, the red and green dots represent the orientation determined by the gyroscopic measurements. The green dots represent the results after calibration. In the ideal case, the orientation after the completion of the movement (approx. data point 2350) should correspond to that of the accelerometer. Both graphs show the same data in different detail.

In this section, the same dataset as in the previous section was used for calibration. Here, the orientation was changed between successive standstills. During the standstills, the exact direction of the gravity vector is determined with the aid of the accelerometer, and the orientation of the IMU is determined by this. During a movement, the orientation is determined by the temporal integration of the angular velocities, which were recorded by the gyroscope. Optimally, the orientation determined by the gyroscope at the end of the motion should correspond to that of the accelerometer.

Considering Fig 5.2, the orientation at the end of the movement (approx. data point 2350) in the uncalibrated case (red points) still has a deviation of approx. 20 ° to the actual orientation (blue points). By calibrating the gyroscope, this deviation is halved to approx. 10 °. The reason for the remaining deviation is not known. Here it is possible that there is still a remaining error due to the numeric integration. In the diagram it is also noticeable that in the calibrated case a different orientation is determined than in the uncalibrated case. It is assumed that by the calibration the values of the orientation lie closer to the true value. However, since no true

values for the orientation are known (apart from the standstill phases), this does not allow a conclusive assessment.

In this case, the following calibration parameters are obtained:

$$\boldsymbol{K}^{\omega} = \begin{bmatrix} 0.8427205966 & 0 & 0 \\ 0 & 0.8965406691 & 0 \\ 0 & 0 & 0.9341525635 \end{bmatrix}$$

$$\boldsymbol{T}^{\omega} = \begin{bmatrix} 1 & 0.4618838528 & 0.3827648811 \\ -0.4840555917 & 1 & 0.0760070500 \\ -0.3428641795 & -0.2175452987 & 1 \end{bmatrix}$$

$$\boldsymbol{b}^{\omega} = \begin{bmatrix} -0.006327691203 \\ 0.0010545279 \\ -0.0018737855 \end{bmatrix}$$

5.3 Summary

In the previous two sections, the results of calibrating an accelerometer, respectively a gyroscope, were explained. Especially the measured values of the accelerometer are improved by the calibration. In order to use an accelerometer for the determination of translational motions, it is necessary to have acceleration measurement values that are as exact as possible. A large measurement error has a large negative influence on the position estimate due to the double integration.

The calibration of the gyroscope also led to an improvement in the measured values. Here, also, the position estimation is dependent on error-free measurement values, because the gyroscope is used during the execution of a movement to determine the exact orientation. A strongly erroneous orientation estimation leads to an error in the position estimation, because the knowledge about the exact orientation of the gravity vector is necessary to adjust the measured values of the accelerometer around this.

Chapter 6

Evaluation of Camera-IMU Extrinsic Calibration

The purpose of this chapter is to show how the system described in Chapter 4 performs. Thereby, the results of the individual steps are presented and evaluated. In the following sections, the methods of evaluation are discussed and the data sets used are presented.

6.1 Methodology

As explained in Section 1.1, the overall purpose of this work is to improve the pose estimation using VIO methods with respect to the *UWSensor* project. More precisely, to improve the initial estimation of the visual scale of the *VINS-Mono* software. The quality of the estimated scale is significantly influenced by the following estimates:

- Extrinsic rotation between camera and IMU frame $\hat{\boldsymbol{q}}_b^c$
- Bias of gyroscope $\hat{\boldsymbol{b}}_\omega$
- Gravity $\hat{\boldsymbol{g}}^{c_0}$
- Translation between camera and IMU frame $\hat{\boldsymbol{p}}_b^c$
- Bias of accelerometer $\hat{\boldsymbol{b}}_a$

Since the quality of these estimates has a direct impact on the estimation of the scale, the following sections focus on the results of these estimates.

In the course of this, the question arises how to determine the quality of the respective estimates. The main problem is that for arbitrary systems, consisting of camera and IMU, the real values $\boldsymbol{q}_b^c$, $\boldsymbol{b}_\omega$, $\boldsymbol{g}^{c_0}$, $\boldsymbol{p}_b^c$ and $\boldsymbol{b}_a$ are not known and cannot be determined without any uncertainties. For example, if the translation between an IMU and a camera, both fixed to a drone, should be estimated, this could be compared to the hand-measured translation. In this case, however, it is impossible to achieve millimeter accuracy or better, since it is usually difficult to determine,

for example, the exact origin of the sensor axes inside the IMU case. Besides measuring by hand, methods are employed which use certain patterns like AprilTags for support like e.g. the work [28]. However, in this case the accuracy of the translation estimate also depends on the camera calibration. In addition, badly estimated biases of gyroscope and accelerometer can distort the result.

Therefore, the only possible option is to compare the accuracy of the estimates of $\hat{\boldsymbol{q}}_b^c$, $\hat{\boldsymbol{b}}_\omega$, $\hat{\boldsymbol{g}}^{c_0}$, $\hat{\boldsymbol{p}}_b^c$ and $\hat{\boldsymbol{b}}_a$ with the resulting pose estimation, in this case of *VINS-Mono*. It can be assumed that better estimated initial values lead to a higher accuracy in the pose estimation.

Nevertheless, in order to compare the accuracy of different methods of estimating the parameters, Eqs. (4.36), (4.45), (4.67) and (4.82) are used. It is assumed here that the estimation is optimal when the error becomes minimal. The disadvantage is that the mentioned errors of the estimations become larger, if the underlying software, here *VINS-Mono*, itself produces errors in the determination of the camera poses. The inaccuracy of the camera poses are not assessable within the scope of this work. Consequently, if two different data sets of a flight of the same drone are given, it is not possible to evaluate which dataset is better suited for the estimation of the parameters, because it is not possible to evaluate which dataset leads to better camera pose estimation using *VINS-Mono*. At this point, however, the assumption is made that the lower the error of the estimates, the more the data set is suitable for calibration. A proper comparison of multiple datasets can only be made using the final resulting pose estimation.

6.2 Choice of Minimization Algorithm

As already explained in Section 2.7, Eigen's LM algorithm is used to find a solution of the Eqs. (4.35), (4.44), (4.66) and (4.81). Since it turned out that in the iterative estimation of scale and accelerometer bias, the estimates sometimes do not converge but alternate, the Ceres Solver was additionally implemented. This behavior is not observed there. A more detailed description of this characteristic follows later.

6.3 Choice of Datasets

To test the performance of the procedure developed in this work, a publicly available dataset and one from the *UWSensor* project are used.

EuRoC MAV Dataset In the course of the development of the software, primarily the online [1] available dataset *MH_02_easy.bag* was used. It was collected in an industrial environment and contains millimeter accurate position ground truth from a laser tracking system. For a detailed description of the equipment used to generate the data, please refer to Article [16]. This dataset has the advantage that the system is manually slewed before flying through the hall, recording data with rotations necessary for calibration. Fig. 6.1 shows one of the images captured in this dataset. The IMU used is identified as *ADIS16448* and the camera with a resolution of 854×480 is identified as *Aptina MT9V034*.

[1] `https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets`

Figure 6.1: Snapshot of dataset *MH_02_easy.bag* during the drone flight through an industrial hall.

***UWSensor* Dataset** Since improving the position estimation of the sensors of the *UWSensor* project is part of the purpose of this work, a data set taken from this project will be used for calibration in the following. Fig. 6.2 shows one of the images captured in this dataset. The IMU used is identified as *KVH 1750* and the camera with a resolution of 3208×2200 is identified as *SONY IMX420*. Here the resolution is scaled to 1600×1100.

As explained in the previous section, the results of the estimated parameters are highly dependent on the camera poses estimated by *VINS-Mono*. The following parameters, among others, can be adjusted within *VINS-Mono* to improve the estimation of camera poses:

max_cnt Defines the maximum number of tracked features.

min_dist Defines the minimum distance between two features.

freq Defines the frequency at which the camera images are evaluated.

WINDOW_SIZE Defines the maximum number of keyframes for which the camera poses are estimated.

The parameter *WINDOW_SIZE* can be adjusted in file *vins_estimator/src/parameters.h*, all others in the respective configuration file in the folder *config/*. The estimation of the camera poses can also be influenced by the provided camera images, which is done in the course of this

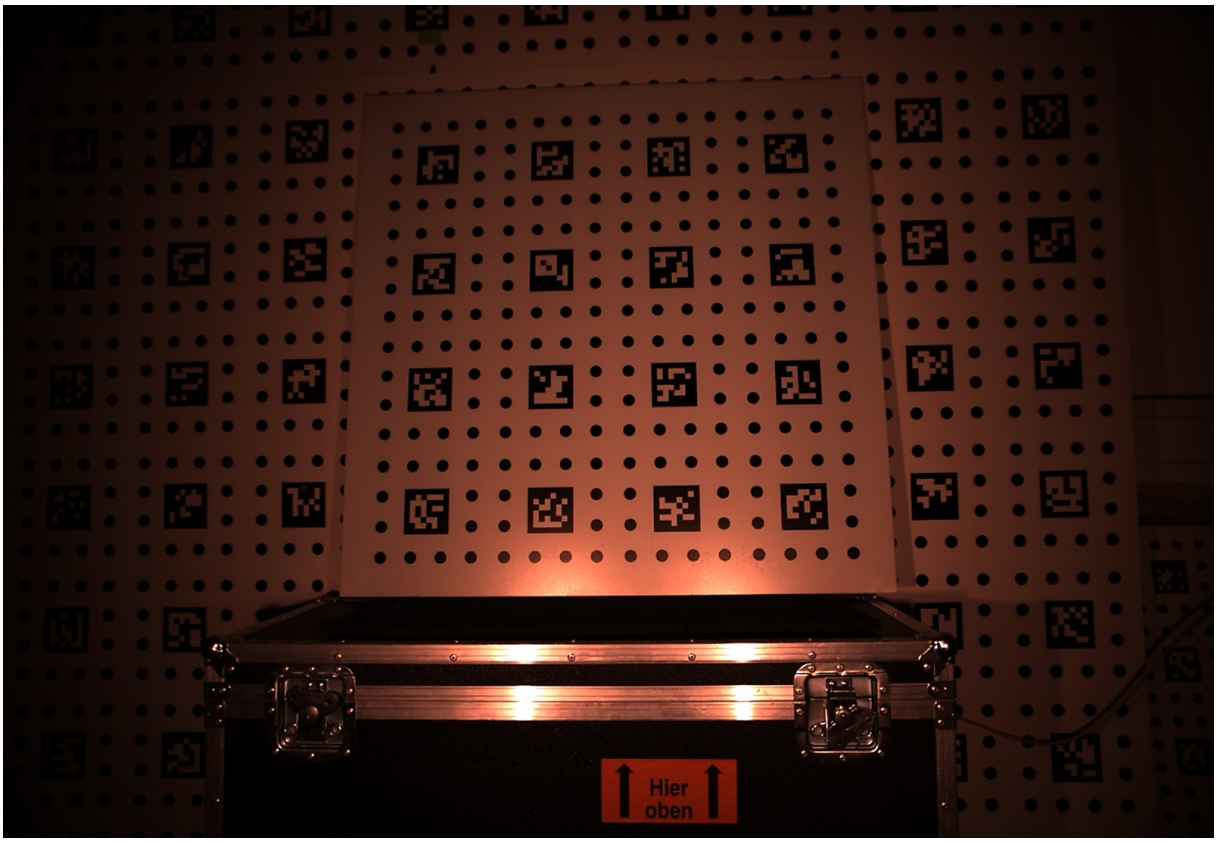

Figure 6.2: Snapshot of a dataset from *UWSensor* project during a calibration process.

work by setting a certain start time when replaying the datasets. In the course of this evaluation, it was found that the results are best when the aforementioned parameters are set to the values shown in Table 6.1.

In this evaluation, the two datasets with different values for *WINDOW_SIZE* are examined. Table 6.2 shows the values used for *WINDOW_SIZE*, an *ID* which is used in the following for the identification of the respective configuration, the resulting error of the estimation of the extrinsic rotation and the error of the estimation of the scales. Here, the resulting errors refer to the average of the errors determined with Eq. (4.36) and Eq. (4.67) respectively. The two configurations shown here in green lead to "good" results in the estimation of the calibration parameters, whereas the results of the configurations shown in red lead to almost unusable results, which is also seen from the significantly larger errors. Both "bad" configurations are included in the evaluation to show the immense influence they have on the quality of the estimation. Two reasons why the estimations with a *WINDOW_SIZE* of 15 lead to "bad" results, is on the one hand an incorrect position estimation by *VINS-Mono*, on the other hand too few rotational and translational movements in the used data.

Parameter	EuRoC MAV dataset	*UWSensor* dataset
max_cnt	300	300
min_dist	15	30
freq	0	5
start time	5s	16s

Table 6.1: Configuration of *VINS-Mono*, which was used for the evaluation of the respective dataset. For *UWSensor*, a larger *min_dist* is chosen because the resolution of the camera used is larger, ensuring that features are evenly distributed across the image.

ID	WINDOW_SIZE	Error rotation	Error scale
EuRoC MAV dataset			
1	30	$1.324E-4$	$6.211E-6$
2	15	$4.712E-4$	$1.814E-5$
***UWSensor* dataset**			
3	30	$2.667E-4$	$1.713E-4$
4	15	$1.007E-3$	$1.051E-3$

Table 6.2: Configurations used in the course of the evaluation. The color red denotes failed estimates and green denotes successful estimates. The errors which are listed here correspond to the final result of the software developed within this thesis, therefore all steps in the Fig. 4.1, resp. 4.2 were executed, using the LM algorithm.

6.4 Evaluation of Gyroscope Bias and Extrinsic Rotation Estimation

In this section, the results of the individual steps of the estimation of extrinsic rotation between camera and IMU frame und gyroscope bias are described.

In general, the purpose of the following sections is to demonstrate that the estimates are refined by applying the two proposed methods: the weighting of the individual datasets and the alternating estimation of extrinsic rotation and gyroscope bias. In each case, for extrinsic rotation and gyroscope bias, first the results are presented, which are obtained without the use of the two methods mentioned and with the use of weighting. Finally, the results obtained by alternating the estimation of the two parameters are shown.

6.4.1 Extrinsic Rotation Estimation

The software developed in the course of this work starts with the estimation of the extrinsic rotation, shown in Fig. 4.1 on the left. Results to be expected in the ideal case are listed in table 6.3, along with the results obtained by the native calculation of *VINS-Mono*. As already explained, the real values for the extrinsic rotation are not known, the values listed in the table

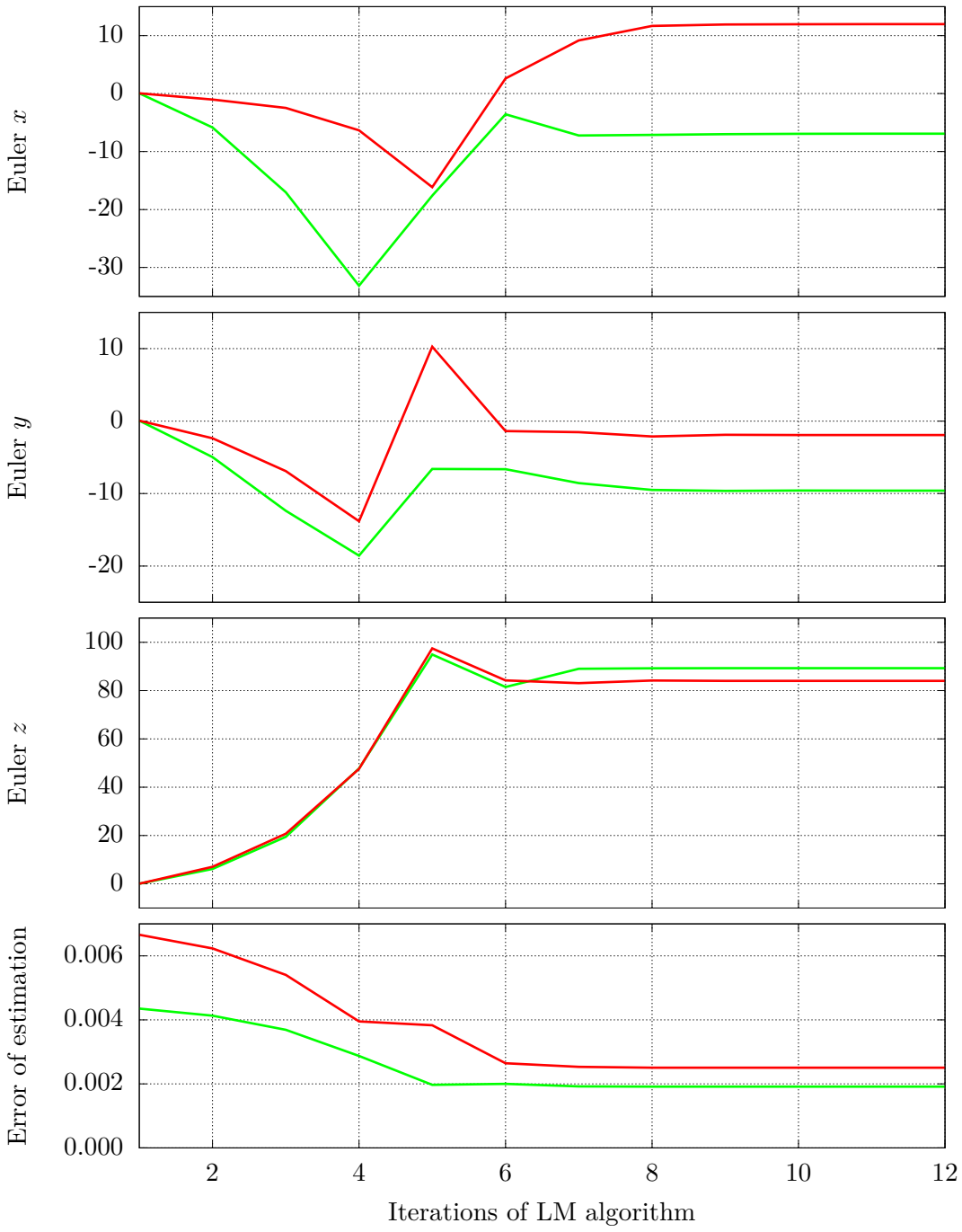

Figure 6.3: The plotted graphs show the result of the estimation of the extrinsic rotation between camera and IMU, represented in Euler angles in °, for each iteration of the LM algorithm. Data sets ID 1 (green) and ID 2 (red) were used. The results shown here correspond to the green block on the left side of the Fig. 4.1.

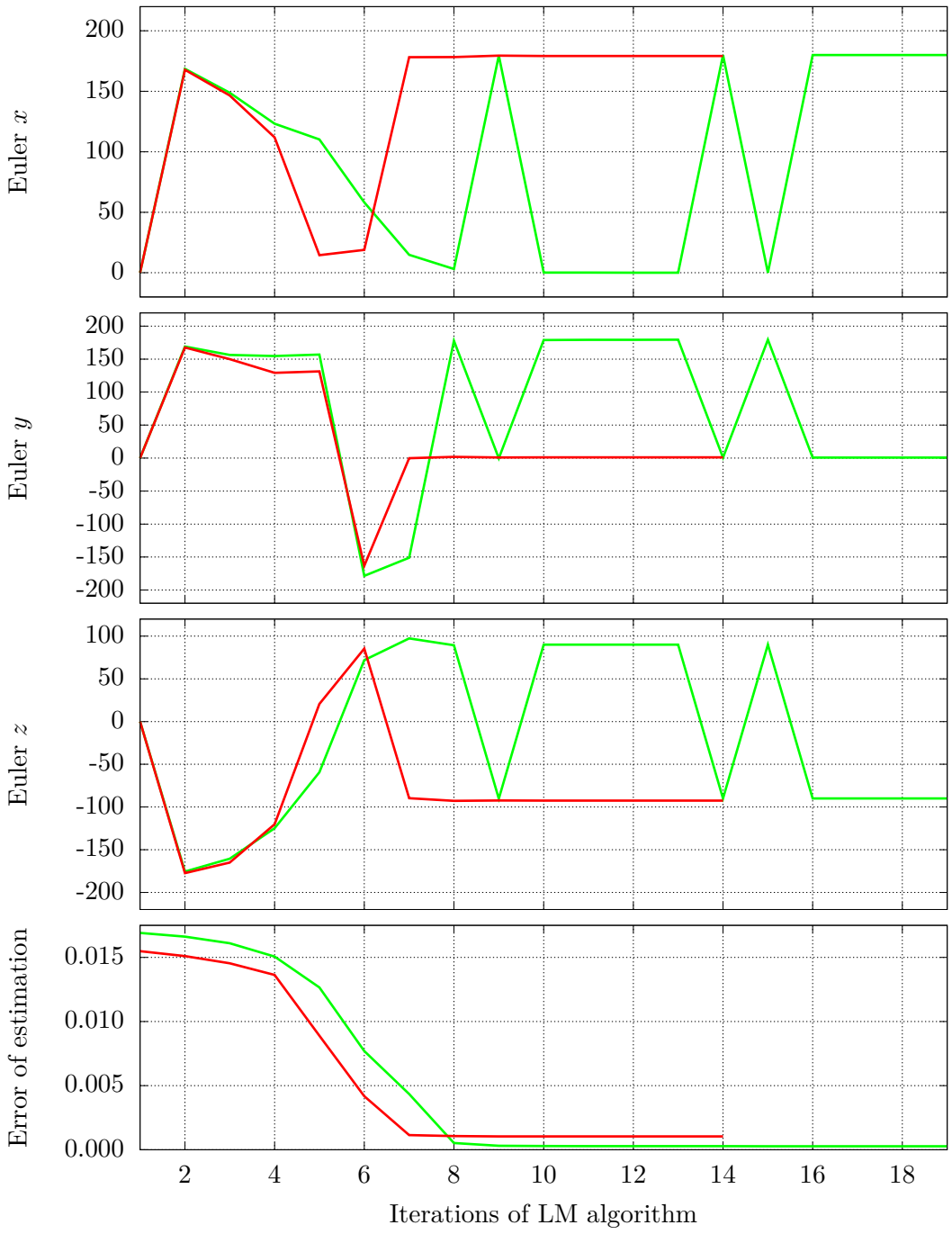

Figure 6.4: The plotted graphs show the result of the estimation of the extrinsic rotation between camera and IMU, represented in Euler angles in °, for each iteration of the LM algorithm. Data sets ID 3 (green) and ID 4 (red) were used. The results shown here correspond to the green block on the left side of the Fig. 4.1.

Source	Euler x	Euler y	Euler z
EuRoC MAV dataset			
VINS-Mono	50.767°	1.829°	98.952°
Ideal result	0.000°	0.000°	90.000°
***UWSensor* dataset**			
VINS-Mono	178.172°	3.877°	−90.004°
Ideal result	180.000°	0.000°	−90.000°

Table 6.3: Results for the extrinsic rotation in Euler angles expected according to the developers of the respective sensor systems along with the results of the native calculation by *VINS-Mono*.

ID	Source	Euler x	Euler y	Euler z	Error
1	LM	−6.9205°	−9.5942°	89.2621°	$1.914E{-}3$
1	Ceres	−5.6991°	−8.9992°	89.2882°	$1.933E{-}3$
2	LM	11.9759°	−1.9178°	84.0069°	$2.506E{-}3$
2	Ceres	10.4164°	−3.7315°	84.4806°	$2.513E{-}3$
3	LM	179.9997°	0.6624°	−89.9629°	$2.762E{-}4$
3	Ceres	179.9908°	0.6538°	−89.9359°	$3.175E{-}4$
4	LM	179.2263°	1.0211°	−92.5425°	$1.040E{-}3$
4	Ceres	169.8912°	0.2874°	−97.2411°	$3.210E{-}3$

Table 6.4: Results of the estimation of the extrinsic rotation between camera and IMU, represented in Euler angles, using the LM algorithm and Ceres Solver. The results shown here correspond to the green block on the left side of Fig. 4.1.

are provided by the developers of the respective sensor systems based on the CAD models and are highly likely to be subject to errors themselves, but nevertheless serve as a reference value here. The values for rotation are given in this section in terms of Euler angles, since these provide a simpler representation compared to the quaternions and rotation matrices used in the algorithms.

First, the results are to be analyzed, in case the weighting is omitted, therefore only the result of the green block on the left side of Fig. 4.1. The results of the iterative process of the LM algorithm are shown in Fig. 6.3 for ID 1 and 2 and in Fig. 6.4 for ID 3 and 4 together with the resulting error of the estimation. Both diagrams have the purpose to give the reader an impression of how the LM algorithm iteratively finds a solution and how the error of the estimation behaves during this process. Final results after termination of the LM algorithm are listed in Table 6.4, along with the results computed by the Ceres Solver.

Considering the results using the EuRoC MAV dataset, it is recognizable that the results of LM and Ceres are much closer to the ideal result than the native result of *VINS-Mono*. The

ID	Best K_0	Euler x	Euler y	Euler z	Error
1	374	$-9.6517°$	$-13.9831°$	$89.5419°$	$1.898E-3$
1	0	$-6.9205°$	$-9.5942°$	$89.2621°$	$1.914E-3$
2	240	$14.6263°$	$-3.3292°$	$82.3015°$	$2.487E-3$
2	0	$11.9759°$	$-1.9178°$	$84.0069°$	$2.506E-3$
3	2112	$179.9438°$	$0.5096°$	$-89.8824°$	$2.759E-4$
3	0	$179.9997°$	$0.6624°$	$-89.9629°$	$2.762E-4$
4	744	$179.4550°$	$0.6554°$	$-91.7345°$	$1.031E-3$
4	0	$179.2263°$	$1.0211°$	$-92.5425°$	$1.040E-3$

Table 6.5: Results of the estimation of the extrinsic rotation between camera and IMU, represented in Euler angles, using the LM algorithm and the weighting strategy. The results shown here correspond to all blocks on the left side of Fig. 4.1. Note that the results for $K_0 = 0$ are taken from Table 6.4 and serve as a comparison if the weighting is not used.

latter deviates strongly from the true value, making it impossible to estimate the visual scale and anything related to it. The results using the *UWSensor* dataset come close to the ideal results, also the results of *VINS-Mono* match better with this dataset, although not as good as the LM algorithm and the Ceres Solver.

Regarding the results of both data sets it is noticeable that in all cases the error of the estimation of the LM algorithm is smaller than that of the Ceres Solver, a reason for this is not known. In addition, the results of the *UWSensor* dataset are generally better, this could be due to the fact that in this sensor setup a much higher quality IMU is used and the expected bias of the gyroscope is much smaller than the bias of the gyroscope used in the EuRoC MAV dataset. At this point it should be pointed out again that the bias of the gyroscope was not taken into account in the estimation of the extrinsic rotation in this section.

Looking at the errors of the estimation of the extrinsic rotation at ID 1 (see table 6.2 and 6.4), it clearly shows that the error ($1.914E-3$) in this step of the software is significantly larger than the error ($1.324E-4$) which is present after the termination of the software. Thus it can be noted, how important the methods presented in the following sections are, i.e. the introduction of a weighting and the alternating estimation of rotation and bias.

Evaluation of the Weighting Scheme

At this point, weighting is introduced in the estimation of extrinsic rotation. Considering again Fig. 4.1, the results are presented which are obtained when the whole process on the left side is executed (red and green blocks). The software now estimates the parameters for different factors K_0 and selects the result for which the error becomes minimal. Therefore, for each K_0, the process explained in the previous section is performed until the result does not change anymore. The results of this are shown in Fig. 6.5 for ID 1 and ID 2 and in Fig. 6.6 for ID 3 and ID 4 together with the resulting error of the estimate. Both diagrams have the purpose to give

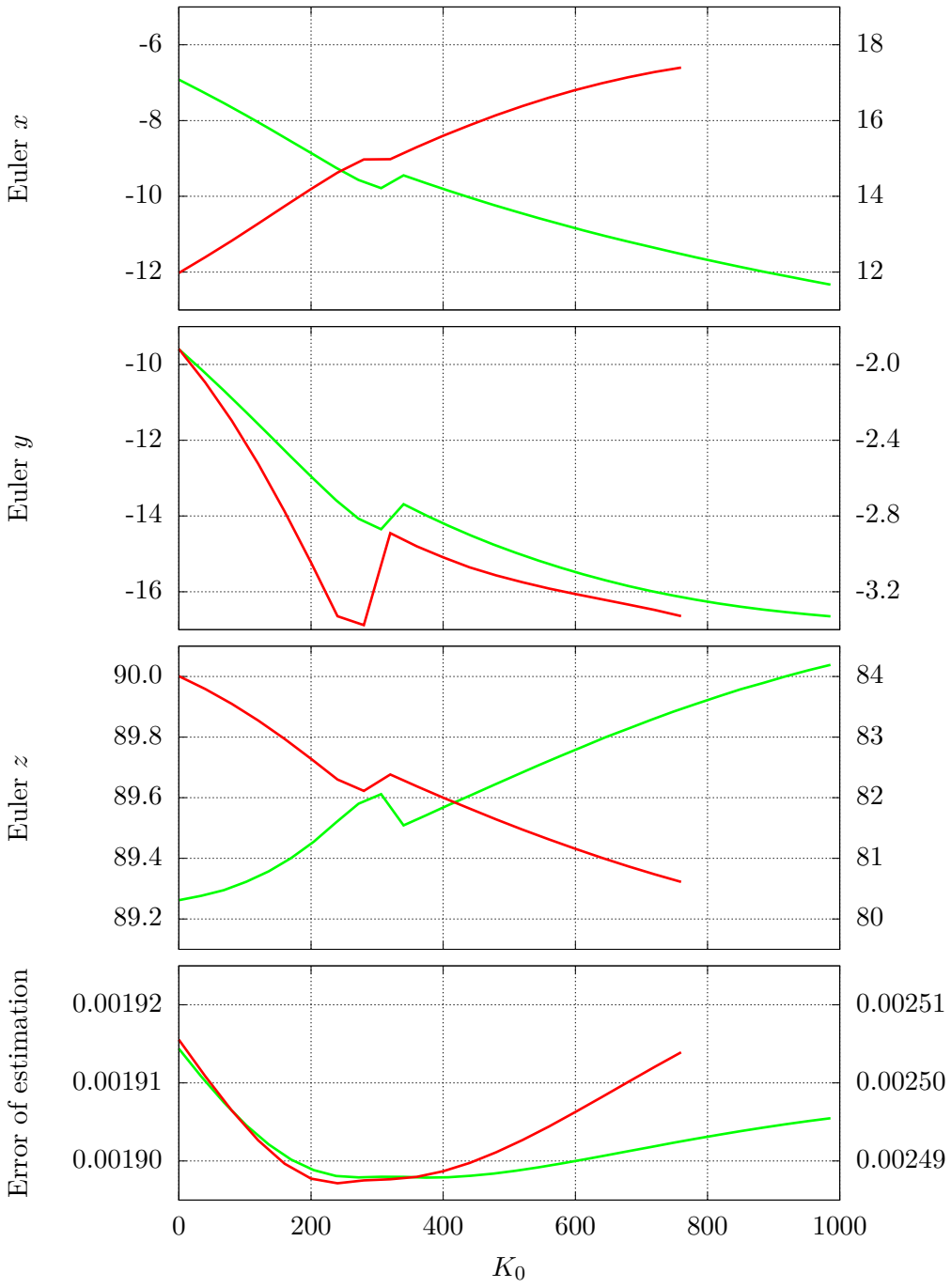

Figure 6.5: The plotted graphs show the result of the estimation of the extrinsic rotation between camera and IMU using the weighting strategy, represented in Euler angles in °, for each K_0. Data sets ID 1 (green) and ID 2 (red) were used, where the left y-axes belong to ID 1 and the right ones to ID 2. The results shown here correspond to all blocks on the left side side of the Fig. 4.1.

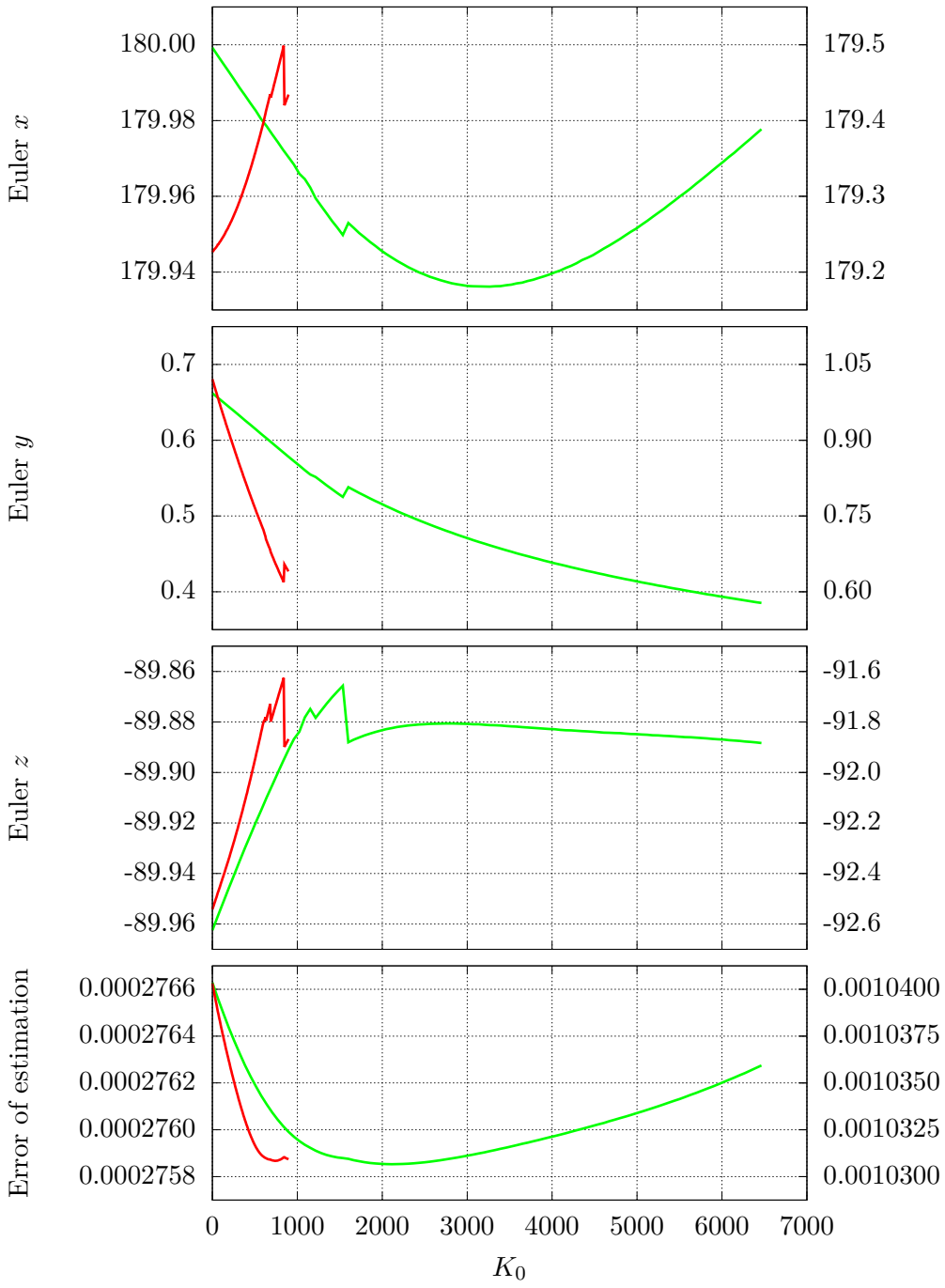

Figure 6.6: The plotted graphs show the result of the estimation of the extrinsic rotation between camera and IMU using the weighting strategy, represented in Euler angles in °, for each K_0. Data sets ID 3 (green) and ID 4 (red) were used, where the left y-axes belong to ID 3 and the right ones to ID 4. The results shown here correspond to all blocks on the left side side of the Fig. 4.1.

Source	Bias x	Bias y	Bias z
EuRoC MAV dataset			
VINS-Mono	0.0441	0.0234	0.0953
***UWSensor* dataset**			
VINS-Mono	$6.218E{-}3$	$1.547E{-}3$	$1.691E{-}3$

Table 6.6: Results of the gyroscope bias in $\frac{\text{rad}}{\text{s}}$ calculation by the native *VINS-Mono*. Note here that there are no known ideal results regarding the gyroscope bias and that here the values are likely to be error-prone, since the extrinsic rotations necessary for the bias estimate (cf. Table 6.3) are already corrupted.

ID	Source	Bias x	Bias y	Bias z	Error
1	LM	$-1.331E{-}3$	0.0232	0.0682	$6.863E{-}4$
1	Ceres	$-7.096E{-}4$	0.0223	0.07228	$4.660E{-}4$
2	LM	-0.0122	0.0592	0.0658	$1.214E{-}3$
2	Ceres	$-8.778E{-}3$	0.0614	0.0707	$9.525E{-}4$
3	LM	$-6.332E{-}5$	$6.538E{-}4$	$2.096E{-}4$	$2.679E{-}4$
3	Ceres	$-1.491E{-}4$	$1.121E{-}3$	$2.112E{-}4$	$3.113E{-}4$
4	LM	$1.212E{-}3$	$8.522E{-}3$	$5.919E{-}4$	$1.234E{-}3$
4	Ceres	$4.561E{-}4$	0.0213	$-3.931E{-}3$	$3.834E{-}3$

Table 6.7: Results of the estimation of the gyroscope bias in $\frac{\text{rad}}{\text{s}}$ using the LM algorithm and Ceres Solver. The results shown here correspond to the green block on the right side of Fig. 4.1.

the reader an impression how the software determines the optimal factor K_0. The results with minimum error (best K_0) are listed in Table 6.5. In addition, the results (without weighting) of the previous section are also listed there to enable a comparison. Note that the weighting is only implemented for the LM algorithm, as this gives better results compared to the Ceres Solver based on the error of the estimation.

Looking at Table 6.5, it is noticeable that the weighting produces a smaller error for each data set. Especially for ID 4 it can be seen that the results using the weighting are much closer to the ideal results. Considering the results of ID 1 and ID 2, it is noticeable that they deviate somewhat more from the ideal results due to the weighting, compared to the values without weighting. Mathematically, however, the results with weighting are still better, since they have a smaller error. At this point it should be pointed out again that so far no gyroscope bias has been included in the calculation.

6.4.2 Gyroscope Bias Estimation

Once the extrinsic rotation has been determined, the bias of the gyroscope is estimated, shown in Fig. 4.1 on the right. Compared to extrinsic rotation, no ideal values are known for the

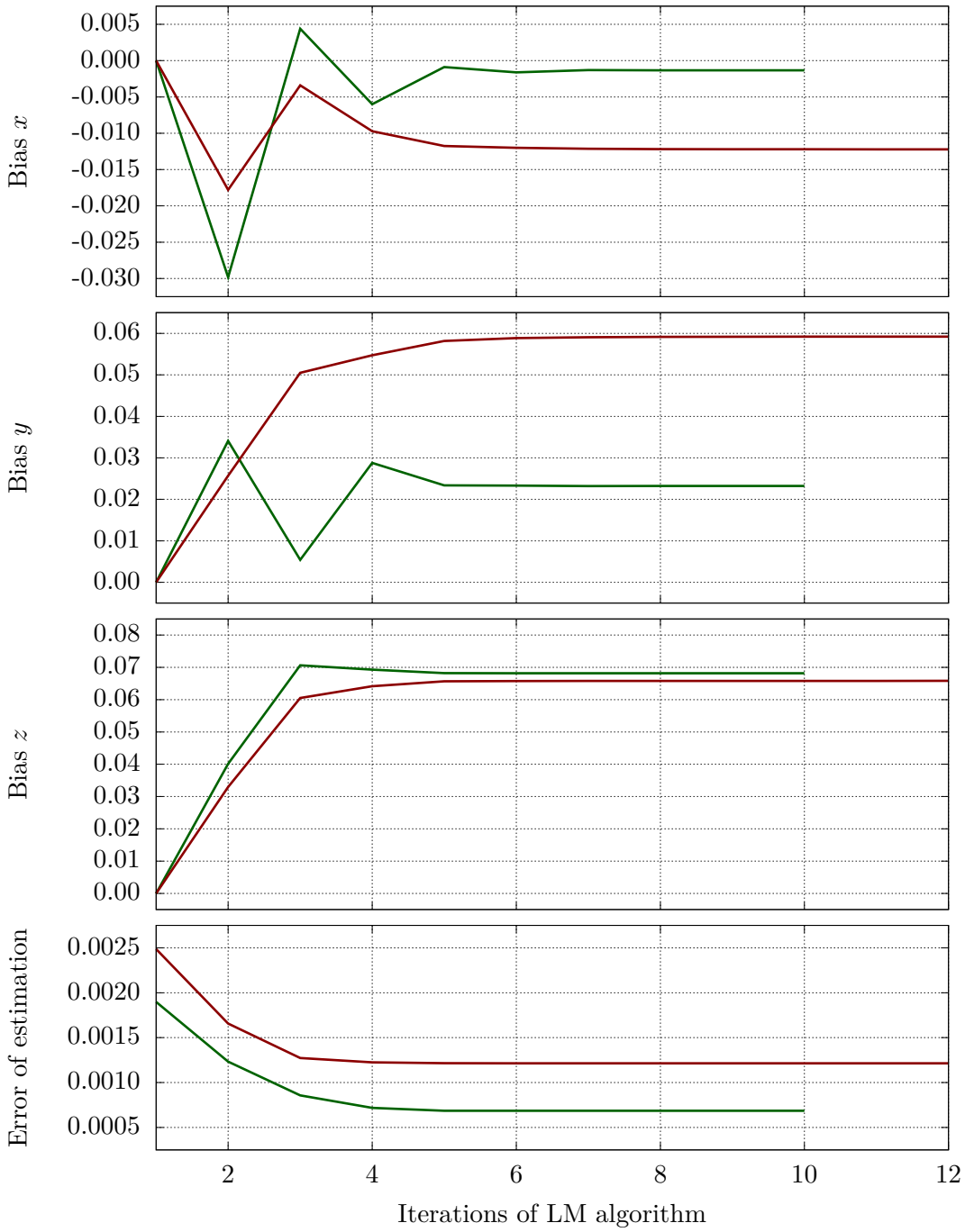

Figure 6.7: The plotted graphs show the result of the estimation of the gyroscope bias in $\frac{\text{rad}}{\text{s}}$, for each iteration of the LM algorithm. Data sets ID 1 (green) and ID 2 (red) were used. The results shown here correspond to the green block on the right side of the Fig. 4.1.

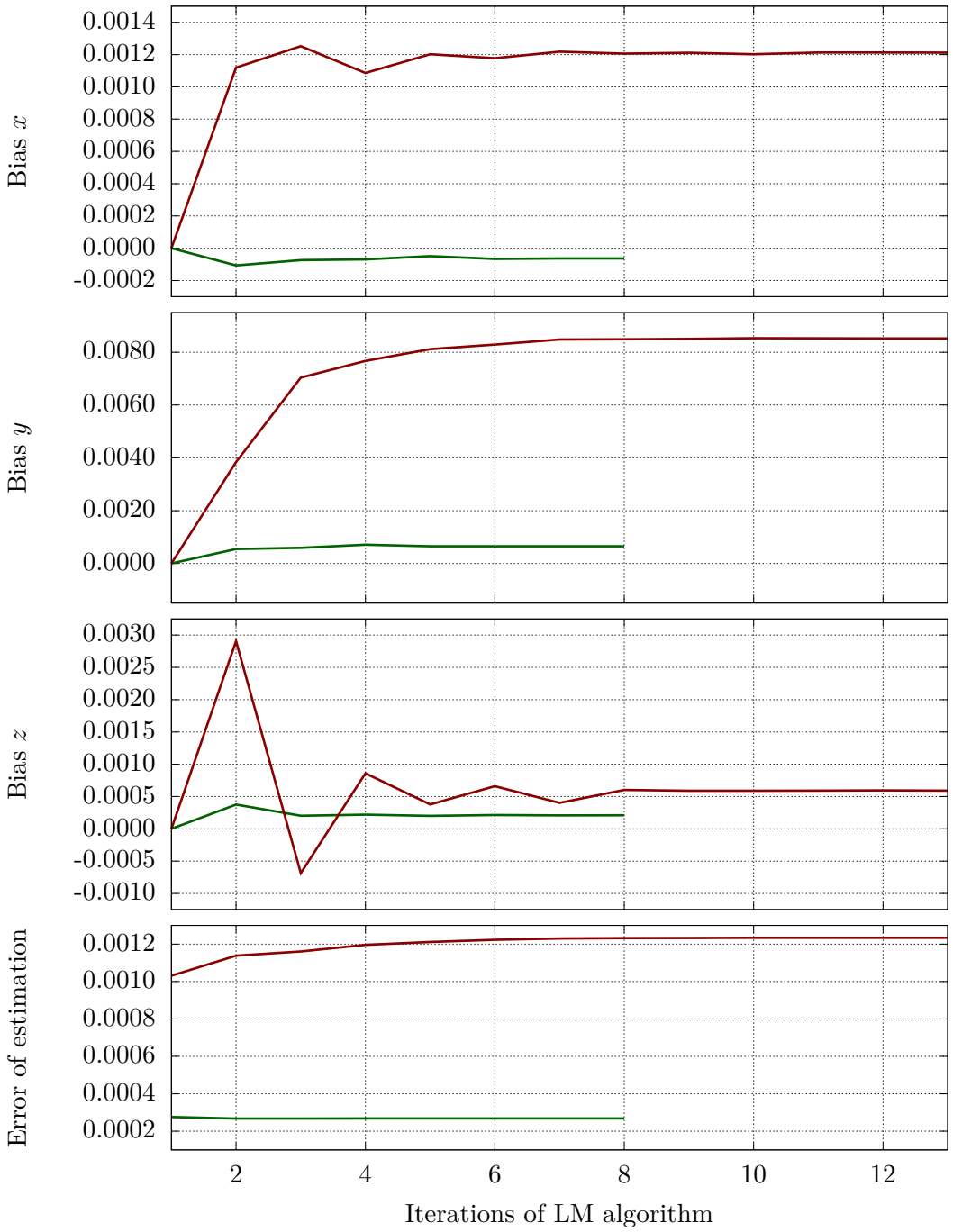

Figure 6.8: The plotted graphs show the result of the estimation of the gyroscope bias in $\frac{\text{rad}}{\text{s}}$, for each iteration of the LM algorithm. Data sets ID 3 (green) and ID 4 (red) were used. The results shown here correspond to the green block on the right side of the Fig. 4.1.

ID	Best K_1	Bias x	Bias y	Bias z	Error
1	1736	$-1.028E-3$	0.0232	0.0713	$6.830E-4$
1	0	$-1.331E-3$	0.0232	0.0682	$6.863E-4$
2	399	-0.0154	0.0627	0.0636	$1.210E-3$
2	0	-0.0122	0.0592	0.0658	$1.214E-3$
3	5561	$-2.600E-4$	$5.765E-4$	$3.347E-4$	$2.668E-4$
3	0	$-6.332E-5$	$6.538E-4$	$2.096E-4$	$2.679E-4$
4	936	$2.872E-4$	$1.074E-3$	$-1.296E-4$	$1.009E-3$
4	0	$1.212E-3$	$8.522E-3$	$5.919E-4$	$1.234E-3$

Table 6.8: Results of the estimation of the gyroscope bias in $\frac{\text{rad}}{\text{s}}$ using the LM algorithm and the weighting strategy. The results shown here correspond to all blocks on the right side of Fig. 4.1. Note that the results for $K_1 = 0$ are taken from Table 6.7 and serve as a comparison if the weighting is not used.

bias of the gyroscope. Since the bias of a gyroscope can vary after each sensor start and is e.g. also temperature dependent, no quantitative statement about the results can be made for this section. However, since in the course of the software (cf. Fig. 4.1) the bias is used to improve the results of the extrinsic rotation, it is concluded whether the results for the bias are realistic. A strongly erroneously estimated bias leads to a degradation of the results of the rotation. In table 6.6 the results are listed, which the native computation of *VINS-Mono* supplies. On the one hand, the magnitudes of the results are plausible, since a smaller bias is expected for the *UWSensor* dataset, due to the significantly higher grade IMU. On the other hand, the estimated rotations between the IMUs and the cameras, which are necessary for the calculation of the bias, are error-prone, especially for the EuRoC MAV dataset (cf. Table 6.3), so that a large error in the estimation of the bias is to be expected.

First, the results are analyzed, in case the weighting is omitted, therefore only the result of the green block on the right side of Fig. 4.1. The results of the iterative process of the LM algorithm are shown in Fig. 6.7 for ID 1 and 2 and in Fig. 6.8 for ID 3 and 4 together with the resulting error of the estimation. Both diagrams have the purpose to give the reader an impression of how the LM algorithm iteratively finds a solution and how the error of the estimation behaves during this process. Final results after termination of the LM algorithm are listed in Table 6.7, along with the results computed by the Ceres Solver.

As already explained, no quantitative statement can be made here about the results. However, it is concluded that the results of the LM algorithm and the Ceres Solver are more realistic compared to the results of *VINS-Mono*, since the estimated rotations (cf. Table 6.5), used for bias estimation, are closer to the ideal results. Also the results of the following sections suggest that the results in Table 6.7 are realistic and do not suffer from a large error.

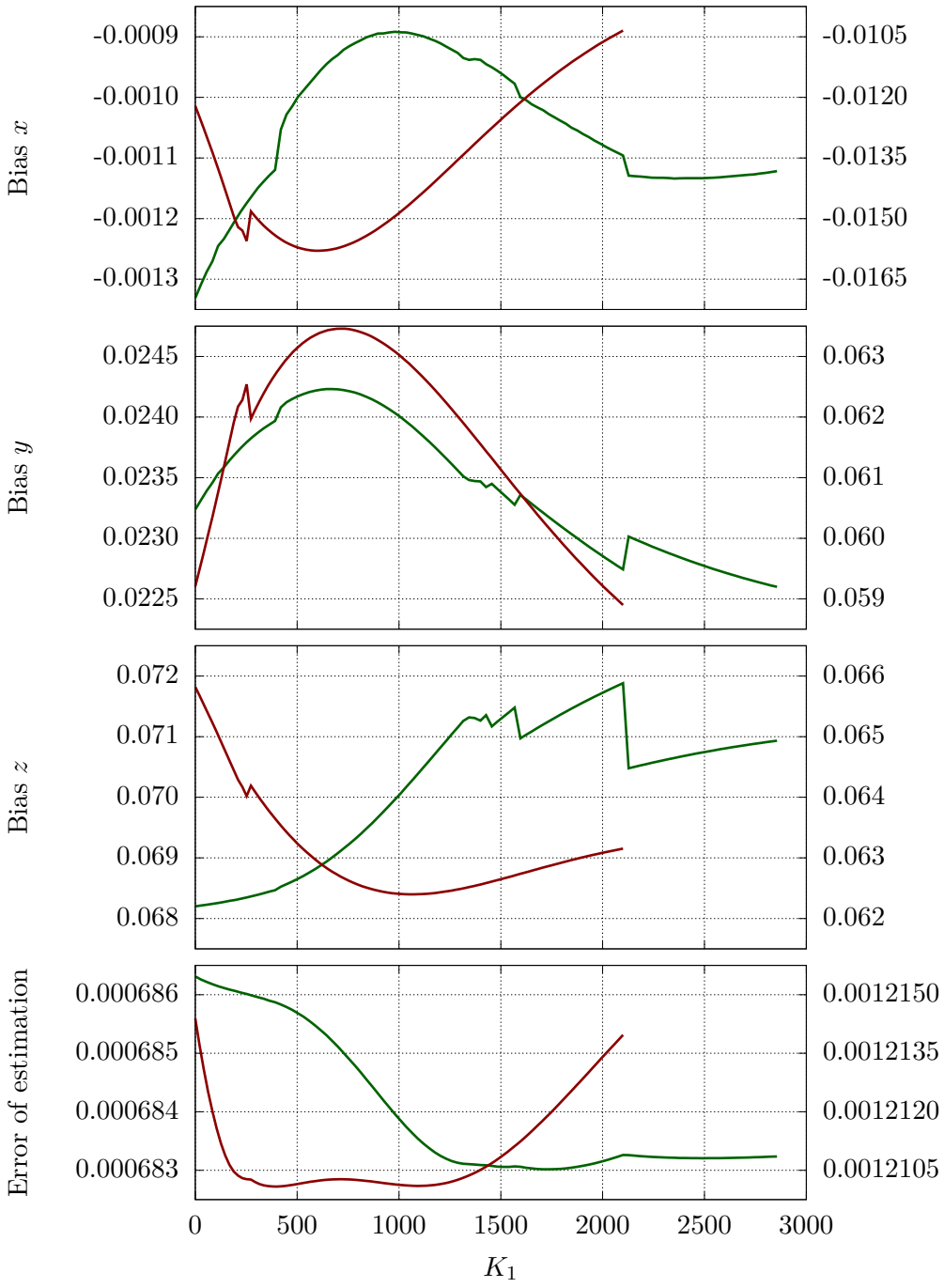

Figure 6.9: The plotted graphs show the result of the estimation of the gyroscope bias in $\frac{\text{rad}}{\text{s}}$ for each K_1 using the weighting strategy. Data sets ID 1 (green) and ID 2 (red) were used, where the left y-axes belong to ID 1 and the right ones to ID 2. The results shown here correspond to all blocks on the right side side of the Fig. 4.1.

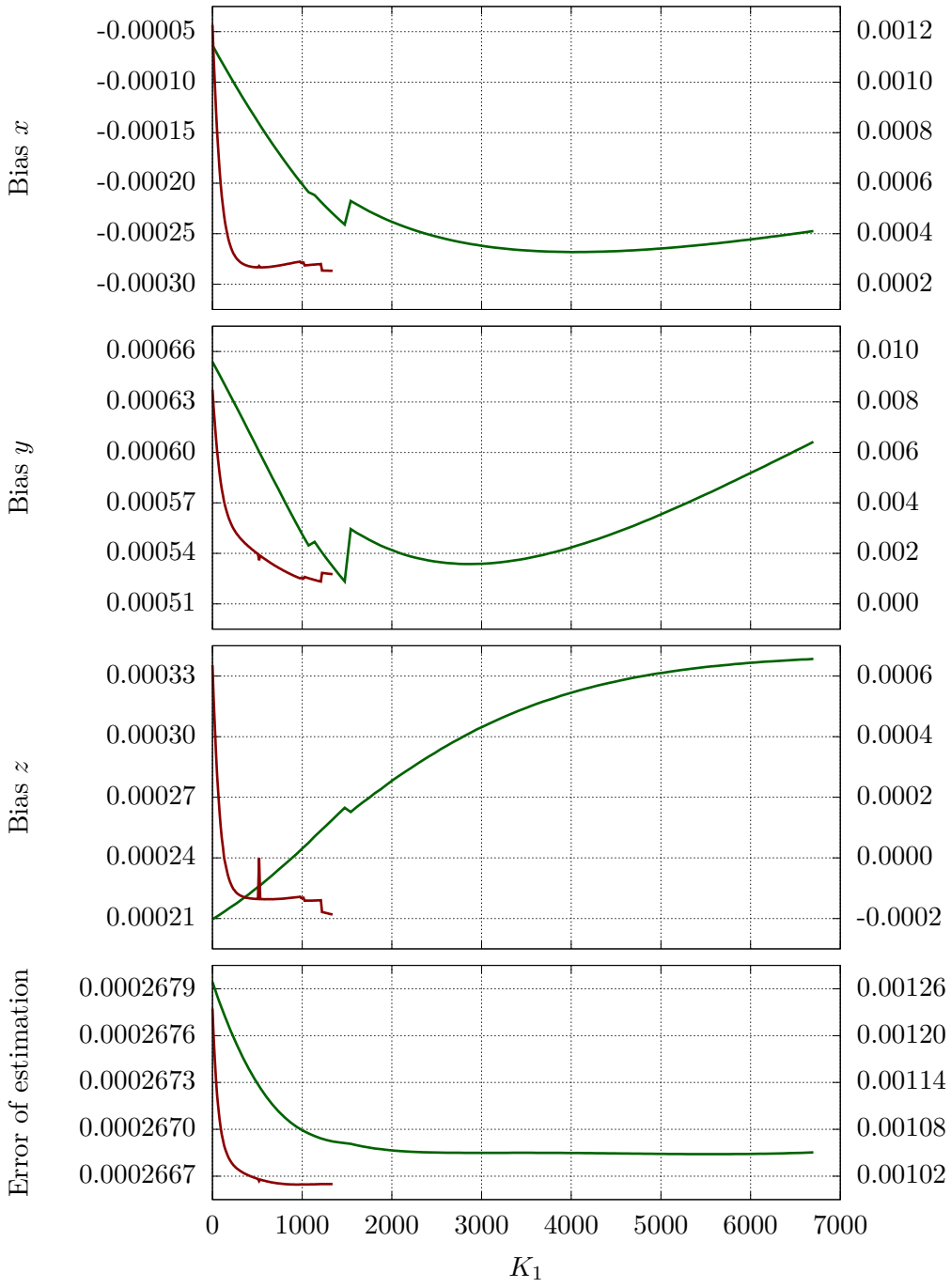

Figure 6.10: The plotted graphs show the result of the estimation of the gyroscope bias in $\frac{\text{rad}}{\text{s}}$ for each K_1 using the weighting strategy. Data sets ID 3 (green) and ID 4 (red) were used, where the left y-axes belong to ID 3 and the right ones to ID 4. The results shown here correspond to all blocks on the right side side of the Fig. 4.1.

ID	Source	Iteration	Euler x	Euler y	Euler z	Error
1	LM	9	$-0.5942°$	$0.8012°$	$89.3780°$	$1.324E-4$
1	LM	0	$-9.6517°$	$-13.9831°$	$89.5419°$	$1.898E-3$
1	Ceres	3	$-0.5838°$	$0.7362°$	$89.5016°$	$1.507E-4$
2	LM	14	$2.2081°$	$2.2999°$	$87.7187°$	$4.712E-4$
2	LM	0	$14.6263°$	$-3.3292°$	$82.3015°$	$2.487E-3$
2	Ceres	19	$1.9212°$	$2.7395°$	$87.8769°$	$4.937E-4$
3	LM	1	$179.9026°$	$0.5620°$	$-89.8961°$	$2.667E-4$
3	LM	0	$179.9438°$	$0.5096°$	$-89.8824°$	$2.759E-4$
3	Ceres	1	$179.9544°$	$0.6391°$	$-89.9130°$	$3.110E-4$
4	LM	7	$179.2842°$	$0.4649°$	$-91.4632°$	$1.007E-3$
4	LM	0	$179.4550°$	$0.6554°$	$-91.7345°$	$1.031E-3$
4	Ceres	0	$169.8912°$	$0.2874°$	$-97.2411°$	$3.210E-3$

Table 6.9: Results of the iterative process of alternating estimation of extrinsic rotation, represented in Euler angles, using the LM algorithm and the Ceres Solver. The iterations here indicate the point at which the error became minimal. Note that the results for zero iterations are taken from Table 6.5 and serve as a comparison if the iterative strategy is omitted.

Evaluation of the Weighting Scheme

At this point, weighting is now introduced in the estimation of the gyroscope bias. Considering again Fig. 4.1, the results are presented which are obtained when the whole process on the right side is executed (red and green blocks). The software now estimates the bias for different factors K_1 and selects the result for which the error becomes minimal. Therefore, for each K_1, the process explained in the previous section is performed until the result does not change anymore. The results of this are shown in Fig. 6.9 for ID 1 and ID 2 and in Fig. 6.10 for ID 3 and ID 4 together with the resulting error of the estimate. Both diagrams have the purpose to give the reader an impression how the software determines the optimal factor K_1. The results with minimum error (best K_1) are listed in Table 6.8. In addition, the results (without weighting) of the previous section are also listed there to enable a comparison. Note that the weighting is only implemented for the LM algorithm.

Looking at Table 6.8, it is noticeable that the weighting achieves a smaller error for each data set.

6.4.3 Evaluation of the Iterative Estimation Process

Once the bias of the gyroscope has been determined, this allows the pre-integrations to be corrected and the software begins estimating extrinsic rotation again. Hence the process of alternating estimation of rotation and bias is applied. It is expected that the estimates are refined at each step.

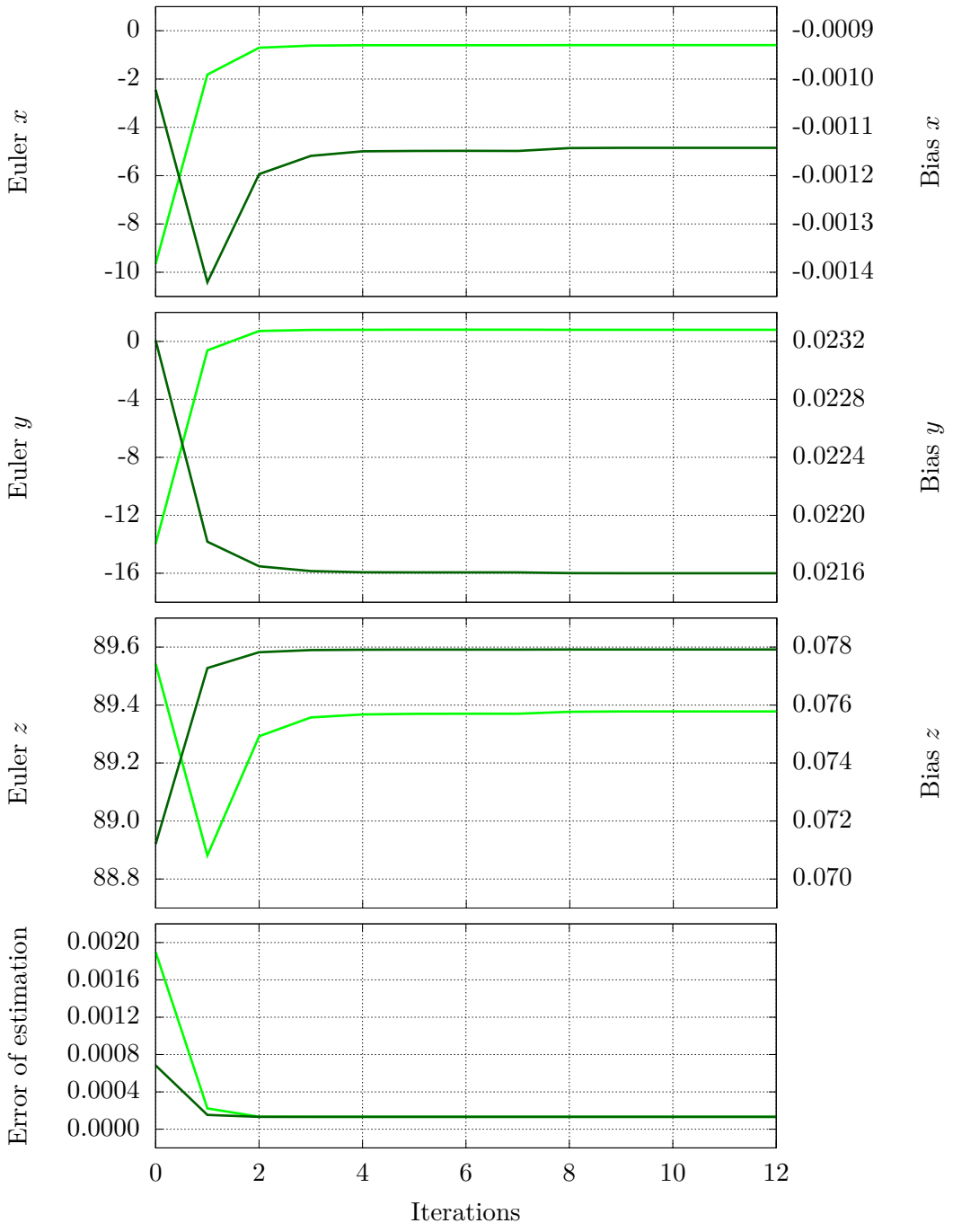

Figure 6.11: The plotted graphs show the result of the iterative process of alternating estimation of extrinsic rotation, represented in Euler angles in °, and gyroscope bias in $\frac{\text{rad}}{\text{s}}$ for each iteration. Dataset ID 1 was used, where the left y-axis belong to the extrinsic rotation (green) and the right y-axis to the gyroscope bias (dark-green). The results shown here correspond to all blocks of Fig. 4.1.

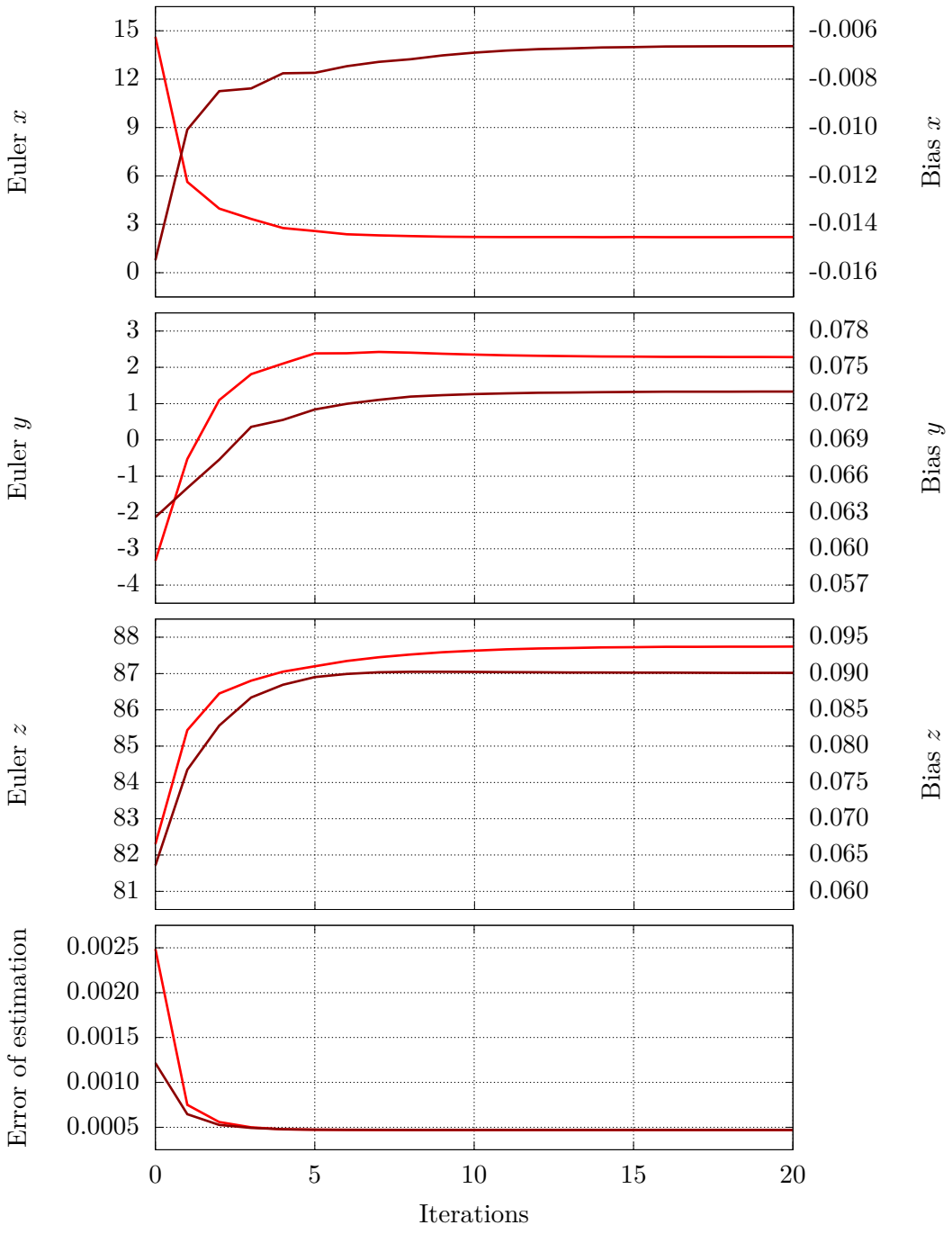

Figure 6.12: The plotted graphs show the result of the iterative process of alternating estimation of extrinsic rotation, represented in Euler angles in °, and gyroscope bias in $\frac{\text{rad}}{\text{s}}$ for each iteration. Dataset ID 2 was used, where the left y-axis belong to the extrinsic rotation (red) and the right y-axis to the gyroscope bias (dark-red). The results shown here correspond to all blocks of Fig. 4.1.

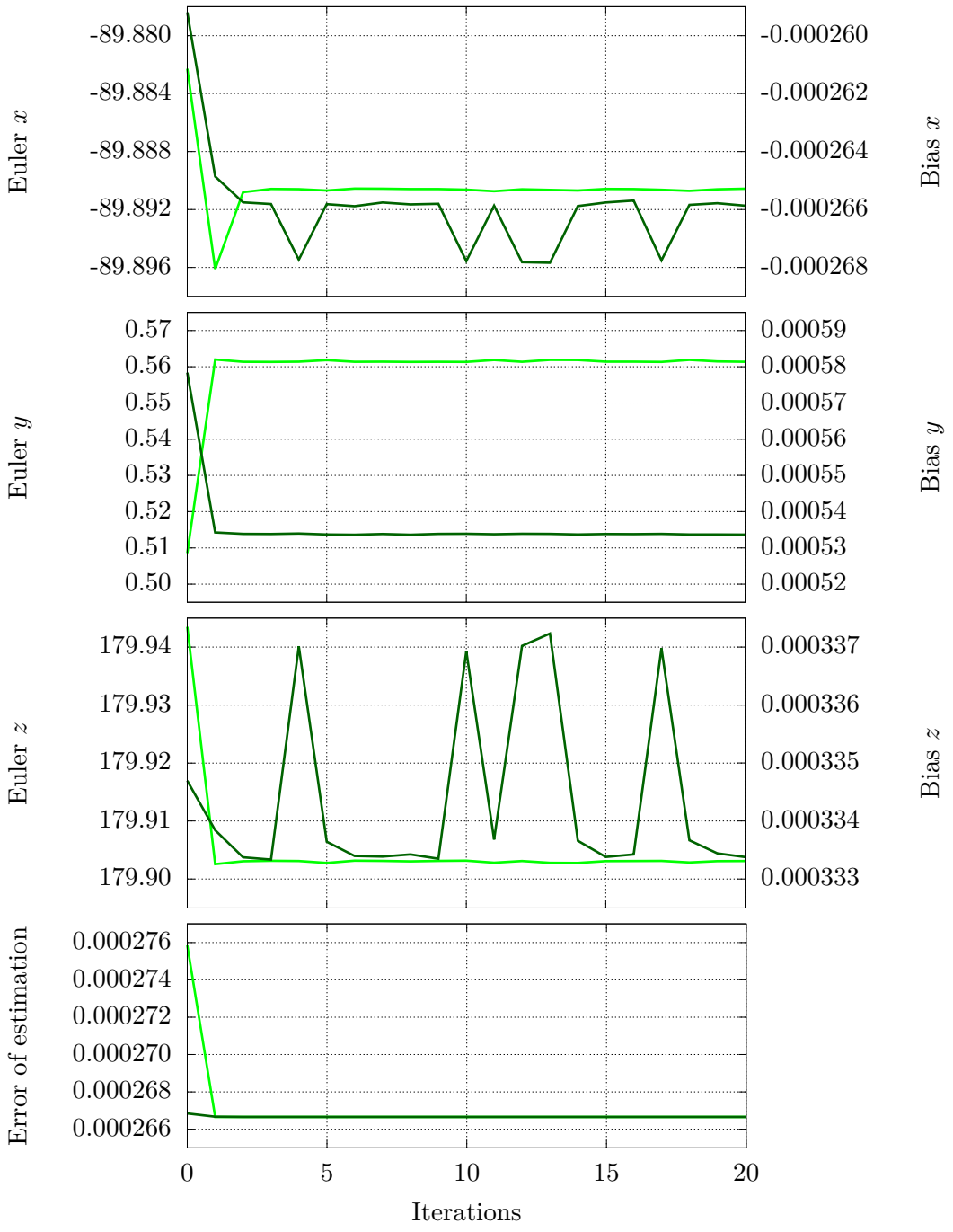

Figure 6.13: The plotted graphs show the result of the iterative process of alternating estimation of extrinsic rotation, represented in Euler angles in °, and gyroscope bias in $\frac{\text{rad}}{\text{s}}$ for each iteration. Dataset ID 3 was used, where the left y-axis belong to the extrinsic rotation (green) and the right y-axis to the gyroscope bias (dark-green). The results shown here correspond to all blocks of Fig. 4.1.

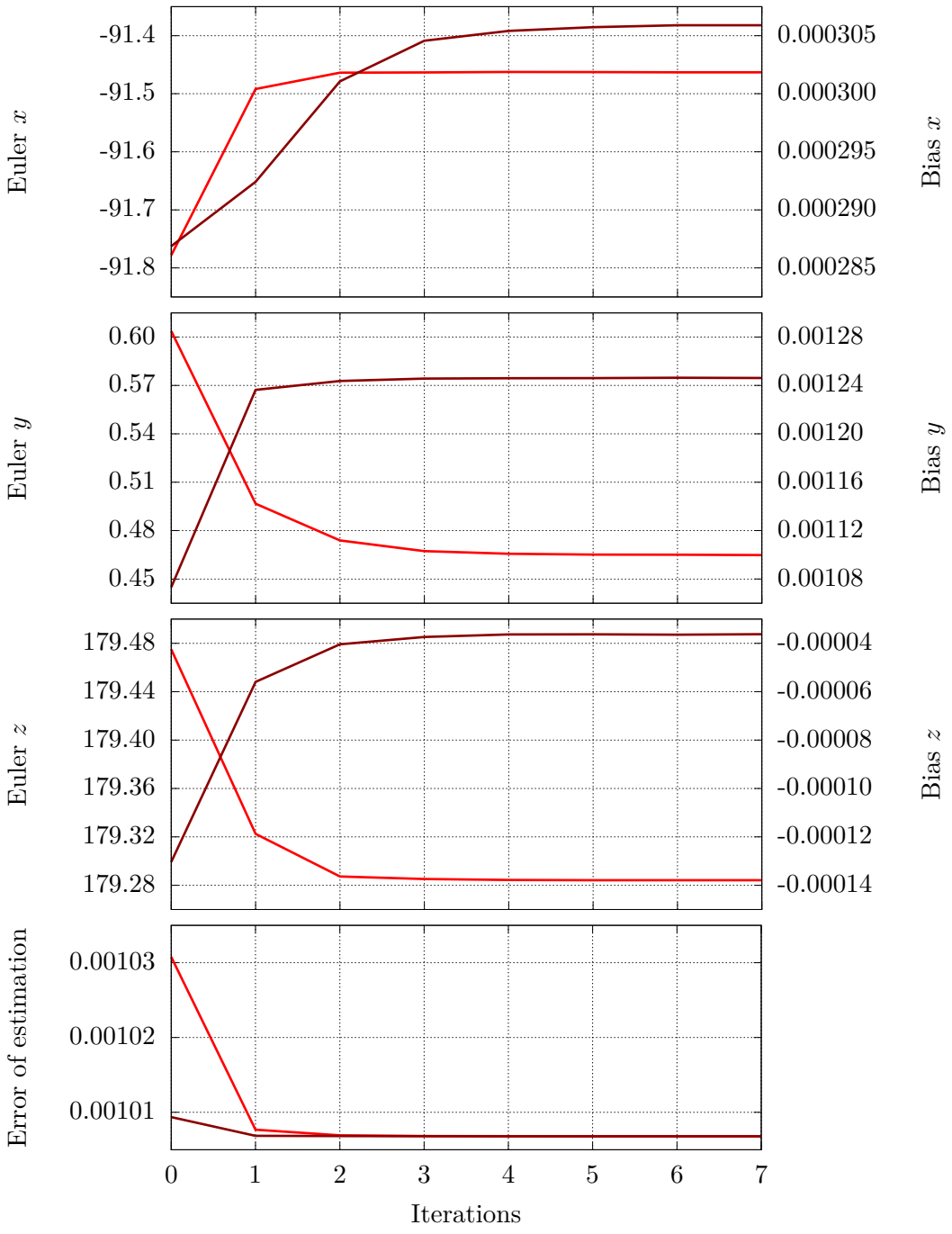

Figure 6.14: The plotted graphs show the result of the iterative process of alternating estimation of extrinsic rotation, represented in Euler angles in °, and gyroscope bias in $\frac{\text{rad}}{\text{s}}$ for each iteration. Dataset ID 4 was used, where the left y-axis belong to the extrinsic rotation (red) and the right y-axis to the gyroscope bias (dark-red). The results shown here correspond to all blocks of Fig. 4.1.

ID	Source	Iteration	Bias x	Bias y	Bias z	Error
1	LM	10	$-1.142E-3$	0.0216	0.0779	$1.324E-4$
1	LM	0	$-1.028E-3$	0.0232	0.0713	$6.830E-4$
1	Ceres	10	$-3.843E-4$	0.0220	0.0782	$1.508E-4$
2	LM	19	$-6.637E-3$	0.0730	0.0901	$4.712E-4$
2	LM	0	-0.0154	0.0627	0.0636	$1.210E-3$
2	Ceres	3	$-2.558E-3$	0.0635	0.0888	$4.934E-4$
3	LM	7	$-2.658E-4$	$5.338E-4$	$3.334E-4$	$2.667E-4$
3	LM	0	$-2.600E-4$	$5.765E-4$	$3.347E-4$	$2.668E-4$
3	Ceres	12	$-1.494E-4$	$1.125E-3$	$2.144E-4$	$3.111E-4$
4	LM	7	$3.059E-4$	$1.246E-3$	$-3.624E-5$	$1.007E-3$
4	LM	0	$2.872E-4$	$1.074E-3$	$-1.296E-4$	$1.009E-3$
4	Ceres	2	$3.512E-3$	0.0258	$-3.448E-3$	$3.750E-3$

Table 6.10: Results of the iterative process of alternating estimation of gyroscope bias in $\frac{\text{rad}}{\text{s}}$, using the LM algorithm and the Ceres Solver. The iterations here indicate the point at which the error became minimal. Note that the results for zero iterations are taken from Table 6.8 and serve as a comparison if the iterative strategy is omitted.

The results of this iterative process are shown for ID 1 in Fig. 6.11, for ID 2 in Fig. 6.12, for ID 3 in Fig. 6.13, and for ID 4 in Fig. 6.14, where after each iteration the rotation and then the bias were estimated. In addition, Table 6.9 lists the final results for extrinsic rotation and Table 6.10 lists the final results for bias. Here, the software selects the results of each iteration with the least error. In the tables and charts, zero iterations represents the results from the previous sections and serves as a comparison value to illustrate the advantage of iterative estimation. In addition, the results are also listed, which the Ceres Solver supplies, although it must be noted that the weighting method is not implemented for this and therefore only the green blocks in Fig. 4.1 were processed.

Looking at the Fig. 6.11 - 6.14, it is noticeable that the values stagnate after a few iterations and the error has clearly decreased. For the fluctuations in the estimation of the bias in Fig. 6.13 no cause is known, since these lie in a range of only $4E-6$, this can be neglected.

Considering Table 6.9, it is noticeable that using the LM algorithm, all errors decrease compared to the results of the previous sections (iteration= 0). Also, the values have a smaller deviation from the ideal results (cf. Table 6.3). Compared to the Ceres Solver, a lower error is obtained with the LM algorithm, which may be due to the non-applied weighting with respect to the Ceres Solver. The results of the Ceres Solver of ID 4 also deviate strongly from the ideal results. For the results of the *UWSensor* dataset (ID 3 and 4), it is noticeable that the iterative process described in this section yields only a small improvement in the error. This is due to the fact that the bias of the gyroscope used here is small and accordingly does not have a large influence on the extrinsic rotation.

Regarding the table 6.10, it is noticeable that the errors of the estimation also decrease when the iterative strategy is applied. However, as is well known, no quantitative assessment of the results of the estimation of the bias can be made. But, since these results led to an improvement of the estimation of the extrinsic rotation, it is concluded that the values of the bias cannot have a large deviation.

6.4.4 Summary

In the previous sections, for the estimation of the extrinsic rotation and the bias of the gyroscope, the individual steps of the proposed calibration approach have been evaluated. Hence, the estimation of the values with and without the use of the weighting strategy and finally the alternating estimation of the values.

In each step a reduction of the error is achieved, so that the values for the extrinsic rotation correspond close to the ideal results. As already mentioned in Section 6.1, it is not possible here to determine the reasons for the remaining deviation from the ideal results. The reasons could be residual errors in the camera poses estimated by *VINS-Mono* or in the minimization algorithms. It is also conceivable and most likely that the values for the ideal results (cf. Table 6.3) are imprecise. For example, for the *UWSensor* dataset it is implausible that the CAD parameter of 180.0° around the X-axis between the camera and the IMU applies exactly, since it is mechanically impossible to align the devices that precisely. Accordingly, a rotation of 179.9026° (cf. Table 6.9 ID 3) might be closer to the real value. However, the actual value is not known and only the effect of the calibration on the pose estimation is evaluated.

6.5 Evaluation of Scale and Translation Estimation

In this section, the results of the individual steps of the estimation of scale, gravity, translation and accelerometer bias is described.

In general, the purpose of the following sections is to show that the estimates are refined by applying the two methods, the weighting of the individual datasets and the alternating estimation of gravity and refined gravity. In each case, for gravity and refined gravity, first the results are presented, which are obtained without the use of the two methods mentioned and with the use of weighting. Finally, the results obtained by alternating the estimation of the parameters are shown.

Since the software estimates many individual parameters in this section, it was decided to limit the evaluation of the estimation of scale, gravity and translation (cf. Section 4.3) to gravity, since the expected norm known for this.

In the course of the refinement and the estimation of the accelerometer bias (cf. Section 4.4), the evaluation is limited to the translation, since the expected values are also known here.

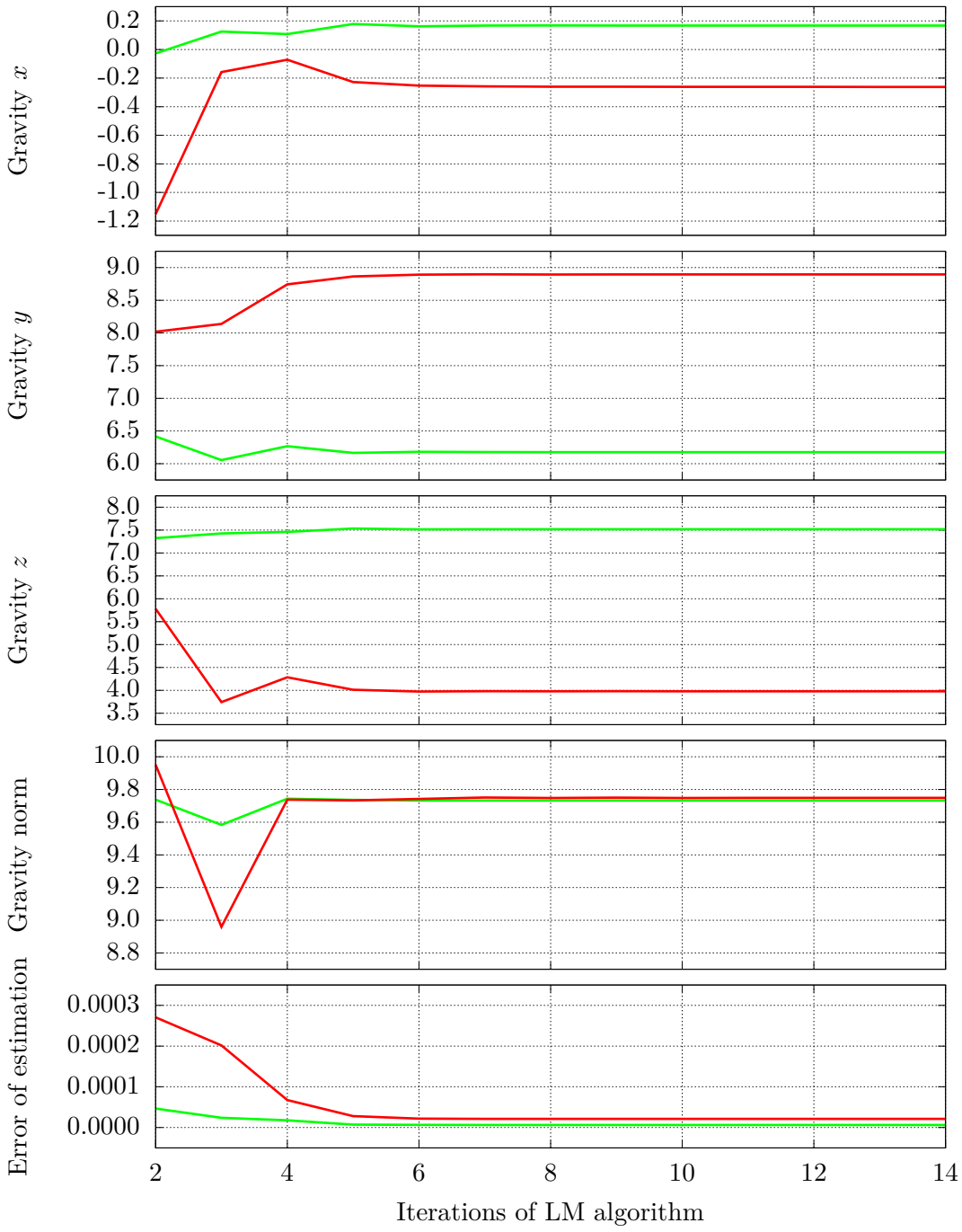

Figure 6.15: The plotted graphs show the result of the estimation of the gravity in $\frac{\mathrm{m}}{\mathrm{s}^2}$, for each iteration of the LM algorithm. Data sets ID 1 (green) and ID 2 (red) were used. The results shown here correspond to the green block on the left side of the Fig. 4.2. For a finer representation, the values are plotted starting with the second iteration, since the initial values are zero at the first iteration.

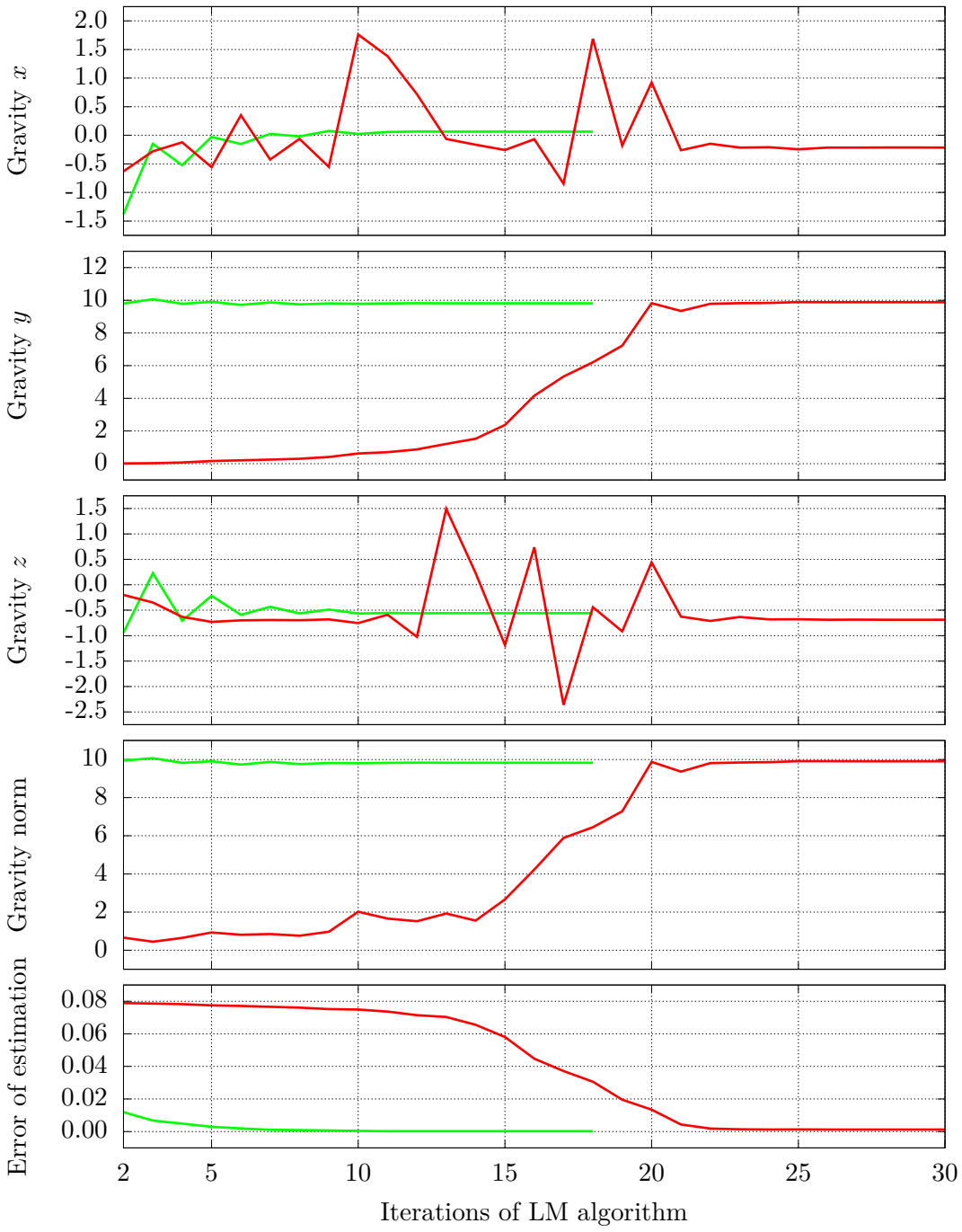

Figure 6.16: The plotted graphs show the result of the estimation of the gravity in $\frac{\mathrm{m}}{\mathrm{s}^2}$, for each iteration of the LM algorithm. Data sets ID 3 (green) and ID 4 (red) were used. The results shown here correspond to the green block on the left side of the Fig. 4.2. For a finer representation, the values are plotted starting with the second iteration, since the initial values are zero at the first iteration.

Source	Gravity x	Gravity y	Gravity z	|Gravity|
EuRoC MAV dataset				
VINS-Mono	1.9261	9.6191	0.02139	9.81007
Ideal result	unknown	unknown	unknown	9.81007
***UWSensor* dataset**				
VINS-Mono	1.0538	9.6529	1.3961	9.81007
Ideal result	unknown	unknown	unknown	9.81007

Table 6.11: Expected results for gravity in $\frac{\mathrm{m}}{\mathrm{s}^2}$ along with the results of the native calculation by *VINS-Mono*.

ID	Source	Gravity x	Gravity y	Gravity z	|Gravity|	Error
1	LM	0.1677	6.1756	7.5202	9.7324	$6.3000E-6$
1	Ceres	0.2052	6.1810	7.5153	9.7328	$6.3187E-6$
2	LM	−0.2615	8.8954	3.9793	9.7484	$2.1200E-5$
2	Ceres	−0.2540	8.8823	4.0097	9.7488	$2.1473E-5$
3	LM	0.06252	9.8174	−0.5570	9.8334	$1.9745E-4$
3	Ceres	0.05960	9.8181	−0.5442	9.8334	$1.9697E-4$
4	LM	−0.2144	9.8795	−0.6863	9.9056	$1.2752E-3$
4	Ceres	−0.3734	9.8141	−1.2893	9.9055	$1.5369E-3$

Table 6.12: Results of the estimation of gravity using the LM algorithm and Ceres Solver. The results shown here correspond to the green block on the left side of Fig. 4.2.

6.5.1 Scale Estimation

Once the extrinsic rotation and the gyroscope bias have been estimated by the software, the estimation of the scale, the gravity and the translation is performed, shown in Fig. 4.2 on the left. As mentioned earlier, solely the results for gravity will be discussed in this section. The values for the expected norm of the gravity vector are listed in Table 6.11, along with the results of the native calculation of *VINS-Mono*. Considering the norm of the gravity vector determined by *VINS-Mono* here, it is assumed that the results are optimal. However, since *VINS-Mono* neither considers the translation between camera and IMU nor the bias of the accelerometer in the determination of the gravity vector, it must be assumed that the results of the gravity vector are falsified here. The expected norm of $9.81007\frac{\mathrm{m}}{\mathrm{s}^2}$ is only achieved by normalizing the norm of the estimated vector to the desired amount, which trivially does not lead to an accurate result. At this point, it should also be mentioned that the real magnitude of the gravity vector is not known exactly, the ideal results in Table 6.11, however, represent a good approximate value.

First, the results are analyzed, in case the weighting is omitted, therefore only the result of the green block on the left side of Fig. 4.2. The results of the iterative process of the LM algorithm

ID	Best K_2	Gravity x	Gravity y	Gravity z	|Gravity|	Error
1	24905	0.1687	6.1744	7.5195	9.7310	$6.2353E-6$
1	0	0.1677	6.1756	7.5202	9.7324	$6.3000E-6$
2	50490	−0.2857	8.9093	4.0270	9.8086	$2.0168E-5$
2	0	−0.2615	8.8954	3.9793	9.7484	$2.1200E-5$
3	5643	0.06204	9.8186	−0.5616	9.8357	$1.9166E-4$
3	0	0.06252	9.8174	−0.5570	9.8334	$1.9745E-4$
4	1131	−0.2214	9.8249	−0.6629	9.8511	$1.1802E-3$
4	0	−0.2144	9.8795	−0.6863	9.9056	$1.2752E-3$

Table 6.13: Shows the results of the estimation of gravity in $\frac{\mathrm{m}}{\mathrm{s}^2}$, using the LM algorithm and the weighting strategy. The results shown here correspond to all blocks on the left side of Fig. 4.2. Note that the results for $K_2 = 0$ are taken from Table 6.12 and serve as a comparison if the weighting is not used.

are shown in Fig. 6.15 for ID 1 and 2 and in Fig. 6.16 for ID 3 and 4 together with the resulting error of the estimation. Both diagrams have the purpose to give the reader an impression of how the LM algorithm iteratively finds a solution and how the error of the estimation behaves during this process. Final results after termination of the LM algorithm are listed in Table 6.12, along with the results computed by the Ceres Solver.

Looking at the table 6.12 it is noticeable that all results for the norm of the gravity vector lie in the proximity of the ideal result. Except for ID 3, the LM algorithm delivers better results than the Ceres Solver, defined by the respective error. The smaller error for the EuRoC MAV datasets (ID 1 and 2) is also remarkable, which is either due to a better camera pose estimation by *VINS-Mono* or to an inherently smaller bias of the accelerometer of the EuRoC MAV.

Comparing Fig 6.15 and 6.16, it is noticeable that the LM algorithm for ID 4 needs comparatively many iterations until the error becomes minimal, also a significant alternation of the gravity estimates is observed. For both observations no reason is known, possibly this is caused by not optimal initial values during the minimization.

Evaluation of the Weighting Scheme

At this point, weighting is now introduced in the estimation of gravity. Considering again Fig. 4.2, the results are presented which are obtained when the whole process on the left side is executed (red and green blocks). The software now estimates the parameters for different factors K_2 and selects the result for which the error becomes minimal. Therefore, for each K_2, the process explained in the previous section is performed until the result does not change anymore. The results of this are shown in Fig. 6.17 for ID 1 and ID 2 and in Fig. 6.18 for ID 3 and ID 4 together with the resulting error of the estimate. Both diagrams have the purpose to give the reader an impression how the software determines the optimal factor K_2. The results with minimum error (best K_2) are listed in Table 6.13. In addition, the results (without weighting) of the previous section are also listed there to enable a comparison. Note that the weighting is

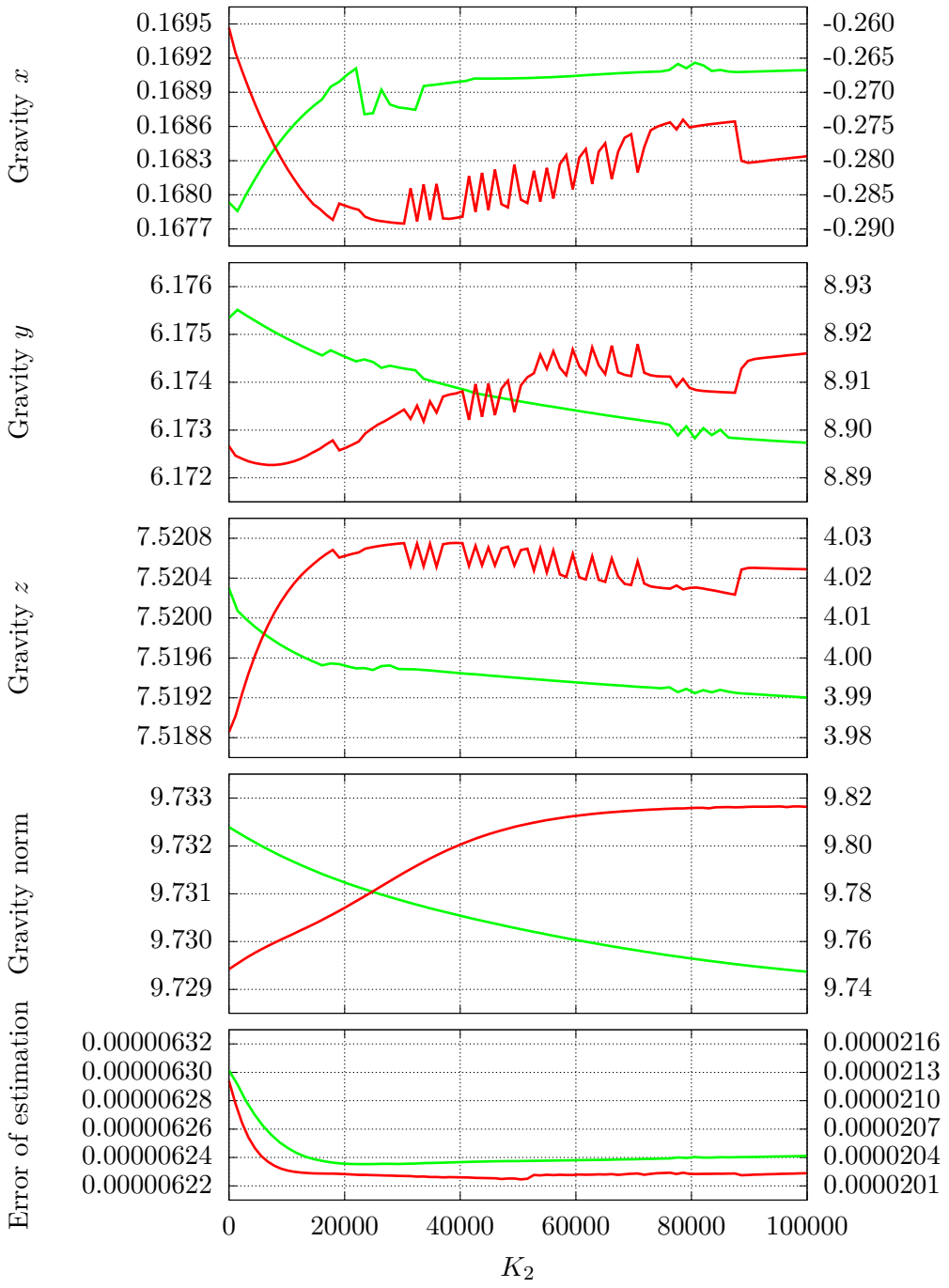

Figure 6.17: The plotted graphs show the result of the estimation of gravity in $\frac{\mathrm{m}}{\mathrm{s}^2}$ for each K_2 using the weighting strategy. Data sets ID 1 (green) and ID 2 (red) were used, where the left y-axes belong to ID 1 and the right ones to ID 2. The results shown here correspond to all blocks on the right side side of the Fig. 4.2.

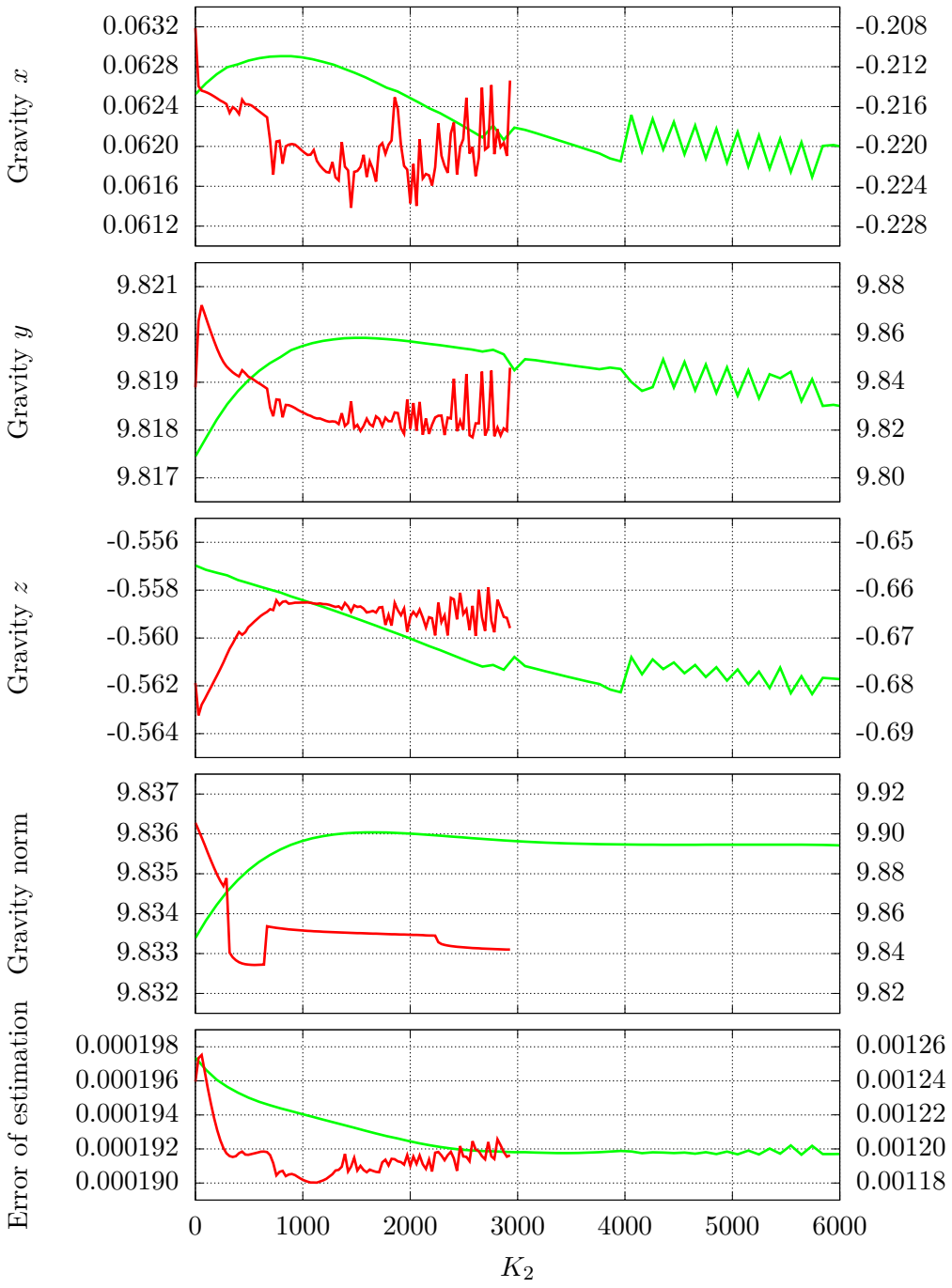

Figure 6.18: The plotted graphs show the result of the estimation of gravity in $\frac{m}{s^s}$ for each K_2 using the weighting strategy. Data sets ID 3 (green) and ID 4 (red) were used, where the left y-axes belong to ID 3 and the right ones to ID 4. The results shown here correspond to all blocks on the right side side of the Fig. 4.2.

Source	Trans. x	Trans. y	Trans. z
EuRoC MAV dataset			
VINS-Mono	0.01563	0.06613	−0.002926
Ideal result	0.02000	0.06000	−0.01000
***UWSensor* dataset**			
VINS-Mono	−0.03098	0.004238	0.3288
Ideal result	0.0	−0.02000	0.3500

Table 6.14: Results for translation in m expected according to the developers of the respective sensor systems along with the results of the native calculation by *VINS-Mono*. Note here that *VINS-Mono* optimizes the values for translation during the entire period of pose estimation, whereby this process is shown in the Fig. 6.19 and 6.20. The results of *VINS-Mono* shown in this table here correspond to the final results of this process.

ID	Source	Trans. x	Trans. y	Trans. z	Error
1	LM	0.01356	0.07136	−0.02129	$7.3500E-6$
1	Ceres	0.01630	0.06454	−0.01943	$6.3516E-6$
2	LM	−0.02085	0.05194	−0.07281	$2.0400E-5$
2	Ceres	−0.01260	0.05975	−0.06559	$1.9405E-5$
3	LM	−0.04710	−0.003861	0.3570	$2.0271E-4$
3	Ceres	−0.04747	−0.01967	0.3562	$1.7695E-4$
4	LM	−0.08502	−0.1929	−0.001135	$1.2659E-3$
4	Ceres	−0.07588	−0.3957	0.01145	$1.0755E-3$

Table 6.15: Results of the estimation of translation in m using the LM algorithm and Ceres Solver. The results shown here correspond to the green block on the right side of Fig. 4.2.

only implemented for the LM algorithm.

Looking at Table 6.13, it is noticeable that the weighting produces a smaller error for each data set. Especially for ID 3 and 4 it is seen that the results using the weighting are much closer to the ideal results. Considering the result of ID 1, it is noticeable that this deviate somewhat more from the ideal results due to the weighting, compared to the values without weighting. Mathematically, however, the results with weighting are still better, since they have a smaller error. At this point it should be pointed out again that so far no accelerometer bias has been included in the calculation.

For the fluctuations seen in Fig. 6.17 and 6.18 no reason is known, nevertheless, since the error of the estimates does not become minimal in the range of these fluctuations (cf. Table 6.13) they have no influence on the result.

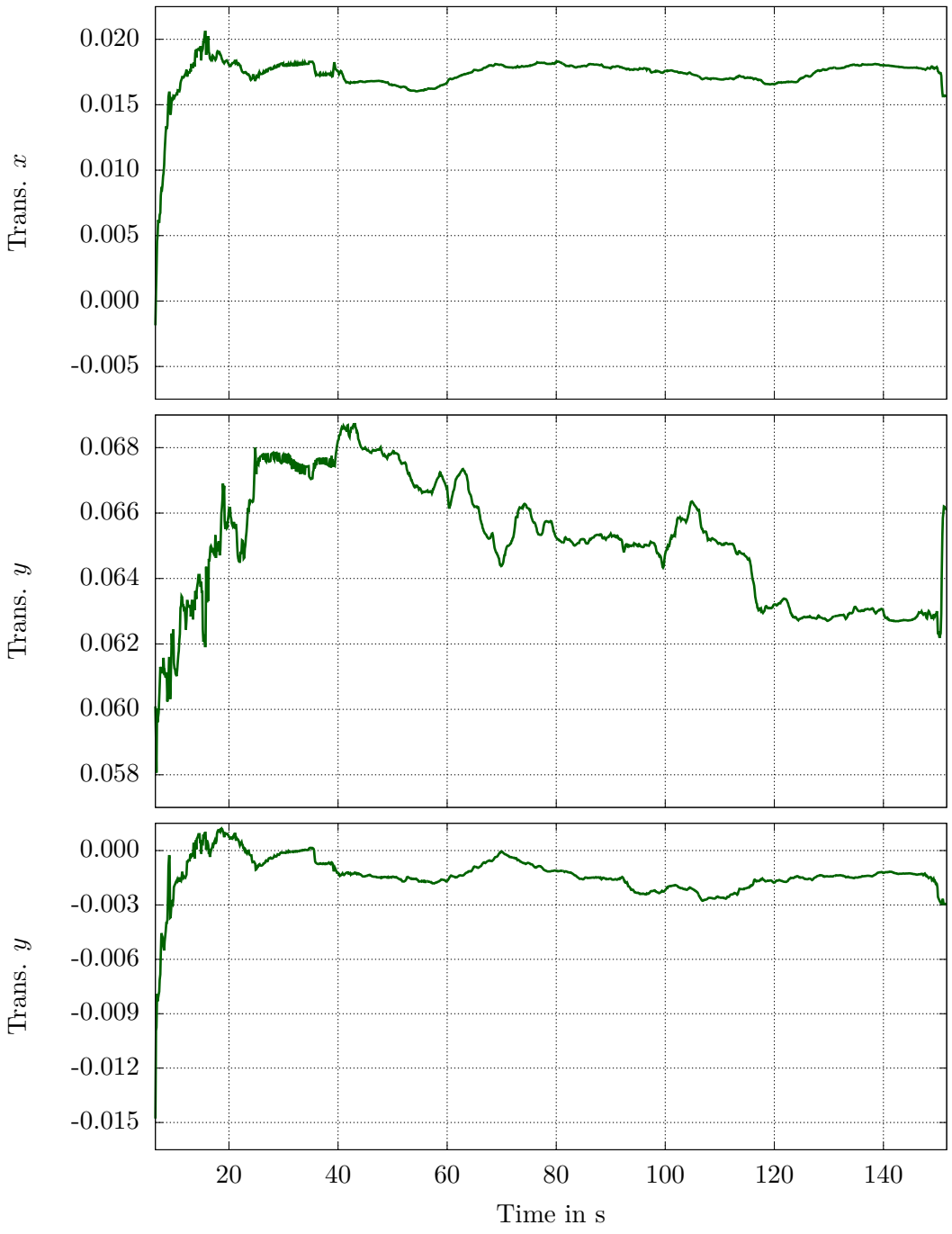

Figure 6.19: The plotted graphs show the translation in m estimated by *VINS-Mono* during the process of pose estimation for the EuRoC MAV dataset. Note that the values are plotted from the time when the first estimate is available, because during the initialization phase of *VINS-Mono* the translation is not taken into account by the software.

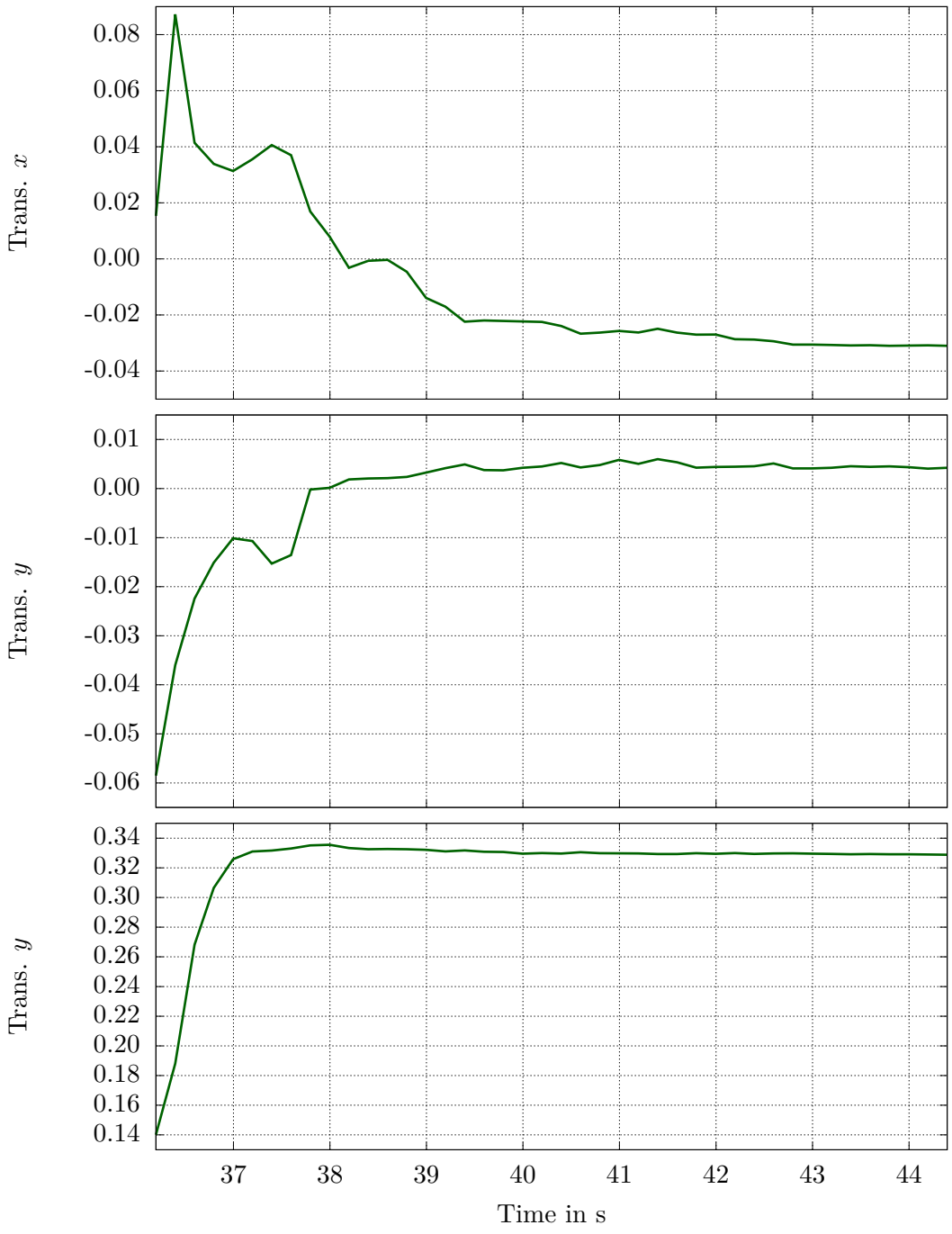

Figure 6.20: The plotted graphs show the translation in m estimated by *VINS-Mono* during the process of pose estimation for the *UWSensor* dataset. Note that the values are plotted from the time when the first estimate is available, because during the initialization phase of *VINS-Mono* the translation is not taken into account by the software.

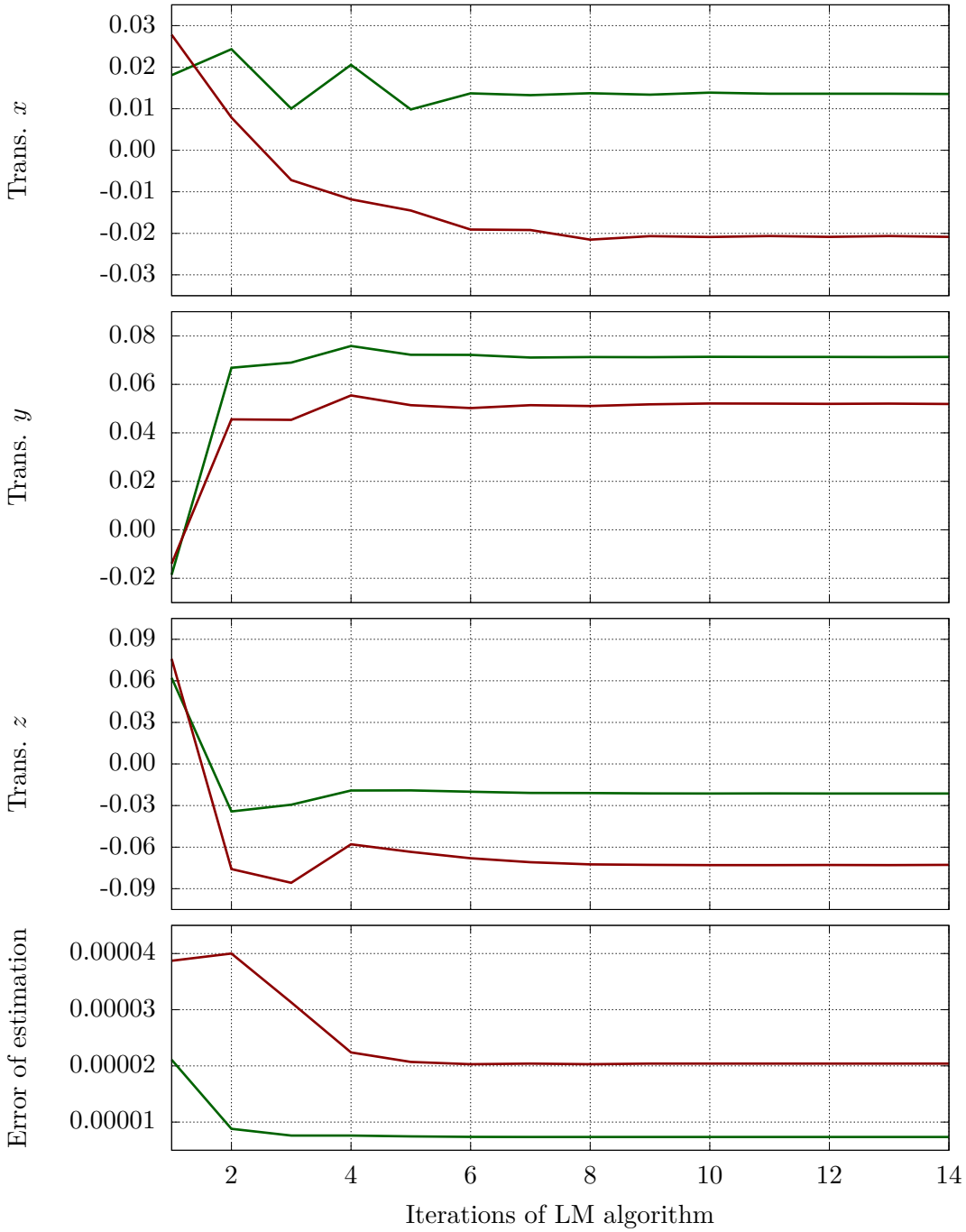

Figure 6.21: The plotted graphs show the result of the estimation of translation in m between camera and IMU, for each iteration of the LM algorithm. Data sets ID 1 (green) and ID 2 (red) were used. The results shown here correspond to the green block on the right side of the Fig. 4.2.

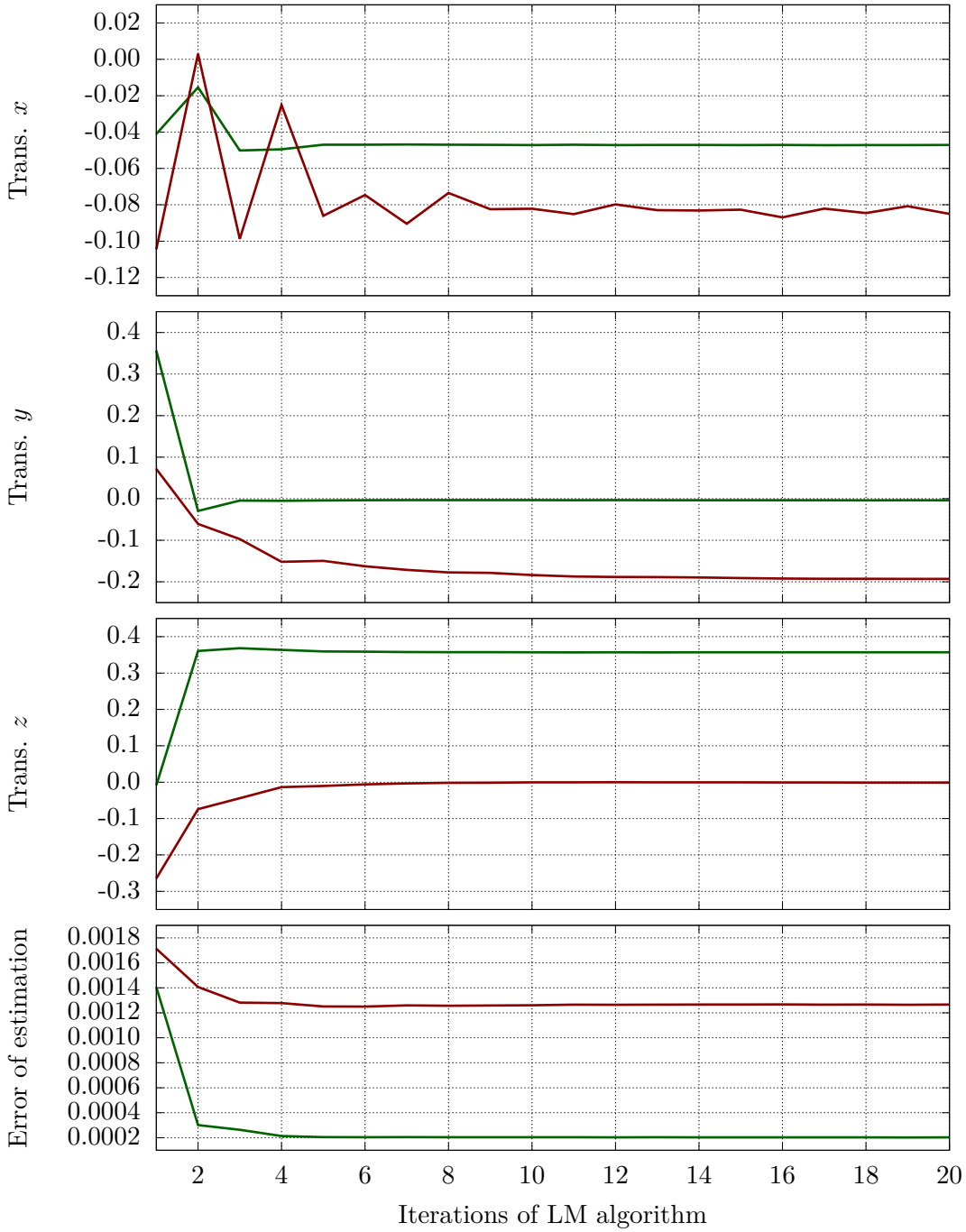

Figure 6.22: The plotted graphs show the result of the estimation of translation in m between camera and IMU, for each iteration of the LM algorithm. Data sets ID 3 (green) and ID 4 (red) were used. The results shown here correspond to the green block on the right side of the Fig. 4.2.

6.5.2 Accelerometer Bias Estimation

Once the scale, the gravity and the translation have been estimated by the software, the refinement of these parameters is performed, shown in Fig. 4.2 on the right. As mentioned earlier, solely the results for translation is discussed in this section. The values for the expected translations are listed in Table 6.14, along with the results of the native calculation of *VINS-Mono*. Since *VINS-Mono* optimizes the translation between the camera and the IMU during the entire process of pose estimation, the final results of this procedure are shown in the Table 6.14. Fig.6.19 and 6.20 show the temporal progression of the translation values for the EuRoC MAV dataset and the *UWSensor* dataset, respectively. Looking at Fig 6.19, it is noticeable that the values converge towards the ideal results, but even the final result shows a deviation of several *cm* in all axes. Here it must be pointed out that the result should improve with increasing time, since more data is continuously available for evaluation. Also, in the course of time, loops are closed with respect to the loop-closing technique of *VINS-Mono* (cf. Article [27]), which further improves the results.

Regarding Fig. 6.20, it is noticeable that although the result of the z-translation converges in the direction of the ideal results, the sign of the other results does not even match.

As will be shown later, the results of *VINS-Mono* have a large deviation from the ideal result because during the initialization phase the visual scale is determined without taking into account the translation between the camera and the IMU and therefore a large error in the translation remains present in the course of the position estimation.

First, the results are analyzed, in case the weighting is omitted, therefore only the result of the green block on the right side of Fig. 4.2. The results of the iterative process of the LM algorithm are shown in Fig. 6.21 for ID 1 and 2 and in Fig. 6.22 for ID 3 and 4 together with the resulting error of the estimation. Both diagrams have the purpose to give the reader an impression of how the LM algorithm iteratively finds a solution and how the error of the estimation behaves during this process. Final results after termination of the LM algorithm are listed in Table 6.15, along with the results computed by the Ceres Solver.

Looking at Table 6.15, it is noticeable that in this case the Ceres Solver provides results with a smaller error in every case. Why in this case compared to the other estimates the Ceres Solver provides a better result compared to the LM algorithm is not known. At this point it turns out why in the introduction (cf. Sec. 6.3) with respect to ID 2 and 4 it is spoken of a bad configuration. For these IDs, the results deviate strongly from the expected results (cf. Table 6.14). Why the estimation is almost unusable with these configurations is not assessed conclusively. Since, in comparison, the estimate of the z-translation for ID 3 shows only a small deviation, it is reasonable to assume that the problem is not the acceleration measurement, but apparently the camera pose estimate of *VINS-Mono*, which is error-prone in the case of ID 2 and 4.

Evaluation of the Weighting Scheme

At this point, weighting is introduced in the estimation of translation. Considering again Fig. 4.2, the results are presented which are obtained when the whole process on the right side is executed

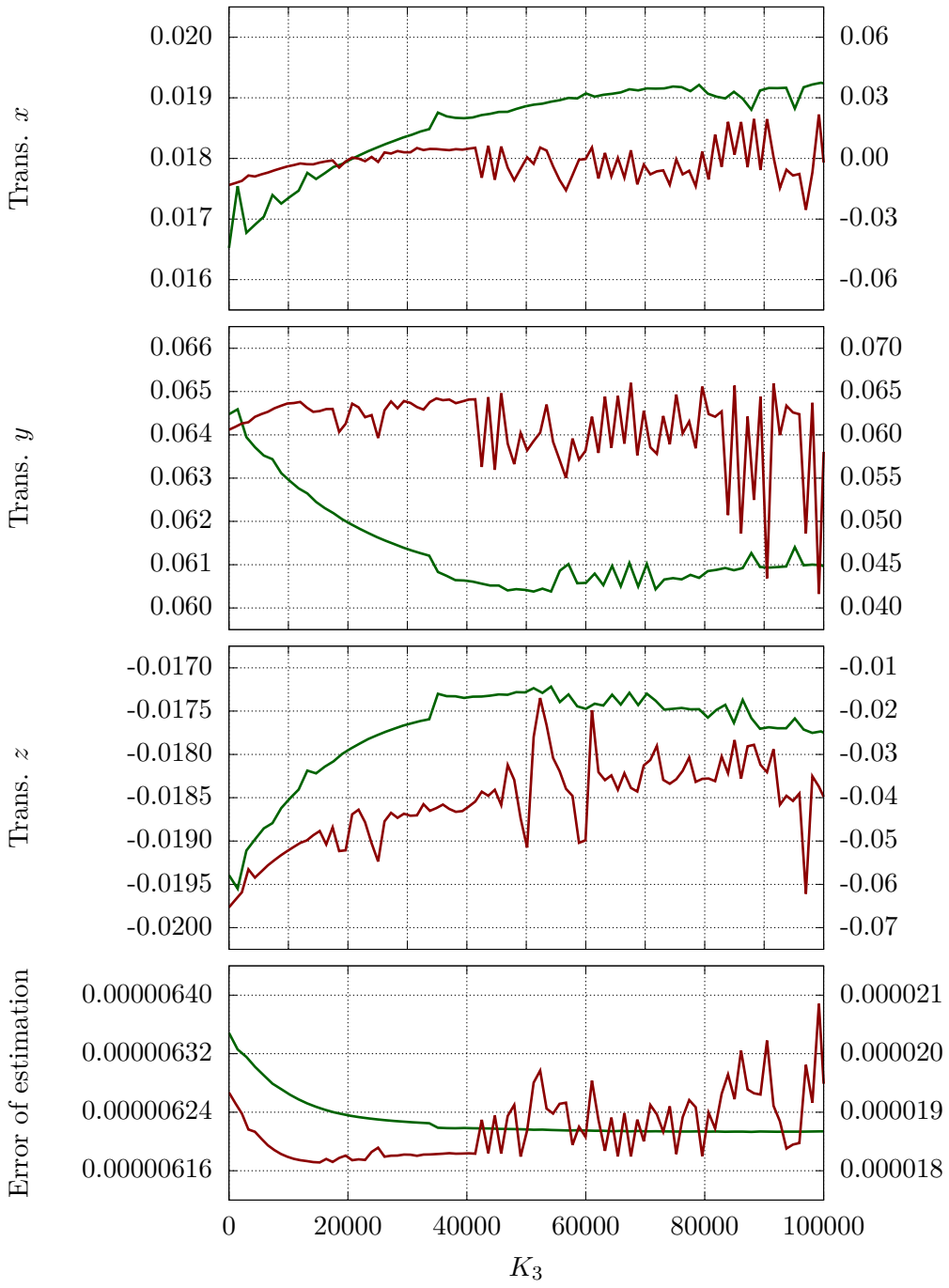

Figure 6.23: The plotted graphs show the result of the estimation of translation in m for each K_3 using the weighting strategy. Data sets ID 1 (green) and ID 2 (red) were used, where the left y-axes belong to ID 1 and the right ones to ID 2. The results shown here correspond to all blocks on the right side side of the Fig. 4.2.

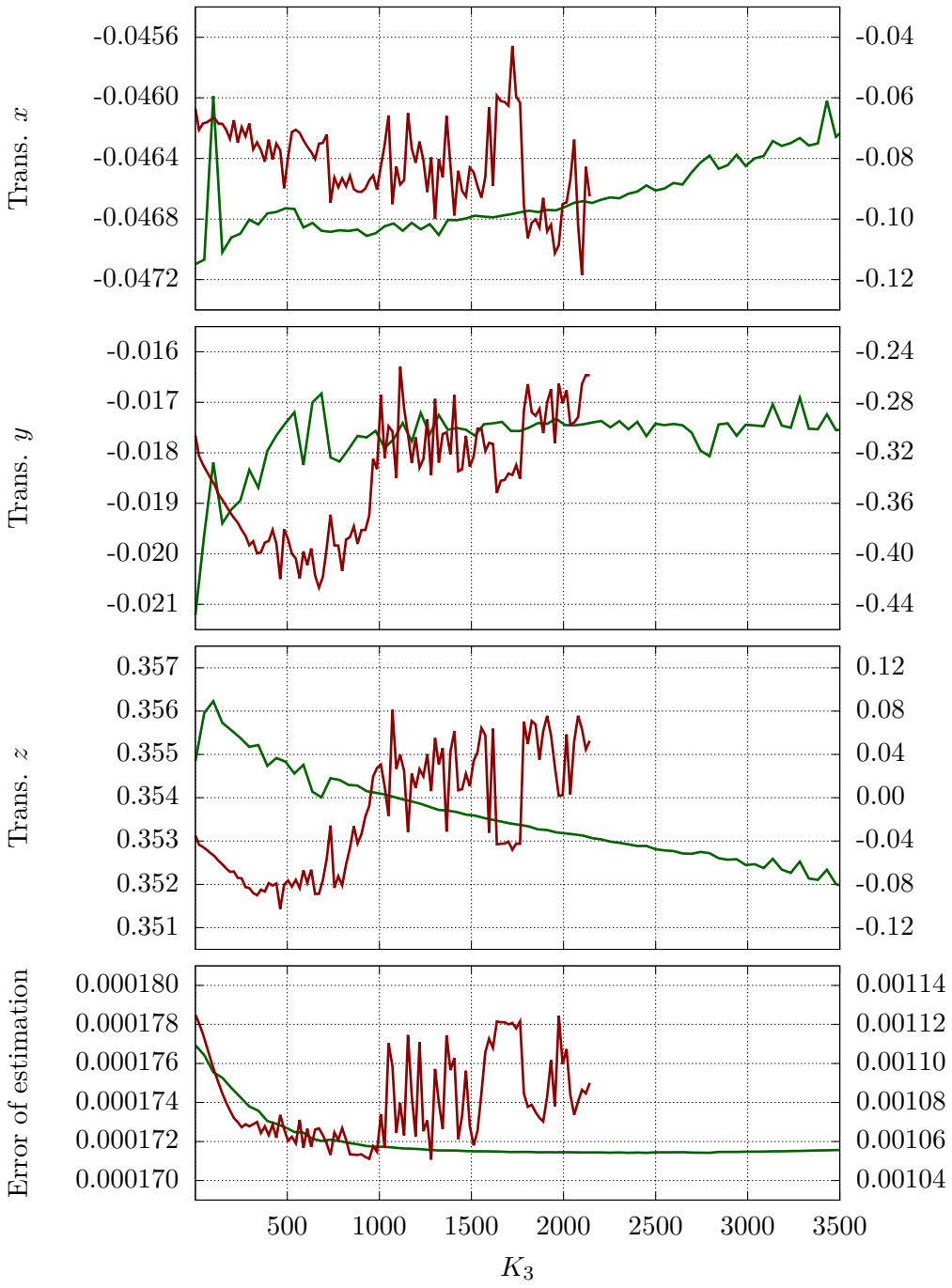

Figure 6.24: The plotted graphs show the result of the estimation of translation in m for each K_3 using the weighting strategy. Data sets ID 3 (green) and ID 4 (red) were used, where the left y-axes belong to ID 3 and the right ones to ID 4. The results shown here correspond to all blocks on the right side side of the Fig. 4.2.

ID	Best K_3	Trans. x	Trans. y	Trans. z	Error
1	87840	0.01881	0.06127	−0.01758	$6.2132E-6$
1	0	0.01356	0.07136	−0.02129	$7.3500E-6$
2	15260	−0.002038	0.06274	−0.04768	$1.8142E-5$
2	0	−0.02085	0.05194	−0.07281	$2.0400E-5$
3	2793	−0.04638	−0.01807	0.3527	$1.7142E-4$
3	0	−0.04710	−0.003861	0.3570	$2.0271E-4$
4	1281	−0.07966	−0.3377	0.006108	$1.0508E-3$
4	0	−0.07588	−0.3957	0.01145	$1.0755E-3$

Table 6.16: Results of the estimation of the translation in m using the LM algorithm and the weighting strategy. The results shown here correspond to all blocks on the left side of Fig. 4.2. Note that the results for $K_3 = 0$ are taken from Table 6.15 and serve as a comparison if the weighting is not used.

(red and green blocks). The software estimates the parameters for different factors K_3 and selects the result for which the error becomes minimal. Therefore, for each K_3, the process explained in the previous section is performed until the result does not change anymore. The results of this are shown in Fig. 6.23 for ID 1 and ID 2 and in Fig. 6.24 for ID 3 and ID 4 together with the resulting error of the estimate. Both diagrams have the purpose to give the reader an impression how the software determines the optimal factor K_3. The results with minimum error (best K_3) are listed in Table 6.16. In addition, the results (without weighting) of the previous section are also listed there to enable a comparison. Note that the weighting is only implemented for the LM algorithm.

Looking at Table 6.16, it is noticeable that by applying the weighting strategy, the error for each data set is reduced. Except for ID 4, it is observed for all IDs that the estimates tend towards the ideal results due to the weighting. Especially with the y-translation of ID 3 a clear improvement of the value can be seen. For ID 4, except for the error, no improvement is observed, which is probably due to poor camera pose estimation. This is apparently also the cause of the fluctuation of the estimates for ID 2 and 4 in Fig. 6.23 and 6.24.

6.5.3 Evaluation of the Iterative Estimation Process

Once the bias of the accelerometer has been determined, this allows the pre-integrations to be corrected and the software begins estimating gravity, hence the process of alternating estimation of scale and refinement is applied. It is expected that the estimates are refined at each step.

The results of this iterative process are shown for ID 1 in Fig. 6.25, for ID 2 in Fig. 6.26, for ID 3 in Fig. 6.27, and for ID 4 in Fig. 6.28, where after each iteration the gravity and then the translation were estimated. In addition, Table 6.17 lists the final results for gravity and Table 6.18 lists the final results for translation. Here, the software selects the results of each iteration with the least error. In the tables and charts, iteration 0 represents the results from the previous sections and serves as a comparison value to illustrate the advantage of iterative

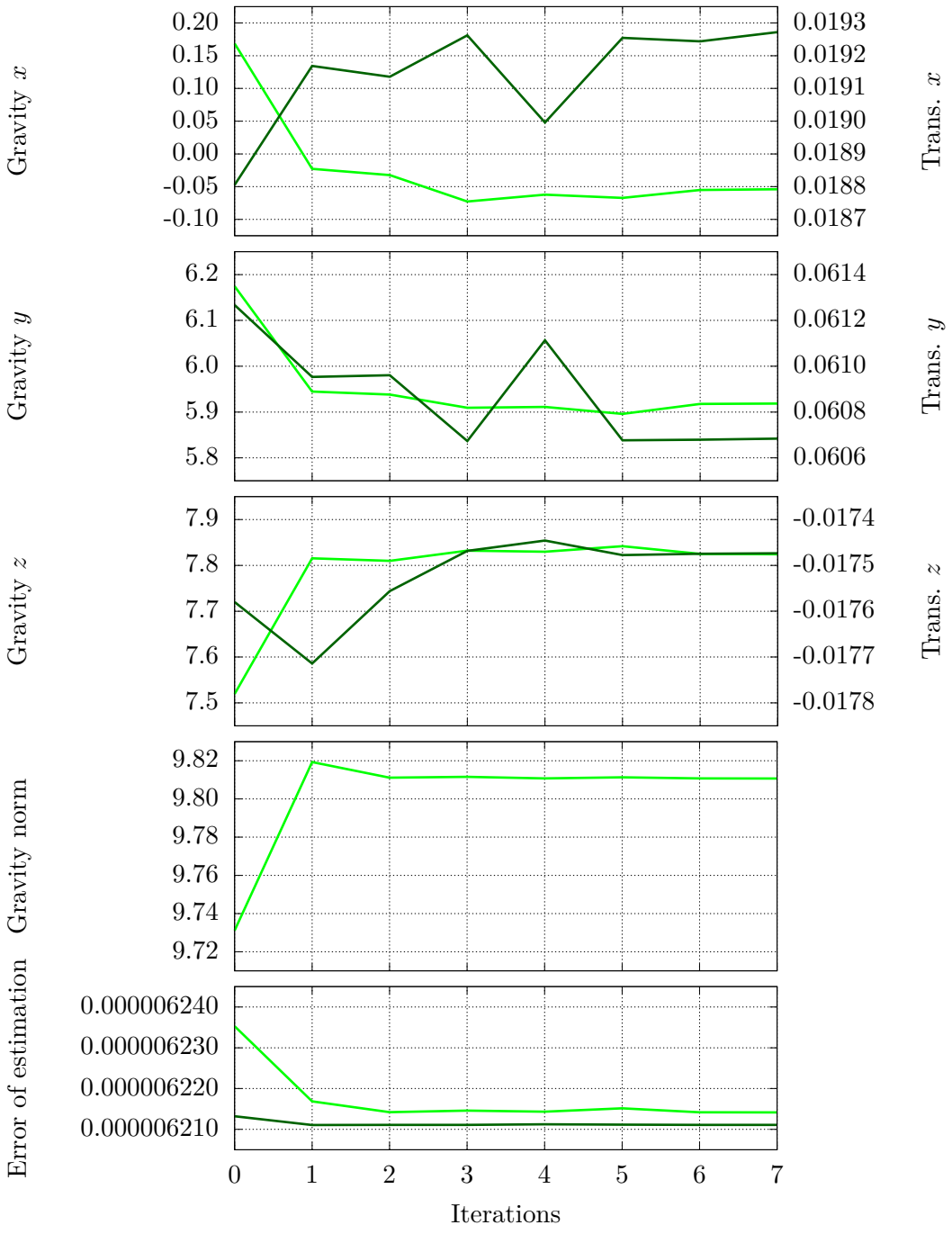

Figure 6.25: The plotted graphs show the result of the iterative process of alternating estimation of gravity in $\frac{\text{m}}{\text{s}^2}$ and translation in m for each iteration. Dataset ID 1 was used, where the left y-axis belong to the gravity (green) and the right y-axis to the translation (dark-green). The results shown here correspond to all blocks of Fig. 4.2.

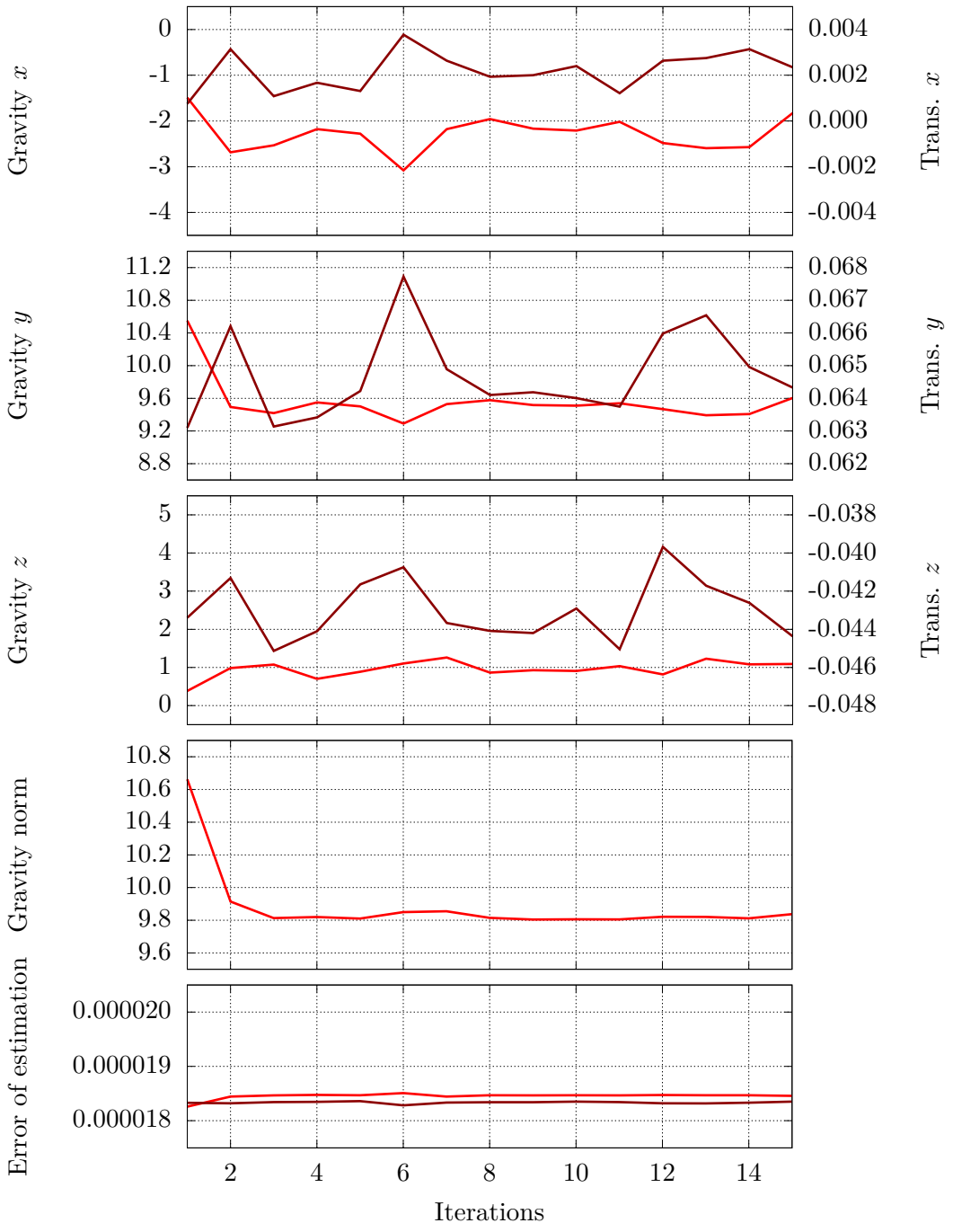

Figure 6.26: The plotted graphs show the result of the iterative process of alternating estimation of gravity in $\frac{\text{m}}{\text{s}^2}$ and translation in m for each iteration. Dataset ID 2 was used, where the left y-axis belong to the gravity (red) and the right y-axis to the translation (dark-red). The results shown here correspond to all blocks of Fig. 4.2.

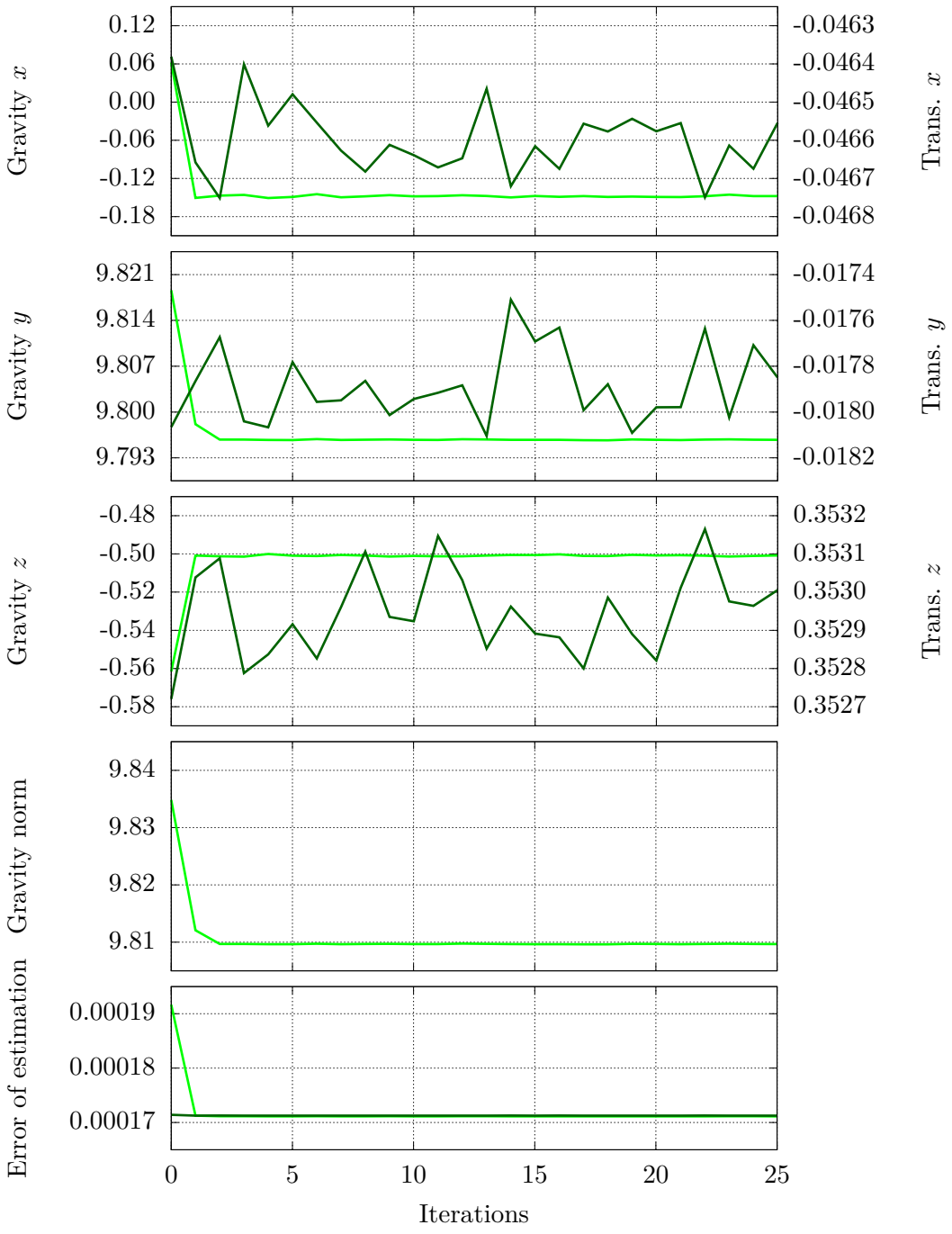

Figure 6.27: The plotted graphs show the result of the iterative process of alternating estimation of gravity in $\frac{\text{m}}{\text{s}^2}$ and translation in m for each iteration. Dataset ID 3 was used, where the left y-axis belong to the gravity (green) and the right y-axis to the translation (dark-green). The results shown here correspond to all blocks of Fig. 4.2.

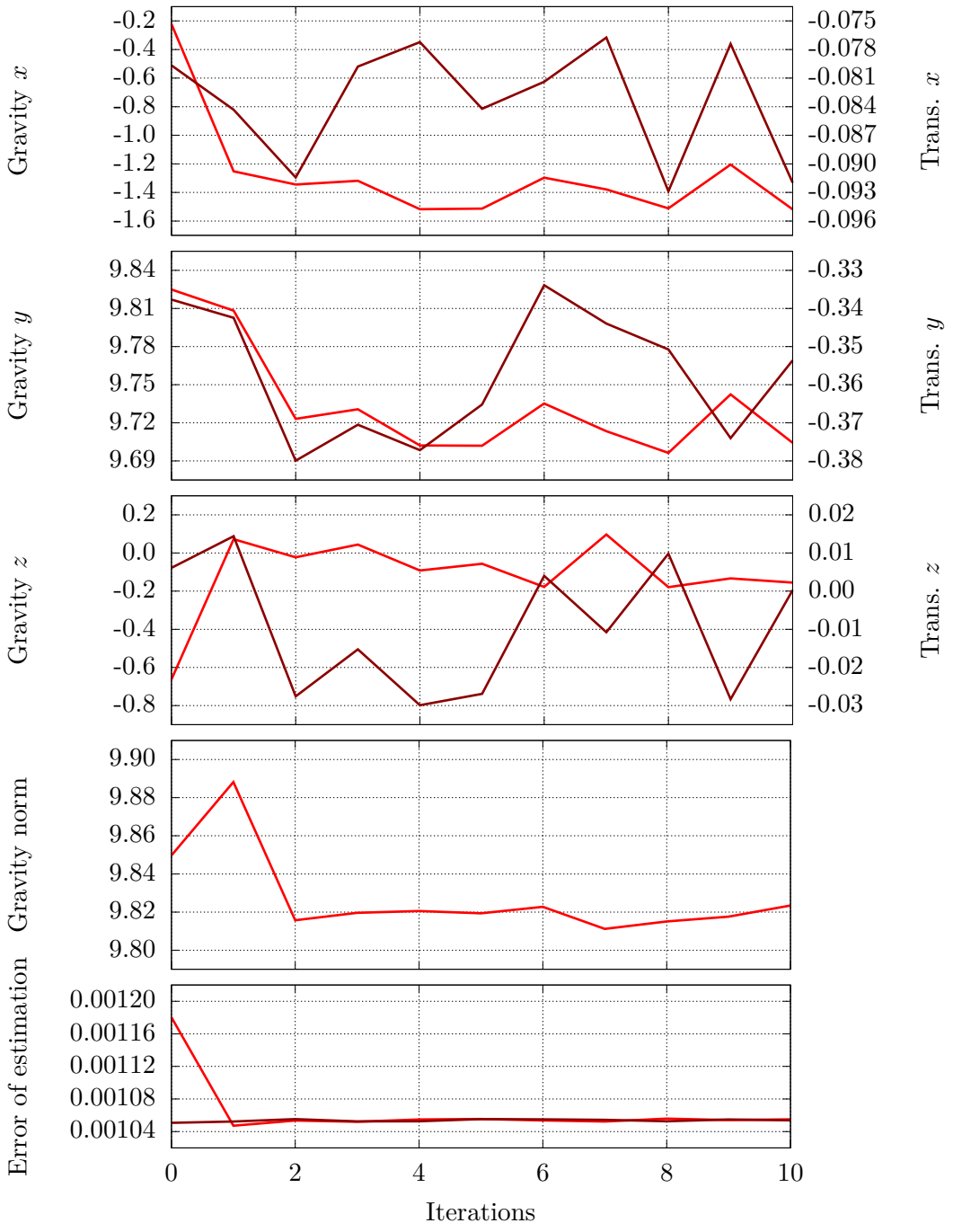

Figure 6.28: The plotted graphs show the result of the iterative process of alternating estimation of gravity in $\frac{\text{m}}{\text{s}^2}$ and translation in m for each iteration. Dataset ID 4 was used, where the left y-axis belong to the gravity (red) and the right y-axis to the translation (dark-red). The results shown here correspond to all blocks of Fig. 4.2.

ID	Source	It.	Grav. x	Grav. y	Grav. z	Grav. norm	Error
1	LM	7	−0.05398	5.9185	7.8242	9.8107	$6.2141E-6$
1	LM	0	0.1687	6.1744	7.5195	9.7310	$6.2353E-6$
1	Ceres	19	0.2953	6.2202	7.5799	9.8098	$6.3528E-6$
2	LM	14	−2.5709	9.4069	1.0820	9.8117	$1.8468E-5$
2	LM	0	−0.2857	8.9093	4.0270	9.8086	$2.0168E-5$
2	Ceres	18	−0.9253	10.5109	1.2345	10.6235	$1.9106E-5$
3	LM	4	−0.1507	9.7957	−0.5000	9.8096	$1.7113E-4$
3	LM	0	0.06204	9.8186	−0.5616	9.8357	$1.9166E-4$
3	Ceres	2	−0.1321	9.7943	−0.5365	9.8099	$1.7703E-4$
4	LM	1	−1.2518	9.8083	0.07264	9.8882	$1.0472E-3$
4	LM	0	−0.2214	9.8249	−0.6629	9.8511	$1.1802E-3$
4	Ceres	19	−2.3024	9.5088	0.7400	9.8115	$1.1061E-3$

Table 6.17: Results of the iterative process of alternating estimation of gravity, using the LM algorithm and the Ceres Solver. The iterations here indicate the point at which the error became minimal. Note that the results for zero iterations are taken from Table 6.13 and serve as a comparison if the iterative strategy is omitted.

ID	Source	Iteration	Trans. x	Trans. y	Trans. z	Error
1	LM	1	0.01917	0.06095	−0.01771	$6.2111E-6$
1	LM	0	0.01881	0.06127	−0.01758	$6.2132E-6$
1	Ceres	1	0.01630	0.06454	−0.01943	$6.3516E-6$
2	LM	1	−0.002038	0.06274	−0.04768	$1.8142E-5$
2	LM	0	−0.002038	0.06274	−0.04768	$1.8142E-5$
2	Ceres	15	0.008038	0.07480	−0.03636	$1.9098E-5$
3	LM	23	−0.04661	−0.01802	0.3530	$1.7126E-4$
3	LM	0	−0.04638	−0.01807	0.3527	$1.7142E-4$
3	Ceres	1	−0.04751	−0.01974	0.3562	$1.7686E-4$
4	LM	1	−0.07966	−0.3377	0.006108	$1.0508E-3$
4	LM	0	−0.07966	−0.3377	0.006108	$1.0508E-3$
4	Ceres	1	−0.07588	−0.3957	0.01145	$1.0755E-3$

Table 6.18: Results of the iterative process of alternating estimation of translation in m, using the LM algorithm and the Ceres Solver. The iterations here indicate the point at which the error became minimal. Note that the results for zero iterations are taken from Table 6.16 and serve as a comparison if the iterative strategy is omitted.

estimation. In addition, the results are also listed, which the Ceres Solver supplies, although it must be noted that the weighting method is not implemented.

Looking at the Fig. 6.25 - 6.28, it is noticeable that most of the values, especially the translation, fluctuate and do not seem to converge. The reason for this is not known, but the fluctuations in the translation are in a millimeter range and can be neglected. Especially for the bad configurations (ID 2 and 4) the gravity estimates also fluctuate strongly, which is either due to insufficient motions in the datasets or due to bad camera pose estimations of *VINS-Mono*.

Looking at the table 6.17, it is noticeable that in all cases an acceptable result is obtained for the norm of gravity. Especially for the good configurations (ID 1 and 3) the norm of gravity is estimated by the LM algorithm almost without errors. An exception is the result of the Ceres Solver for ID 2. There the result seems to be strongly falsified, whereby no cause is known. Also the error is lowered for each data set in comparison to not using the described iterative process, whereby the LM algorithm achieves a better result than the Ceres Solver in each case.

Considering the table 6.18, it is noticeable that also here by the LM algorithm better results are obtained than by the Ceres Solver. Except for the "bad" configurations, the error could be reduced by using the iterative process. With the result of the LM algorithm of ID 3 it is noticeable that the values for the translation if the iterative process is applied seem to deviate more from the ideal result, in comparison to it if the iterative process is omitted. Nevertheless, the estimation by the iterative process achieves a lower estimation error.

6.5.4 Summary

In the previous sections, for the estimation of the visual scale, the gravity, the translation between camera and IMU and the bias of the accelerometer, the individual steps of the structure seen in Fig. 4.2 have been evaluated. Hence, the estimation of the values with and without the use of the weighting strategy and finally the alternating estimation of the values.

In each step a reduction of the error is achieved, so that the values for the norm of gravity and the translation correspond close to the ideal results. Solely for ID 4 and ID 2, the iterative process of estimating the translation did not reduce the error (cf. Table 6.18). As already mentioned in Section 6.1, it is not possible here to determine the reasons for the remaining deviation from the ideal results. The reasons could be an error in the camera poses estimated by *VINS-Mono* or still existing problems in the minimization algorithms. It is also conceivable and most likely that the values for the ideal results (cf. Table 6.11 and 6.14) are incorrect. For example, for the *UWSensor* dataset it is implausible that an exact translation of $0.0m$ in x direction between the camera and the IMU applies, since it is mechanically impossible to align the devices that precisely. Accordingly, a translation of $-0.04661m$ (cf. Table 6.18 ID 3) might be closer to the real value.

Comparing the results for the translation of the software of this thesis with the results of *VINS-Mono* (cf. Table 6.14), it is noticeable that the results of *VINS-Mono* show a larger deviation in all cases compared to the results of the "good" configurations (ID 1 and 3). Only with the translation in x-direction for the *UWSensor* dataset *VINS-Mono* achieves an apparently better result, but as already said, in this case it cannot be clearly determined which result is closer to

the true value. Solely the effect of the calibration parameters on the pose estimation is evaluated based on reference data in the following chapter.

Chapter 7

Evaluation of Pose Estimation

The purpose of this chapter is to show how the extrinsic calibration parameters determined in the previous chapter improve the pose estimation by *VINS-Mono*. The EuRoC MAV dataset on the one hand and the *UWSensor* dataset on the other hand is evaluated.

7.1 Methodology

As already explained in the previous chapter, the true values for all calibration parameters are not known or only approximately known. Thus, only a qualitative statement about the results was made, i.e. whether the values seem plausible. In order to verify whether the methods for the estimation, which were presented in the context of this work, provide better results than the native calculation by *VINS-Mono*, the resulting trajectories, respectively point clouds are compared in this chapter.

For the EuRoC MAV, a trajectory is created using *VINS-Mono*'s own methods for estimating the calibration parameters and a comparison is made with the trajectory obtained using the results of this work for translation and rotation between camera and IMU. In this case, a ground truth trajectory exists with which the trajectories determined by *VINS-Mono* are compared.

To evaluate the *UWSensor* system, a dataset is used which was recorded underwater in the Baltic sea. In this case the trajectories are not compared, because no ground truth is known, instead the point clouds generated by *UWSensor* are compared, using reference targets placed in the scene.

7.2 EuRoC MAV

A ground truth trajectory exists for the flight of the EuRoC MAV. The flight of this drone was recorded with a laser tracking device. Thus it is possible to compare the trajectory estimated by *VINS-Mono* with this ground truth. At this point, the fact that the trajectory of the laser tracking device is defined in a different coordinate system must be taken into account. In order to

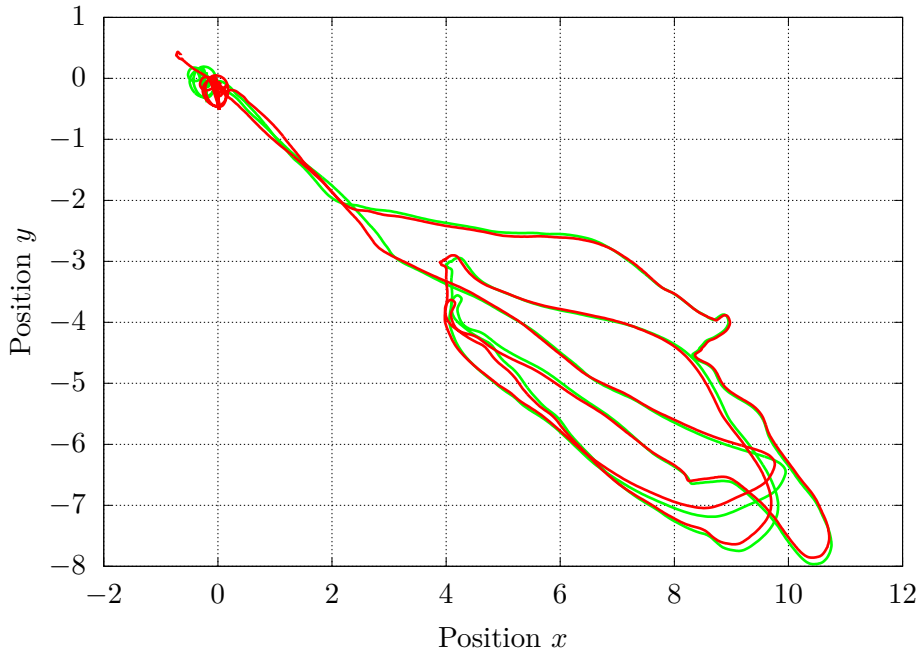

Figure 7.1: Shows the trajectory in x and y direction in m. For the sake of clarity, the z direction is omitted, since it barely changes during the flight. Here the green line represents the ground truth and the red line the resulting trajectory of *VINS-Mono* using the rotation and translation estimated in the course of this work. Start and end point are around the coordinates $(0, 0)$.

compare the two trajectories, a transformation must be found that transforms the trajectory of the ground truth into the coordinate system of *VINS-Mono*. To align the two trajectories to each other the script *evaluate_ate.py* from [1] is used. Both trajectories are aligned to each other using a least squares approach based on singular value decomposition. However, using this method, no correct quantitative statement can be made about the accuracy of the trajectory estimated by *VINS-Mono*, since it is not possible to estimate whether the cause for a deviation of the trajectories is due to an incorrect pose estimation or an incorrect transformation of the ground truth. Nevertheless, a qualitative statement is made about different trajectories of *VINS-Mono*, since it is assumed that the deviations from the ground truth become smaller, if the trajectory estimated by *VINS-Mono* is less error-prone.

In Fig. 7.1 the resulting trajectory is shown in red using the translation (cf. Table 6.18 ID 1) and rotation (cf. Table 6.9 ID 1) estimated in the course of this work. For comparison, the aligned ground truth is shown in green. Fig. 7.2 shows the same data, but only the beginning of the pose estimation (until time $42s$) to allow a more detailed representation. Here it is observed that the trajectories are shifted. This is due to the aforementioned problem that the two trajectories have to be registered against each other in order to compare the trajectory of *VINS-Mono* with the ground truth. The script *evaluate_ate.py* searches here for the optimal transformation, for which

[1] `https://vision.in.tum.de/data/datasets/rgbd-dataset/tools`

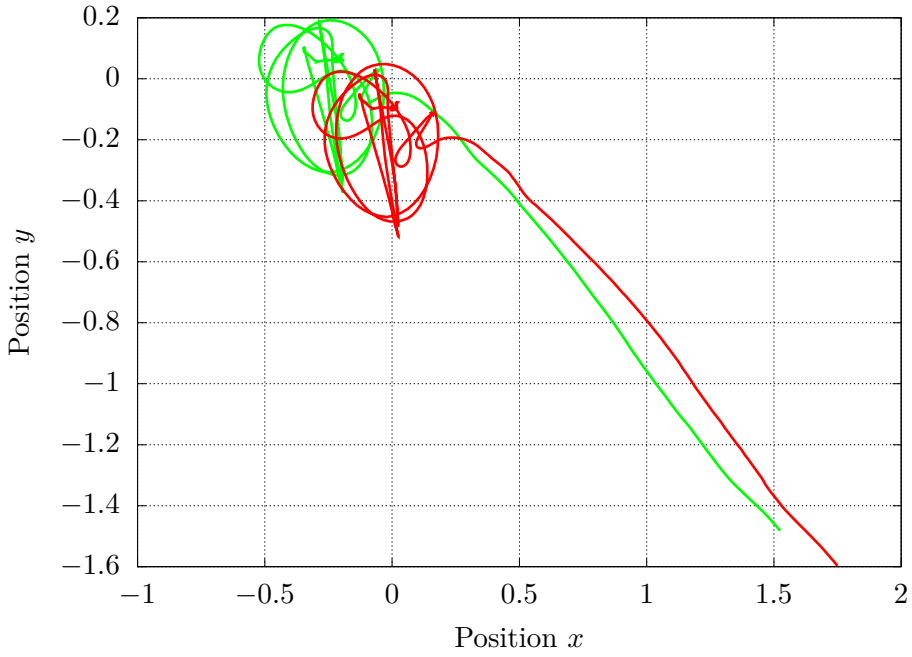

Figure 7.2: Shows the same position data as Fig 7.1. However, here only the trajectory up to time $42s$ after start to allow a more accurate representation of the beginning of the trajectory.

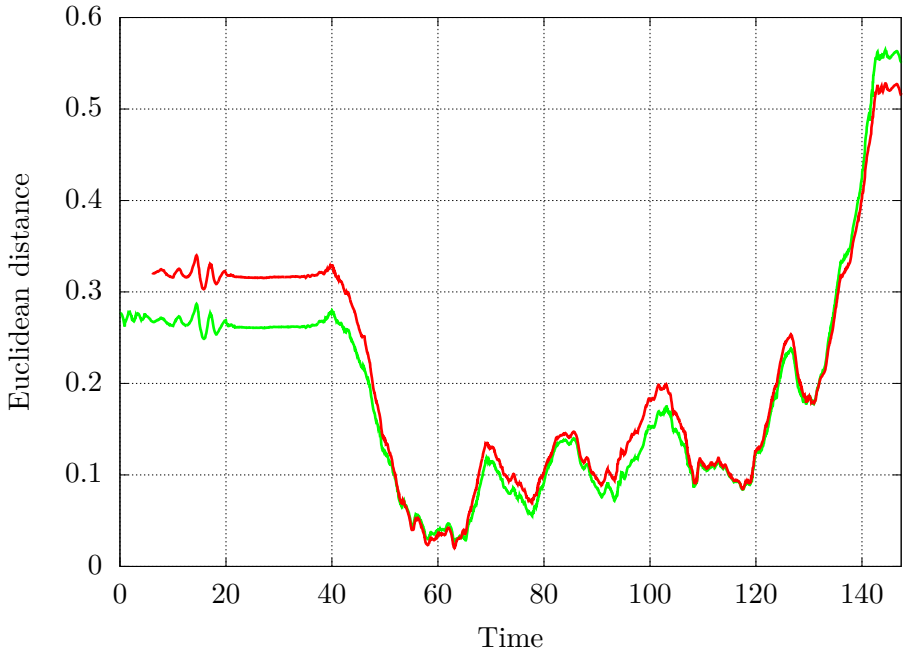

Figure 7.3: Euclidean distance to the ground truth. Where the red line represents the deviation of the trajectory from *VINS-Mono* and the green line represents the deviation of the trajectory using the rotation and translation estimated as part of this thesis. The distance is given in m and the time in s.

Profile ID	Length	Error
Initialization with methods of this thesis		
1	2.2029	−0.007115
2	2.2047	−0.005260
3	2.2167	0.006735
Initialization with native *VINS-Mono*		
1	2.2194	0.009369
2	2.2297	0.01972
3	2.2441	0.03411

Table 7.1: Sphere distances determined from the point clouds (cf. Fig. 7.6) using the native *VINS-Mono* and using the translation and rotation between the camera and IMU estimated as part of this thesis. The error was calculated from the deviation to the actual length of the profiles ($2.21m$). All units here are in m.

the deviations of the trajectories become minimal. Therefore, it cannot be clearly judged at this point whether the transformation is faulty or the trajectory of *VINS-Mono*. If we look at the complete trajectory (cf. Fig. 7.1), however, it seems rather as if the trajectory of *VINS-Mono*, at least at the beginning, is error-prone, since the trajectory was well registered from time $42s$ on and the mentioned shift is not observed here.

Fig. 7.3 shows the Euclidean distances to ground truth. The green line represents the distances of the trajectory using the parameters of this work and the red line the trajectory of the native calculation of *VINS-Mono*. Since *VINS-Mono* continuously optimizes the translation and rotation, especially the beginning (until about time $40s$) is relevant. Here it can be clearly seen that using the calibration methods of this work a smaller deviation to the ground truth is achieved. Also in the further course of the pose estimation the green line is mostly below the red line. The reason why the green line shows a larger deviation at the end of the trajectory cannot be clearly stated. It can be assumed that a not correct transformation between the trajectory and the ground truth is the cause, because in the other areas the pose estimation of the green trajectory is better than the red one.

7.3 *UWSensor*

Compared to EuRoC MAV, there is no ground truth of the trajectory traversed for the *UWSensor* dataset. However, since the *UWSensor* system has the possibility to create a point cloud due to the installed sensor technology, the quality of the resulting point cloud allows to compare the trajectories estimated by *VINS-Mono*. It is assumed that the less accurate the estimated trajectory is, the worse the quality of the point cloud calculated from it.

For the evaluation a dataset was used, which was recorded under water in the sea. Four aluminum profiles were scanned from a distance of about $1m$ over a length of about $10m$. The aluminum

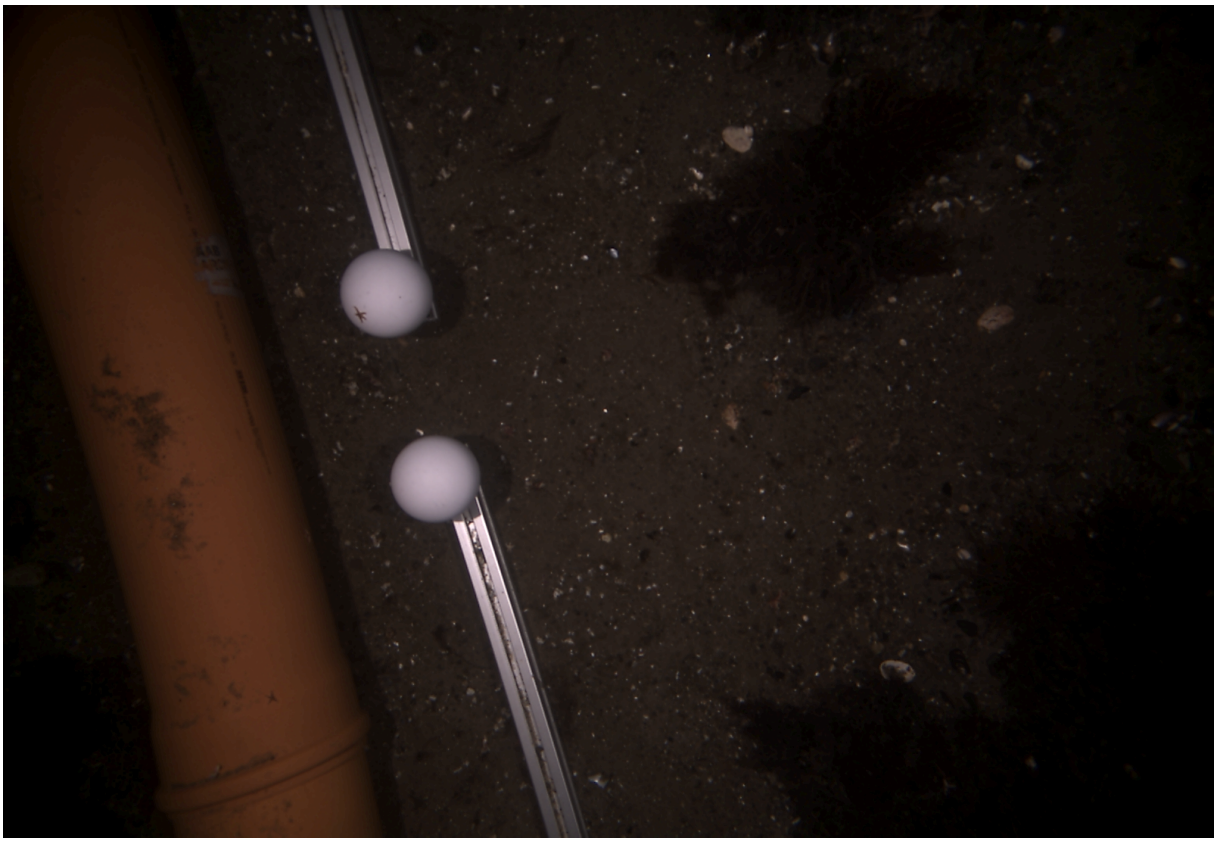

Figure 7.4: Snapshot of a data set from the *UWSensor* project during a trajectory under water. The spheres seen here are used in the evaluation of the resulting point cloud.

profiles have a length of $2.21m$ and have a sphere at each end, as shown in Fig. 7.4. To estimate the accuracy of the resulting point cloud, the lengths of the aluminum profiles determined by the point clouds are compared with the actual length of the profile.

Fig. 7.6 shows the two point clouds estimated using the trajectory of the native *VINS-Mono* (black) and using the translation and rotation between the camera and IMU estimated as part of this thesis (yellow). Here the scaling deviation is observable. The lengths of the profiles determined from these point clouds are shown in Table 7.1, with profiles 1-3 shown from top to bottom in Fig. 7.6. Note here that the first profile is ignored because the point clouds do not clearly show the first sphere. Table 7.1 shows that the deviations from the real profile length using the methods of this work are smaller than from the native *VINS-Mono*. In addition, it is noticeable that the deviations of the profile lengths increase continuously with the native *VINS-Mono*, which is due to an incorrectly estimated scale. Consequently, it is concluded that using the methods presented in this thesis to estimate the translation and rotation between camera and IMU, the scale is better estimated by *VINS-Mono*, leading to a better result in the estimated poses.

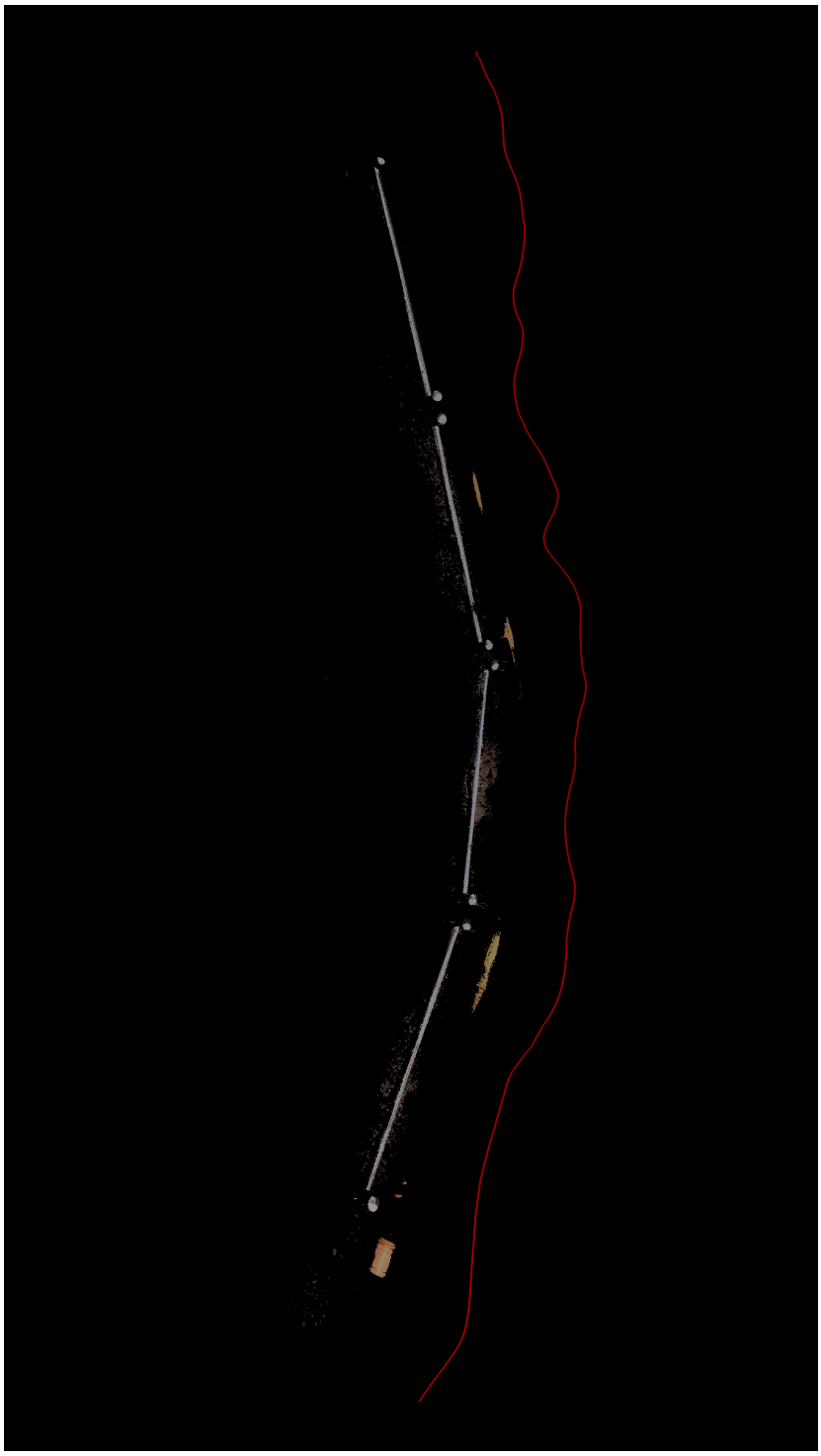

Figure 7.5: Point cloud generated by *UWSensor*, which was colored using the color camera data. *VINS-Mono* was used to estimate the poses, using the translation and rotation between the camera and IMU estimated as part of this thesis. The trajectory determined using this is shown as a red line. Areas which can be recognized as circles are the spheres which is seen in Fig. 7.4. The trajectory starts at the top of the figure and ends at the bottom.

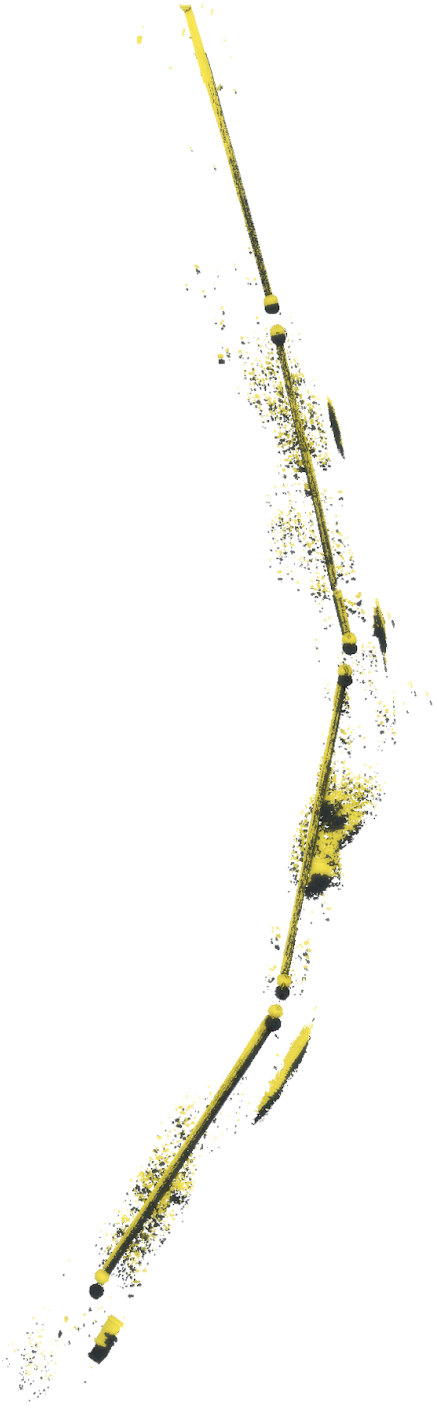

Figure 7.6: Shows the point clouds generated by *UWSensor*, which were estimated using the native *VINS-Mono* (black) and the resulting point cloud using the translation and rotation between the camera and IMU estimated as part of this thesis (yellow). Areas which can be recognized as circles are the spheres which is seen in Fig. 7.4. The trajectory starts at the top of the figure and ends at the bottom.

Chapter 8

Summary and Outlook

8.1 Résumé

To develop a sensor system that is capable of determining the poses of i.e. a mobile platform, it is convenient to use the sensors camera and IMU. These are, depending on the quality, inexpensive to purchase and are in terms of pose estimation only dependent on sufficient structure in the environment to find sufficient feature in the camera image. There are various VIO methods that process the data from the camera and IMU to determine the position and orientation of the sensor system. Since absolute visual scale information cannot be obtained from the image data with only one camera, each VIO method must determine it using the IMU data. In order to merge the data of two sensors, the sensor data must be available in the same coordinate system. Since the sensors are usually physically distant from each other and rotated relative to each other, a transformation must be found that transforms the data from one sensor coordinate system into the other. Hence there is a new problem, since the exact transformation is usually not known, due to the fact that it is impossible to align the sensors exactly to each other by hand. Accordingly, the transformation must be estimated using the data from the camera and the IMU. This thesis presents the necessary methods and algorithms.

First, methods and algorithms are presented in the course of this thesis, to calibrate an IMU. Error-free measurement values of the IMU are necessary, because later the distance traveled has to be determined from these data, therefore the acceleration values are integrated twice. A large measurement error would lead to an enormous deviation in the determined distance due to the double integration. The calibration methods for the accelerometer take advantage of the fact that it must experience the same gravitational force at standstill, regardless of its orientation to the earth. This is used to calculate the deterministic calibration factors. The calibration methods for the gyroscope use the calibrated values of the accelerometer to determine the orientation before and after a movement, the orientation determined by the gyroscope must match this. Using this, the deterministic calibration factors are calculated.

Next, methods and algorithms are presented that first determine the rotation between the camera and IMU using the poses determined by the camera and the pre-integrated readings from the

IMU. In the next step, the bias of the gyroscope can be determined. The subsequent alternating estimation refines the results. With the known rotation between the sensors the gravity vector in the camera coordinate system can be determined. Afterwards this is refined with the help of the known norm of the gravity on earth. In the same step, the visual scale and the translation between the camera and the IMU are determined. By then alternately estimating the gravity vector and translation, the results are refined. In this work, a weighting strategy is developed which optimally weights the sensor data such that the error of the estimates of the parameters just mentioned decreases.

Subsequently, the methods used to calibrate the IMU are evaluated. Hereby the measurement error, related to the accuracy of the gravity measurement, was reduced by a factor of 30. The measurement error of the gyroscope, related to the determination of the orientation, was halved.

In the following, the methods for determining the extrinsic calibration parameters between camera and IMU are evaluated. No correct quantitative evaluation is possible here, since no real values are known for the estimated values, to which these can be compared. However, it is shown that the estimates of the parameters using the methods from this thesis are at least better than those of the software *VINS-Mono*, which was used as VIO method in the course of this thesis. It is also shown that the estimated trajectory of the EuRoC MAV drone using *VINS-Mono* is more accurate when using the parameters for rotation and translation between the camera and the IMU, which were estimated in the course of this thesis. With the data from the *UWSensor* system, it is shown that the accuracy of the pointclouds determined using the estimated trajectory also improves when using the calibration methodology of this work for translation and rotation.

8.2 Future Work

Needless to say, a lot of work remains to be done. In the future, the two software systems developed in this work shall form a unified software. Therefore it should be possible to estimate the calibration parameters of the IMU as well as the extrinsic parameters between camera and IMU simultaneously. This has the advantage that all calibration parameters can be optimized jointly and as a result more accurate results can be expected. To achieve this, the equations for the estimation of the extrinsic parameters have to be extended, to allow the estimation of the scaling factors and the rotations of the individual axes of the accelerometer and the gyroscope via the presented minimization problems.

VINS-Mono has the disadvantage that only a few parameters can be adjusted regarding the camera pose estimation. As a result, many parameter configurations have to be tested before obtaining reliable values from *VINS-Mono*. Therefore, it is desirable that the software developed in this thesis is no longer dependent on *VINS-Mono*. Hence a strategy for the evaluation of the camera images has to be developed, which can reliably determine the rotation and translation between successive images. Furthermore, the pre-integration of the IMU measurement has to be implemented.

Bibliography

[1] Beschleunigungssensor. https://en.wikipedia.org/wiki/Accelerometer. Accessed on 14.01.2023.

[2] Einführung in das Robot Operating System. https://www.heise.de/developer/artikel/Einfuehrung-in-das-Robot-Operating-System-3273655.html?seite=all. Accessed on 14.01.2023.

[3] Inertial Measurement Unit. https://en.wikipedia.org/wiki/Inertial_measurement_unit. Accessed on 14.01.2023.

[4] MEMS accelerometer gyroscope magnetometer and arduino. https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/. Accessed on 14.01.2023.

[5] MEMS-Gyroskop für genaue Drehratenmessungen in rauen Umgebungen bei hohen Temperaturen. https://www.channel-e.de/designcorner/artikel/article/mems-gyroskop-fuer-genaue-drehratenmessungen-in-rauen-umgebungen-bei-hohen-temperaturen.html. Accessed on 14.01.2023.

[6] MEMS magnetic field sensor. https://en.wikipedia.org/wiki/MEMS_magnetic_field_sensor. Accessed on 14.01.2023.

[7] Robot Operating System. http://wiki.ros.org/de. Accessed on 14.01.2023.

[8] Robot Operating System. http://www.ros.org/about-ros/. Accessed on 14.01.2023.

[9] Robot Operating System. http://www.ros.org/history/. Accessed on 14.01.2023.

[10] Robot Operating System. https://www.generationrobots.com/blog/de/ros-robot-operating-system/. Accessed on 14.01.2023.

[11] Roboter Operating System. https://de.wikipedia.org/wiki/Robot_Operating_System. Accessed on 14.01.2023.

[12] IEEE standard specification format guide and test procedure for single-axis interferometric fiber optic gyros. *IEEE Std 952-1997*, pages 1–84, 1998.

[13] Sameer Agarwal, Keir Mierle, and the Ceres Solver Team. Ceres Solver, 3 2022.

[14] Michael S Andrle and John L Crassidis. Geometric integration of quaternions. *Journal of Guidance, Control, and Dynamics*, 36(6):1762–1767, 2013.

[15] Leslie Barreda Pupo. Characterization of errors and noises in MEMS inertial sensors using Allan variance method. Master's thesis, Universitat Politècnica de Catalunya, 2016.

[16] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016.

[17] Yung-Yu Chuang. Camera calibration. In *tech. rep.* Citeseer, 2005.

[18] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. Analysis and modeling of inertial sensors using Allan variance. *IEEE Transactions on instrumentation and measurement*, 57(1):140–149, 2007.

[19] Henri P Gavin. The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems. *Department of Civil and Environmental Engineering, Duke University*, 19, 2019.

[20] Jeroen D Hol, Thomas B Schon, and Fredrik Gustafsson. A new algorithm for calibrating a combined camera and IMU sensor unit. In *2008 10th International Conference on Control, Automation, Robotics and Vision*, pages 1857–1862. IEEE, 2008.

[21] Weibo Huang and Hong Liu. Online initialization and automatic camera-IMU extrinsic calibration for monocular visual-inertial SLAM. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5182–5189. IEEE, 2018.

[22] Christopher Jekeli. *Inertial navigation systems with geodetic applications*. de Gruyter, 2012.

[23] Roman Lesjak. Die Rolle einer inertialen Messeinheit in der Anwendung Moving-Base Gravimetry. 2010.

[24] Mark Looney. The basics of MEMS IMU/gyroscope alignment. *Analog Dialogue*, 49:1–6, 2015.

[25] Alaa Makdissi, François Vernotte, and Emeric De Clercq. Stability variances: A filter approach. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 57(5):1011–1028, 2010.

[26] Lawrence C Ng and Darryll J Pines. Characterization of ring laser gyro performance using the Allan variance method. *Journal of Guidance, Control, and Dynamics*, 20(1):211–214, 1997.

[27] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[28] Dengqing Tang, Tianjiang Hu, Lincheng Shen, Zhaowei Ma, and Congyu Pan. April-tag array-aided extrinsic calibration of camera-laser multi-sensor system. *Robotics and biomimetics*, 3(1):1–9, 2016.

[29] David Tedaldi, Alberto Pretto, and Emanuele Menegatti. A robust and easy to implement method for IMU calibration without external equipments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3042–3049. IEEE, 2014.

[30] G Terzakis, P Culverhouse, G Bugmann, et al. On quaternion based parametrization of orientation in computer vision and robotics. 2014.

[31] Eric Udd. An overview of fiber-optic sensors. *review of scientific instruments*, 66(8):4015–4030, 1995.

[32] Vladyslav Usenko. *Visual-inertial navigation for autonomous vehicles*. PhD thesis, Technische Universität München, 2019.

[33] Roberto G Valenti, Ivan Dryanovski, and Jizhong Xiao. Keeping a good attitude: A quaternion-based orientation filter for IMUs and MARGs. *Sensors*, 15(8):19302–19330, 2015.

[34] Vladimir Vukmirica, Ivana Trajkovski, and N Asanovic. Two methods for the determination of inertial sensor parameters. *Scientific Technical Review*, 3(1), 2018.

[35] Oliver J Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, Computer Laboratory, 2007.

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Würzburg, 10. February 2023