Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2017

Fusion of IMU and monocular-SLAM in a loosely coupled EKF

Henrik Fåhraeus

LINKÖPING UNIVERSITY

Master of Science Thesis in Electrical Engineering

Fusion of IMU and monocular-SLAM in a loosely coupled EKF

Henrik Fåhraeus

LiTH-ISY-EX--17/5033--SE

Supervisor: **Hanna Nyqvist**
ISY, Linköpings universitet

Examiner: **Gustaf Hendeby**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Sammanfattning

Kamerabaserad navigering är ett område som blir mer och mer populärt och är ofta hörnstenen i augmenterad och virtuell verklighet. Dock så är navigeringssystem som använder kamera mindre pålitlig under snabba rörelser och ofta resurskrävande vad gäller CPU- och batterianvändning. Bildbehandlingsalgoritmen introducerar också fördröjningar i systemet, vilket gör att informationen om den nuvarande positionen blir försenad.

Den här uppsatsen undersöker om en kamera och IMU kan fusioneras i ett löst kopplat EKF för att reducera dessa problem. En IMU introducerar omärkbara fördörjningar och prestandan försämras inte under snabba rörelser. För att en IMU ska kunna användas för noggrann navigering behövs en bra estimering av dess bias. Därför prövades en ny metod i ett kalibreringssteg för att se om det kunde öka prestandan. Även en metod att skatta den relativa positionen och orienteringen mellan kameran och IMU:n utvärderades.

Filtret visar upp lovande resultat vad gäller skattnigen av orienteringen. Filtret klarar av att skatta orienteringen utan märkbara fördröjningar och påverkas inte nämnvärt av snabba rörelser. Filtret har dock svårare att skatta positionen och ingen prestandaförbättring kunde ses vid användningen av IMU:n. Några metoder som troligtvis skulle förbättra prestandan diskuteras och föreslås som framtida arbete.

Abstract

Camera based navigation is getting more and more popular and is the often the cornerstone in Augmented and Virtual Reality. However, navigation systems using camera are less accurate during fast movements and the systems are often resource intensive in terms of CPU and battery consumption. Also, the image processing algorithms introduce latencies in the systems, causing the information of the current position to be delayed.

This thesis investigates if a camera and an IMU can be fused in a loosely coupled Extended Kalman Filter to reduce these problems. An IMU introduces unnoticeable latencies and the performance of the IMU is not affected by fast movements. For accurate tracking using an IMU it is important to estimate the bias correctly. Thus, a new method was used in a calibration step to see if it could improve the result. Also, a method to estimate the relative position and orientation between the camera and IMU is evaluated.

The filter shows promising results estimating the orientation. The filter can estimate the orientation without latencies and can also offer accurate tracking during fast rotation when the camera is not able to estimate the orientation. However, the position is much harder and no performance gain could be seen. Some methods that are likely to improve the tracking are discussed and suggested as future work.

Acknowledgments

Many thanks to my supervisor Hanna Nyqvist at Linköping University for all the guidance and inputs during the whole thesis. It has always felt easy to ask questions and you have always made a strong effort to answer them.

Also, I would like to thank Gustaf Hendeby for making this thesis possible.

Linköping, May 2017
Henrik Fåhraeus

Contents

Notation

Abbreviations

Abbreviation	Description
VR	Virtual Reality
IMU	Inertial Measurement Unit
MEMS	Micro-machined ElectroMechanical Systems
UWB	Ultra WideBand
EKF	Extended Kalman Filter
KF	Kalman Filter
PF	Particle Filter
MPF	Marginalized Particle Filter
UKF	Unscented Kalman Filter
PDF	Probability Density Function
SLAM	Simultaneous Localization And Mapping
A/D	Analog/Digital

1 Introduction

This master's thesis work was performed at a company that develops positioning and tracking for Virtual Reality (VR). The hardware they use is a headset and a cellphone to track and show a virtual world to the user. The company primarily uses a camera to track the user. The problem addressed in thesis is if an Inertial Measurement Unit (IMU) can be added to improve the tracking and reduce the CPU and battery consumption. This chapter include the background to the problem, the problem formulation and its limitations.

1.1 Background

The usage of IMUs for navigation has mostly been limited to the aviation and marine industry due to the cost and size [24]. However, during the last years the field of application has broaden. During the last decades the IMU has become less expensive with the introduction of micro-machined electromechanical systems (MEMS) technology allowing many consumer products such as cellphones, cameras and game consoles to include an IMU [24].

The research of navigation using cameras has a long history dating back to work such as [22]. The recent advances in computer processing has enabled image processing algorithms to be run in real time even on consumer products and together with the less expensive IMU, vision-aided inertial navigation has become a very popular field of research [41, 40].

An interesting application where IMU and cameras are applied is Virtual Reality (VR). In VR the idea is to show a virtual world which the user interacts with. One step in the real world should correspond to one step in the virtual world and rotating 180° should make the user see what just was behind their back. For this

purpose it is necessary to track the user's movements.

VR is maybe most famous as a gaming application but is today used in far more areas. One such area is the military where battlefield simulation allows engagement between soldiers in a safe environment to a lower cost than traditional training methods [2, 39]. The healthcare industry is another user of VR, where surgeries can be simulated without any danger for real patients [2, 39]. VR can also be very helpful in the construction industry. VR makes it possible to experience the buildings as they would appear in real life, reducing the errors in the completed building [2]. These are only three examples of industries that take advantages of the possibilities of VR, but there are a lot more areas such as education, sports and fashion where VR can have a big impact.

The technology behind VR faces different challenges but the biggest one is to avoid motion sickness caused by bad tracking and latencies in the system.

1.2 Problem description

The IMU measures acceleration and angular velocity and the measurements can be integrated over time to obtain a position and orientation. The noise inherent in the IMU's measurements are also included in the integration and will cause the estimates of the position and orientation to drift away from the true value. An IMU can be sampled with a very high sampling frequency and sense fast motions very well, but because of the drift it can only be used without an aiding sensor during shorter periods of time.

A camera can estimate the pose (position and orientation) accurately during longer periods of time under slow motion but suffers heavily from motion blur and rolling shutter effects during fast motion.

An image does not represent a single instant of time, but the scene during the exposure time. Motion blur is observed if the scene changes during the exposure time of the camera. This effect is more apparent during fast motion since the scene is then changed more and the image will be blurred. This can be seen in Figure 1.1a where a bus is moving and a telephone booth is not.

Global shutter and rolling shutter are two methods used for image capturing. A global shutter camera captures the whole scene at the same time whilst rolling shutter camera scans the scene vertically or horizontally. This means that for rolling shutter cameras the whole scene is not recorded at the same time. If the scene is scanned horizontally starting from the top, the bottom of the scene is scanned a little bit later and might have changed from the time when the camera started to scan the top. In Figure 1.1b straight rotor blades of a helicopter seems

to be curved as a result of this effect.

The images from a camera need to be processed by an image processing algorithm to obtain a position. The image processing takes time and limits the frequency at which positions can be obtained from a camera. The time spent in the algorithm will also introduce latencies in the system. As a result the position returned from the algorithm does not correspond to the current time but instead to the time when the image was captured. An IMU can often often be sampled in 200 Hz while a camera often is sampled in 30 Hz. The low frequency of the camera can make the VR experience jerky if a faster sensor is not used together with the camera.

Another complication, when creating a virtual world in this way, is that the images from a camera is a projection of a 3D scene onto a 2D plane. The information of the depth in the scene is lost when projecting onto a 2D plane and that is why a camera can only be used to estimate the pose up to a scale. The IMU however, returns the acceleration in known units (m/s^2) and can therefore be used to estimate the unknown scale.These are reasons why a camera and an IMU might preferably be combined in order to get a more accurate tracking of both slow and fast motions, without latencies.

The company's tracking algorithm, does today mostly rely on a monocular camera for pose tracking and the image processing is resource intensive in terms of CPU and battery consumption. Also, a rolling shutter camera is used to capture images leading to worse performance of the tracking algorithm during fast movements causing the user to be motion sick. Since the IMU can sense fast motions better than cameras, a combination of the two sensors might be more suitable. Relying more on the IMU can also free resources, increase battery time and avoid overheating. This thesis will investigate if such an assumption is true.

(a) *Motion blur effects of a driving bus [45]*

(b) *Rolling shutter effects for rotor blades of a helicopter [44]*

Figure 1.1: *Illustration of common image artifacts*

1.3 Purpose of this thesis

To track the user and show a virtual world instead of the real one, the company normally uses the headset Samsung GearVR together with a phone. However, in this thesis tracking for a hand held phone is considered and seen as the sensor carrier instead of the headset.

The purpose of this thesis is to

- investigate if the sensors camera and IMU in the phone can be used together to accurately estimate position and orientation of fast and slow motion, and
- evaluate if fusion of camera and IMU information allows for the frame rate of the camera to be lowered to reduce the CPU and battery consumption

1.4 Limitations

Many systems for VR uses external sensors. External sensors are sensors that can not be worn by the user and have to be placed in the user's environment. An example is HTC Vive which uses infrared cameras that are placed in the room [1]. The company's idea is to only use a headset and a cellphone, making the system portable. In this report only the IMU and camera in the phone will be used for tracking.

All image processing in this work was done by an algorithm called ORB-SLAM [37]. The algorithm will not be investigated and will be considered a black box. The algorithm supplies poses of the camera and can be fused with measurements from the IMU to track the phone.

1.5 Thesis outline

This Master's thesis is structured as,

- Chapter 2 explains all the theory used in this thesis.
- Chapter 3 explains different calibration steps and tests to evaluate the methods used.
- Chapter 4 presents the results from the different calibrations and tests along with discussions.
- Chapter 5 presents the conclusions and the author's recommendations for future work.

2

Filtering with Camera and IMU

This chapter will start by introducing general filter theory that leads to the the chosen filter. The chosen filter will then be more thoroughly described in Section 2.2. This is followed by models of the system's dynamics and sensors, that are used in the filter. The models introduce a couple of parameters that need to be estimated and in Section 2.4 some methods to determine these parameters are explained.

2.1 Filter Theory

The purpose of a filter is to continuously estimate quantities of interest, for example position, velocity and orientation, given noisy measurements from different sensors. The quantities of interest will from now on be called the states. In this work a filter is a computer program that takes measurements as inputs and calculates states. Common sensors for indoor tracking are accelerometer, gyroscope, odometer, UWB, Wifi and camera. The system's dynamics, which describes how the states changes over time, is often described by a motion model. The filter's task is to fuse the measurements from the sensors and the information from the motion model in the best possible way to accurately estimate the states. The problem to accurately fuse information from several sensors and a motion model, often called sensor fusion, is to figure out how much to trust the different sensors and the motion model.

The approaches to fuse measurements from different sensors can be categorized in two categories, loosely and tightly coupled. In loosely coupled approaches the measurements from one or more sensors are pre-processed before they are fed to a filter that estimates the states, see Figure 2.1. In tightly coupled solutions the measurements are directly used in a filter without any kind of pre-processing.

As an example we can consider a filter fusing camera and IMU measurements. The measurements can either be sent directly to the filter or the camera measurements can be pre-processed by an image processing algorithm that estimates the pose (position and orientation) of the camera, which then is used as a measurement in the filter. In tightly coupled solutions the correlations between all measurements can be taken into account, leading to approaches that are computational expensive [41, 31], but more accurate. However, if the correct probability density function is returned from the pre-processing algorithm, the loosely and tightly coupled solution become equivalent [24]. In summary the tightly coupled solution is often more computational complex but more accurate since it uses all the information about the measurements.

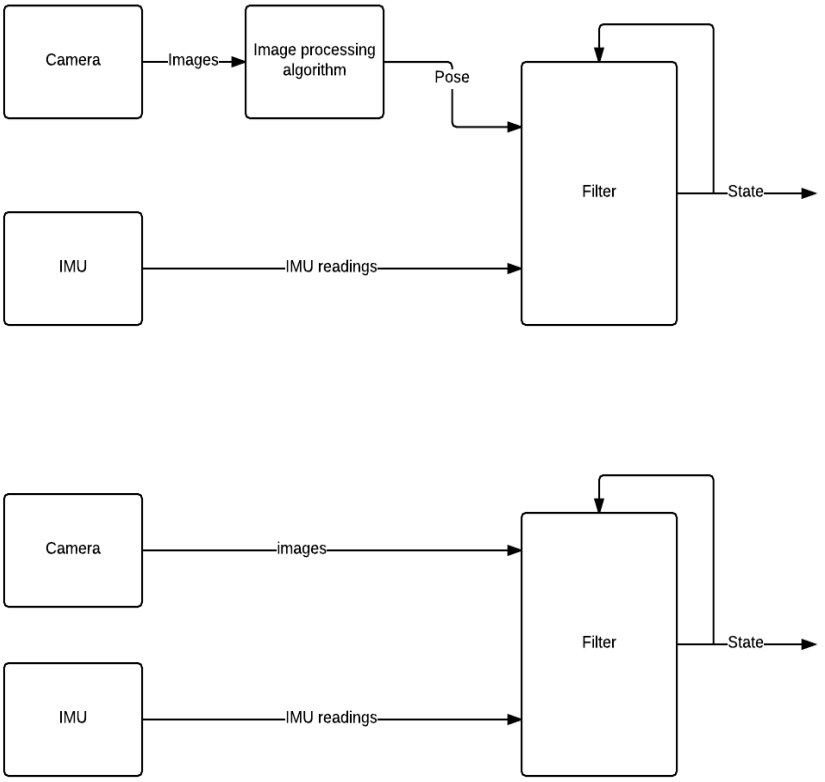


Figure 2.1: *Loosely (above) and tightly (beneath) coupled solutions*

A common estimation method is the Extended Kalman Filter (EKF) [21] which

is a nonlinear extension of the famous Kalman Filter (KF) [29]. Some references that used the EKF to successfully fuse camera and IMU are [24], [41] and [8] . Another common method to fuse different sensors is the Unscented Kalman Filter (UKF)[28]. In [27] and [4] a comparison of the UKF and EKF has been done. The comparison shows that the accuracy is roughly the same for head and hand orientation and the conclusion is that the computational overhead of the UKF makes EKF the better choice for Virtual Reality applications.

Particle Filter (PF) [5] is another popular filter for tracking. It applies to both nonlinear and non-Gaussian models. The main limiting factor is the computational complexity of the algorithm. For most problems some of the states are modelled as linear and Gaussian and then a more efficient way is to use the Marginalized Particle Filter (MPF), which is a combination of the PF and KF. The idea is to marginalize out the states that are linear with Gaussian noise and use a KF for these states [38]. [9] uses a modified MPF to fuse camera and IMU and compares to an EKF. To keep the dimension of the state vector low, to reduce the computational complexity, acceleration and angular velocity were considered as control inputs. The bias of the accelerometer were hardly observable among other errors such as model errors and [9] chose to only include the gyroscope bias in the states. The reduction of the dimension of the state vector lead to real-time performance. The modified MPF showed performance gains compared to the traditional MPF and PF. However, compared to the EKF, the MPF showed no performance gain and a higher computational cost.

The company's solution is today already resource intensive. Given the information that tightly coupled solutions are more computational complex, a loosely coupled filter was chosen. Given that EKF is the least computationally complex filter and has been shown to work well in similar cases, a loosely coupled EKF was chosen. The filter will fuse the information from ORB-SLAM with the measurements from the IMU.

2.2 Extended Kalman Filter

This section will explain the steps in the Extended Kalman Filter. To be able to use the theory of Extended Kalman Filter the system has to be modelled by a state-space model on the form,

$$x_{t+1} = f(x_t, u_t, v_t, T), \tag{2.1a}$$

$$y_t = h(x_t, u_t, e_t), \tag{2.1b}$$

where x_t is the state at time t, u_t is input to the system at time t, y_t is the measurement at time t, T is the sampling interval, v_t is referred to as process noise and e_t is referred to as measurement noise. (2.1a) is often called motion model and (2.1b) is often called measurement model.

In the derivation of the Kalman Filter both $f(x_t, u_t, v_t, T)$ and $h(x_t, u_t, e_t)$ must

be linear and the noise v and e are assumed to be zero mean multivariate Gaussian noises. The Kalman Filter estimates the probability density function (pdf) of the states. Since the system is assumed to be linear, the states x_t and the measurements y_t are all a linear combination of x_0, v_t and e_t. If x_0, v_t and e_t are assumed to be independent and have a Gaussian distribution, then x_t and y_t will be jointly Gaussian. A Gaussian distribution is defined by its expected values and its covaraince matrix. The Kalman Filter estimates the expected values, $\hat{x}$, and the covariance matrix of the states, P. (2.1a) is used to predict the pdf of the states and (2.1b) is used to correct the pdf given new measurements, using the theory of conditional distribution.

If the state-space model instead is nonlinear, an EKF can be used. In a nonlinear model the states and measurements do not need to be a linear combination of x_0, v_t and e_t and as a consequence x_t and y_t are not guaranteed to be jointly Gaussian. However, this is overlooked and the pdf in the EKF is still approximated with a Gaussian distribution. The idea of the EKF is to approximate the state-space model defined by (2.1) by a first order Taylor expansion and then apply the theory of the Kalman Filter [29]. The EKF can hence be implemented as a time update which predicts $\hat{x}$ and P and a measurement update that corrects them, just like the KF. The two updates will be described in more detail in the following two sections.

2.2.1 Time Update

In the time update the states and inputs are propagated through the motion model (2.1a). The covariance of the states, P, is updated by adding uncertainty because of the uncertainty of the current states and the process noise v. The time update can be summarized by,

$$\hat{x}_{t+1|t} = f(\hat{x}_{t|t}, u_t, \hat{v}_t, T), \tag{2.2a}$$

$$P_{t+1|t} = F_t P_{t|t} F_t^T + L_t Q_t L_t^T, \tag{2.2b}$$

where

$$F_t = \frac{\partial f}{\partial x}\Big|_{\hat{x}_{t|t}, u_t, \hat{v}_t, T}, \tag{2.3a}$$

$$L_t = \frac{\partial f}{\partial v}\Big|_{\hat{x}_{t|t}, u_t, \hat{v}_t, T}. \tag{2.3b}$$

Q_t is the covariance of v_t, T is the sampling interval which might be nonuniform and $\hat{v}_t$ is the expected value of the process noise. Q_t models the errors in the model and describes how much the motion model can be trusted. Q_t is a tuning parameter and it is important that it is tuned correctly in order for the EKF to work well. $\hat{x}_{t+1|t}$ is the predicted state of time $t+1$ given measurements up to time t and $\hat{x}_{t|t}$ is the filtered state of time t given measurements up to time t. As mentioned earlier in Section 2.2, the KF estimates the distribution of the states and the Time Update predicts the pdf of the states for the next time step. If the

state distribution at time t and given measurements up to time, $x_{t|t}$, is distributed as

$$x_{t|t} \sim \mathcal{N}(\hat{x}_{t|t}, P_{t|t}),$$

then the distribution of the prediction $x_{t+1|t}$ is

$$x_{t+1|t} \sim \mathcal{N}(\hat{x}_{t+1|t}, P_{t+1|t}),$$

with the expected value $\hat{x}_{t+1|t}$ and covariance matrix $P_{t+1|t}$ defined by (2.2).

2.2.2 Measurement Update

The measurement update compares the observed measurement y_t with the predicted measurement $h(\hat{x}_{t|t-1})$ and updates the state proportional to the prediction error. The information from the new measurement also decreases the covariance of the state estimate. The measurement update can be summarized by,

$$K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + M_t R_t M_t^T)^{-1}, \tag{2.4a}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - h(\hat{x}_{t|t-1}, u_k, \hat{e}_t)), \tag{2.4b}$$

$$P_{t|t} = (I - K_t H_t) P_{t|t-1}, \tag{2.4c}$$

where

$$H_t = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{t|t-1}, u_t, \hat{e}_t}, \tag{2.5}$$

$$M_t = \left.\frac{\partial h}{\partial e}\right|_{\hat{x}_{t|t-1}, u_t, \hat{e}_t}. \tag{2.6}$$

R_t is the covariance of e_t and $\hat{e}_t$ is the expected value of the measurement noise. R_t should describe the uncertainty of the measurements and tells how much the measurements can be trusted. R_t is a tuning parameter just as Q_t and it is also important that it is tuned correctly in order for the EKF to work well.

For the EKF to be well defined, the filter has to be initiated with an initial guess of $\hat{x}_{0|0}$ and $P_{0|0}$.

2.3 System Modelling

This section describes the system and how it is modelled. The system is composed by Samsung Galaxy S6 with an IMU and a camera, ORB-SLAM and an EKF. ORB-SLAM can both estimate a pose of the camera and a map of the environment, and will in the following text be referred as SLAM (Simultaneous Localization And Mapping). The SLAM is modelled as a black box that returns the pose of the camera. The SLAM is used as a pre-processing step before the EKF, see Figure 2.1. To use the Extended Kalman Filter for the system, the state-space model (2.1) has to be specified. The state-space model should both model the dynamics of the system and the relationship between the states and the measurements. To talk about the system's state-space model, different coordinate systems have to be introduced which are described in the next section.

2.3.1 Coordinate systems

This section will describe the coordinate systems and how they are related to eachother.

There are four coordinate systems, the SLAM system S, the Navigation system N, the Camera system C and the IMU system I. S and N are both fixed with respect to the earth. C is rigidly attached to the camera and I is rigidly attached to the IMU. Since both C and I are rigidly attached to the phone the relative pose between the camera and IMU is fixed and they are only moved with respect to S and N.

S and C are needed because SLAM provides measurements of the pose of C relative to S. I is needed because all the IMU measurements are resolved in this system. The origin and orientation of S is decided by the SLAM algorithm and the orientation relative gravity is unknown and therefore N is needed. N is vertically aligned which will be important in a later stage when using the EKF to easily subtract the gravity. The states in the EKF are also expressed in system N. To summarize, the coordinate systems used are,

- **SLAM (S)**: The camera pose (position and orientation) is estimated relative this coordinate system. The coordinate system is fixed with respect to the earth and the origin and orientation is decided by the SLAM algorithm
- **Navigation (N)**: The filter uses this coordinate systems for tracking. The coordinate system is fixed with respect to the earth and is vertically aligned, to easily subtract gravity from one axis.
- **IMU (I)** This coordinate system is rigidly attached to the IMU and all IMU measurements are resolved in this coordinate system. The x-axis of I is to the right of the phone, the y-axis is pointed upwards of phone and the z-axis is pointed backwards of the phone seen in Figure 2.2
- **Camera (C)** This coordinate system is rigidly attached to the camera. The x-axis of C is to the right of the phone, the y-axis is pointed downwards of phone and the z-axis is pointed along the optical axis which is approximately forward of the phone seen in Figure 2.2

A vector in one coordinate system can be expressed in another by rotating it by a rotation matrix R , defined by the relative rotation between the coordinate systems and translating it by a vector t, the relative translation between the coordinate systems' origins. Coordinates of a point x are related in the following way:

$$x_A = R^{AB} x_B + t_A^{B/A}, \tag{2.7}$$

where x_A is a point resolved in coordinate system A and R^{AB} is the rotation matrix rotating a vector from coordinate system B to A and $t_A^{B/A}$ is the vector from A to B resolved in system A. This notation will be used through out this thesis.

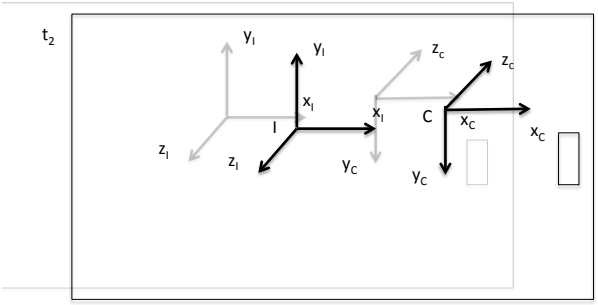


Figure 2.2: *The four coordinate systems. The phone is showed at time instances t_1 and t_2. Coordinate systems I and C are moved along with the phone since they are rigidly attached to it. The coordinate systems N and S are fixed and have not moved from t_1 to t_2. The image also illustrates that N is vertically aligned, which S does not need to be. An example how to go from system N to S is showed.*

2.3.2 Quaternion Representation

The rotation matrix R discussed in the section above is one way of describing rotations. Two other ways of representing rotations are Euler angles and quaternions. The Euler angle representation suffer from singularities, called gimbal lock and is avoided by using quaternions [12].

A quaternion is an extension of the complex numbers and is commonly used in navigation applications to represent rotation as an alternative to the traditional Euler angles. A quaternion is represented by a complex number $q_0+q_1i+q_2j+q_3k$ or a quadtuple (q_0, q_1, q_2, q_3) with the identities, $i^2 = j^2 = k^2 = ijk = -1$. q_0 is often called the real part and q_1, q_2, q_3 the imaginary part of the quaternion. For a quaternion to represent a rotation in $3D$ the vector has to be constrained to length 1.

Any rotation can be represented by a unit vector $u = \left(u_x, u_y, u_z\right)$ and a scalar θ according to Euler's rotation theorem, and the quaternion representing this rotation is [10],

$$q = \cos\left(\frac{\theta}{2}\right) + (u_x i + u_y j + u_z k)\sin\left(\frac{\theta}{2}\right). \tag{2.8}$$

The same rotation can be achieved by rotating $2\pi - \theta$ around $-u$ and therefore q and $-q$ represent the same rotation. The corresponding rotation matrix R can be calculated from the quaternion as,

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_0q_2 + 2q_1q_3 \\ 2q_0q_3 + 2q_1q_2 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & -2q_0q_1 + 2q_2q_3 \\ -2q_0q_2 + 2q_1q_3 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}. \tag{2.9}$$

The derivative of a unit quaternion can be expressed as,

$$\frac{dq^{NI}}{dt} = q^{NI} \odot \frac{1}{2}\widetilde{\omega}_I^{NI}, \tag{2.10}$$

where q^{NI} represents the rotation of the I-frame relative to the N-frame and $\widetilde{\omega}_I^{NI}$ is composed of the angular velocity, ω_I^{NI}, as $\left(0, \omega_I^{NI}\right)$ [24, 10]. $\odot$ represents quaternion multiplication [10, 24]. Using Zero Order Hold and small angle approximation, (2.10) can in discrete time be expressed as [21],

$$q_{t+T}^{NI} = \left(I_4 + \frac{T}{2} S\left(\omega_{t,I}^{NI}\right)\right) q_t^{NI}, \tag{2.11}$$

where

$$S(\omega) = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix}. \tag{2.12}$$

It is often of interest to know the relative rotation between two quaternions q^{10} and q^{20}. The rotation from q^{10} to q^{20} can be described by a third quaternion by,

$$q^{21} = q^{20} \odot q^{-10}, \tag{2.13}$$

where q^{-10} is the inverse of q^{10}, which in case of working with unit quaternions is the same as the conjugate [10]. The conjugation of a quaternion is done, as for complex number, by changing the sign of the imaginary part. The angle θ^{21} of q^{21} could be used as a measure of the distance between q^{10} and q^{20}. The angle expresses how much q^{10} has to be rotated around an axis to coincide with q^{20}. The angle can be solved for using (2.8). This does not guarantee the smallest angle since $-q^{21}$ represent the same rotation. The quaternion with a positive real part will always represent the smallest angle leading to the following expression,

$$\theta_{min}^{21} = 2 \arccos\left(\left|q_0^{21}\right|\right). \tag{2.14}$$

2.3.3 Accelerometer

The accelerometer is the sensor in the IMU that measures acceleration. To be able to specify the motion model of the system (2.1a), the measurement model of the accelerometer needs to be specified to relate the acceleration of the body to the measurements of the accelerometer. The accelerometer used in this thesis is a MEMS accelerometer. Low budget MEMS accelerometers suffer from different errors and the errors that will be accounted for in this thesis are measurement noise and bias. The model used in this thesis is the same as in [24, 42]. At time

t the accelerometer measures y_t^a and is related to the the acceleration of the IMU in the N system, $\ddot{p}_{t,N}^{I/N}$, as

$$y_t^a = R^{IN}(\ddot{p}_{t,N}^{I/N} - g_N) + \delta_{t,I}^a + v_{t,I}^a, \tag{2.15}$$

where $g_N = [\,0\ -9.82\ 0\,]$ is the gravity and $v_{t,I}^a$ is the measurement noise, modelled as white Gaussian noise. δ_t^a is the bias and is modelled as random walk,

$$\delta_{t+1,I}^a = \delta_{t,I}^a + v_{t,I}^{\delta_a}, \tag{2.16}$$

where $v_{t,I}^{\delta_a}$ also is modelled as white Gaussian noise.

2.3.4 Gyroscope

The gyroscope is the sensor in the IMU that measures angular velocity. The gyroscope used in this thesis is a MEMS gyroscope and suffer, like the accelerometer, from errors. The measurement models are similar to the accelerometer's model with bias and measurement noise. The gyroscope measures angular velocity and will also sense the earth's angular velocity. The earth's angular velocity is small compared to the noise of the gyroscope and can therefore be neglected which gives the following measurement model similar to [24],

$$y_t^\omega = \omega_{t,I}^{NI} + \delta_{t,I}^\omega + v_{t,I}^\omega, \tag{2.17}$$

where $\omega_{t,I}^{NI}$ is the angular velocity of the gyroscope observed from the N-frame and resolved in the I-frame, $v_{t,I}^\omega$ is modelled as white Gaussian noise and the bias δ_t^ω is modelled as random walk,

$$\delta_{t+1,I}^\omega = \delta_{t,I}^\omega + v_{t,I}^{\delta_\omega}, \tag{2.18}$$

where $v_{t,I}^{\delta_\omega}$ also is modelled as white Gaussian noise.

2.3.5 Pose measurements from SLAM algorithm

SLAM uses images from a camera to calculate the camera's pose and to better understand how this can work some general theory of image processing is introduced [13, 6]. However, a full description of the SLAM algorithm will not be presented since it is modelled as a black box and is outside the scope of this Master's thesis.

An image processing algorithm detects features in the image and associate them to 3D points in the world. Features are specific structures in the image, e.g. points, edges or objects. The correspondences between the features in the image and the 3D points called landmarks in the world can then be used to calculate the pose. To express the correspondence a new coordinate system i is introduced to express the feature in the image plane. The correspondences can be expressed as,

$$p_{t,i}^{m/i} = P(p_{t,C}^{m/C}), \tag{2.19}$$

where $p_{t,i}^{m/i}$ is the 2D coordinates of feature m in the image at time t, $p_{t,C}^{m/C}$ is the corresponding landmark resolved in frame C and P is the projection function relating the two points. An illustration of the projection is seen in Figure 2.3. P can for example be the famous pinhole model which relates the point $p_{t,i}^{m/i} = (u, v)$ to $p_{t,C}^{m/C} = (X, Y, Z)$ according to,

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{2.20}$$

where f is the focal length of the camera [24]. If the 3D points are known in the

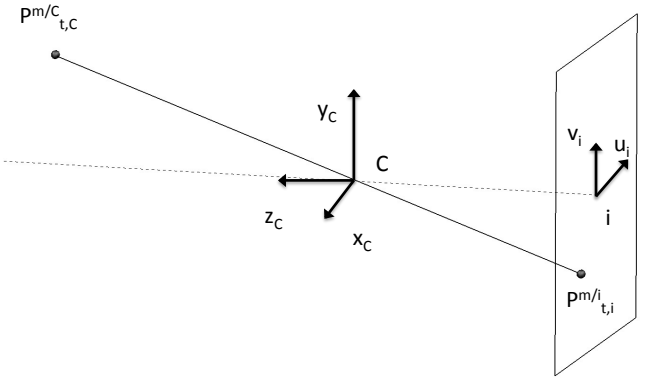


Figure 2.3: *Camera projection*

SLAM frame S, $p_{t,S}^{m/S}$, (2.19) can instead be formulated as

$$p_{t,i}^{m/i} = P\left(R^{CS}(q_t^{CS})p_S^{m/S} + t_{t,C}^{S/C}\right), \tag{2.21}$$

where q_t^{CS} is the orientation of the camera at time t and $t_{t,C}^{S/C}$ is the position of the camera at time t resolved in C. The landmark $p_{t,S}^{m/S}$ is the same independently of which image the point is observed from since S is the same for all images. Thus it is possible to drop index t, which is not the case if the landmark is expressed in C.

If M features are observed in T images at times $(t_1 \cdots t_T)$, the poses of the camera can be estimated by,

$$\hat{q}^{CS}, \hat{t}_C^{S/C} = \underset{q_t^{CS}, t_{t,C}^{S/C}}{\arg\min} \sum_{m=1}^{M} \sum_{t=t_1}^{t_T} v_{m,t} \| p_{t,i}^{m/i} - P\left(R^{CS}(q_t^{CS})p_S^{m/S} + t_{t,C}^{S/C}\right) \|, \tag{2.22}$$

where $v_{t,m}$ is 1 if feature m is seen in image t otherwise it is 0. $\hat{q}^{CS}$ and $\hat{t}_C^{S/C}$ are vectors containing the orientations and translations of the camera of each image, $\hat{q}^{CS} = [q_{t_1}^{CS} \cdots q_{t_T}^{CS}]$ respectively $\hat{t}_C^{S/C} = [t_{t_1,C}^{S/C} \cdots t_{t_T,C}^{S/C}]$. If only the last pose for

image frame T is of interest the problem is instead formulated as,

$$\hat{q}_{t_T}^{CS}, \hat{t}_{t_T,C}^{S/C} = \underset{q_{t_T}^{CS}, t_{t_T,C}^{S/C}}{\arg\min} \sum_{m=1}^{M} v_{m,t_T} \| p_{t_T,i}^{m/i} - P\left(R^{CS}(q_{t_T}^{CS}) p_S^{m/S} + t_{t_T,C}^{S/C}\right) \|. \tag{2.23}$$

Often when simultaneously tracking the camera and building a map, (2.22) is optimized over the landmarks as well. This optimization problem is often called bundle adjustment and the optimization problem instead looks like,

$$\hat{p}_S, \hat{q}^{CS}, \hat{t}_C^{S/C} = \underset{p_S^m, q_t^{CS}, t_{t,C}^{S/C}}{\arg\min} \sum_{m=1}^{M} \sum_{t=t_1}^{t_T} v_{m,t} \| p_{t,i}^{m/i} - P\left(R^{CS}(q_t^{CS}) p_S^{m/S} + t_{t,C}^{S/C}\right) \|, \tag{2.24}$$

where $\hat{p}_S = [\hat{p}_S^{1/S} \cdots \hat{p}_S^{M/S}]$.

SLAM solutions can simultaneously track the camera and build a map of the environment. The map is often a rather complex data structure but the main task is to form correspondences between features and landmarks. The map saves the 3D points along with a descriptor, D, of the corresponding feature. Common descriptors are SURF [7] and SIFT [33]. Detected features in an image can then be compared to the map's descriptors. If a match is found the feature $p_{t,i}^{m/i}$ and the corresponding 3D point $p_S^{m/S}$ can be used in the optimization problems above. If a match is not found the feature can be added to the map. This can be done if the feature has been found in two images and the corresponding 3D point can then be estimated by triangulation [23].

The SLAM algorithm estimates the position from C to S and the orientation of C relative S, where the orientation is expressed as a quaternion. The position can only be estimated up to a scale, s, since a monocular camera is used. The estimates suffer from errors and can be modelled with an additive measurement noise model leading to the relation,

$$y_t^p = \frac{1}{s} t_{t,C}^{S/C} + e_t^p, \tag{2.25a}$$

$$y_t^q = q_t^{CS} + e_t^q, \tag{2.25b}$$

where y_t^p is the estimate of the position from SLAM and y_t^q is the estimate of the orientation both seen as measurements in the loosely coupled Extended Kalman Filter. q_t^{SC} and $t_{t,C}^{S/C}$ are the true values and e_t^p and e_t^q are modelled as white Gaussian noise.

2.3.6 Motion Model

Now when the different coordinate systems, the measurement models for the accelerometer and gyroscope and the theory of quaternions have been introduced, the motion model (2.1a) for the system can be specified. The states included

in the motion model are position $p_{t,N}^{I/N}$, quaternion to describe orientation $q_t^{N/I}$, velocity $\dot{p}_{t,N}^{I/N}$ and biases of the accelerometer $\delta_{t,I}^a$ and gyroscope $\delta_{t,I}^\omega$. The position describes the position of the system I relative to system N, resolved in N. The quaternion describes the orientation of system N relative to system I. The biases are included since it changes from time to time when the system is turned on and off and also online when the system is running because of the random walk, explained in Sections 2.3.3 and 2.3.4. The acceleration and angular velocity are used as inputs to the motion model, as in [24, 35]. The acceleration and angular velocity could also be considered as measurements and used in the measurement update. The drawback is that the acceleration and angular velocity both have to be included in the state vector which increase the computational complexity. However, if the predictions of the accelerations and angular velocity are needed the alternative to include them in the state vector has to be used [21]. The motion model that will be used here is the same as in [24],

$$p_{t+1,N}^{I/N} = p_{t,N}^{I/N} + T\dot{p}_{t,N}^{I/N} + \frac{T^2}{2}(R_t^{NI}(q_t^{NI})(y^a - v_{t,I}^a - \delta_{t,I}^a) + g_N), \tag{2.26}$$

$$\dot{p}_{t+1,N}^{I/N} = \dot{p}_{t,N}^{I/N} + T(R_t^{NI}(q_t^{NI})(y^a - v_{t,I}^a - \delta_{t,I}^a) + g_N), \tag{2.27}$$

$$q_{t+1}^{NI} = q_t^{N/I} + \frac{T}{2}S(y_{t,I}^\omega - v_{t,I}^\omega - \delta_{t,I}^\omega)q_t^{NI}, \tag{2.28}$$

$$\delta_{t+1,I}^a = \delta_{t,I}^a + v_{t,I}^{\delta_a}, \tag{2.29}$$

$$\delta_{t+1,I}^\omega = \delta_{t,I}^\omega + v_{t,I}^{\delta_\omega}, \tag{2.30}$$

where T is the sampling interval and the matrix $S(y_{t,I}^\omega - v_{t,I}^\omega - \delta_{t,I}^\omega)$ is calculated by using (2.12). The noises v are all white Gaussian noise. The variances of v, Q are parameters in the EKF see (2.2) and needs to be estimated before running the EKF. The estimation can be done by Allan variance and is explained in Section 2.4. Q is still a tuning parameter and the estimation from the Allan variance can be seen as a good start value.

2.3.7 Measurement Model

To get the full state-space model (2.1) the measurement model (2.1b) needs to be defined. The measurements y in (2.1b) are the outputs from the SLAM algorithm see Section 2.3.5. The measurement model that relates the measurements to the states is,

$$y_t^p = \frac{-1}{s}(-t_C^{S/N} + R^{CI}R_t^{IN}p_{t,N}^{I/N} - t_C^{I/C}) + e_t^p, \tag{2.31}$$

$$y_t^q = q^{CI}q_t^{IN}q^{NS} + e_t^q, \tag{2.32}$$

where $t_N^{S/N}$ is translation from N to S resolved in N, q^{NS} is the orientation of N relative to S, $t_I^{C/I}$ is the position of the camera relative the IMU and q^{CI} is the quaternion describing the orientation of the camera relative to the IMU. All these are parameters and needs to be estimated before running the Extended

Kalman Filter. The noises e are all white Gaussian noise. The variance of e, R, is a parameter in the EKF see (2.4) and needs to be estimated before running the EKF.

2.4 Calibration

In the above Section 2.3 some parameters have been introduced and this section will describe methods to estimate them.

2.4.1 IMU noise tuning using Allan variance

The noise parameters are important to estimate to get a good EKF. The filter needs accurate values of the noise parameters to decide how to trust the different measurements. This section will introduce the theory of Allan variance that can be used to estimate the variances of the noises of the IMU, $v_{t,I}^{a}$, $v_{t,I}^{\omega}$, $v_{t,I}^{\delta_a}$ and $v_{t,I}^{\delta_\omega}$, discussed in the Sections 2.3.3 and 2.3.4.

Allan variance was originally used to estimate different noise terms in crystal oscillators and atomic clocks, but is today also used to estimate noise terms in different sensors including inertial sensors [14], [43]. The Allan variance can be calculated as in [21],

$$\hat{s}_m = \frac{1}{N}\sum_{k=1}^{N} y_{(m-1)N+k} \qquad m = 1, 2, ..., M, \tag{2.33}$$

$$\hat{\lambda}(N) = \frac{1}{2(M-1)}\sum_{m=1}^{M-1}(\hat{s}_{m+1} - \hat{s}_m)^2, \tag{2.34}$$

where N is the length of the averaging interval, M is the number of averaging intervals and $\hat{\lambda}(N)$ is called Allan variance. If the total length of the dataset is L then M is $\lfloor L/N \rfloor$. The key to understand why the Allan variance can be used to calculate different noise terms is the relationship

$$\hat{\lambda}(T) = 4\int_0^\infty S_y(f)\frac{\sin^4(\pi f T)}{(\pi f T)^2}df, \tag{2.35}$$

where y is a random process for example measurement from a sensor with noise and $S_y(f)$ is the Power Spectral Density (PSD) of y. The derivation of (2.35) can be found in [36]. T is the time of each interval such that $T = Nt_0$, where t_0 is the length of the sampling interval. Two common noise models for inertial sensors are white noise and random walk see (2.15), (2.17), (2.16), (2.18) and their connection to the Allan variance will be explained in the two following subsections.

White Noise

High-frequency noise that have much shorter correlation time than the sample rate is called white noise and is characterized by its constant PSD. If the measurements y are sampled when the sensor is stationary and only white noise is present the PSD of the signal can be expressed as $S_y(f) = Q$. Substituting $S_y(f)$ in (2.35) and performing the integration the Allan variance becomes,

$$\hat{\lambda}(T) = \frac{Q}{T}, \tag{2.36}$$

where Q is the variance of the white noise for example the variance of $v^a_{t,I}$ or $v^\omega_{t,I}$ in (2.15) respectively (2.17) [14]. In a log-log plot of $\sqrt{\hat{\lambda}(T)}$ versus T, the white noise will appear with a slope of $\frac{-1}{2}$ [16, 14] and if only white noise is present $\log\sqrt{Q}$ can be read out at $T = 1$. If however more noise terms than white noise is present $\log\sqrt{Q}$ can not be read out directly at $T = 1$. Instead a straight line with gradient $\frac{-1}{2}$ can be fitted to the segment with slope $\frac{-1}{2}$ in the log-log plot. Reading the value of the line at $T = 1$ instead gives the value $\log\sqrt{Q}$ as seen in Figure 2.4.

Random Walk

Random Walk is noise with very large correlation time, often called brown/red noise, and can be modelled as integrated white noise and is characterized by the PSD $S_y(f) = \left(\frac{\sqrt{Q}}{2\pi}\right)^2 \frac{1}{f^2}$. Substituting $S_y(f) = \left(\frac{\sqrt{Q}}{2\pi}\right)^2 \frac{1}{f^2}$ into (2.35) and performing the integration gives,

$$\hat{\lambda}(T) = \frac{QT}{3}, \tag{2.37}$$

where Q is the variance of the the white noise that is integrated in the random walk process [14]. In a log-log plot of $\sqrt{\hat{\lambda}(T)}$ versus T, the Random Walk will appear with a slope of $\frac{1}{2}$. Fitting a straight line with gradient $\frac{1}{2}$ and reading the value at $T = 3$ gives the value of $\log\sqrt{Q}$ as seen in Figure 2.4.

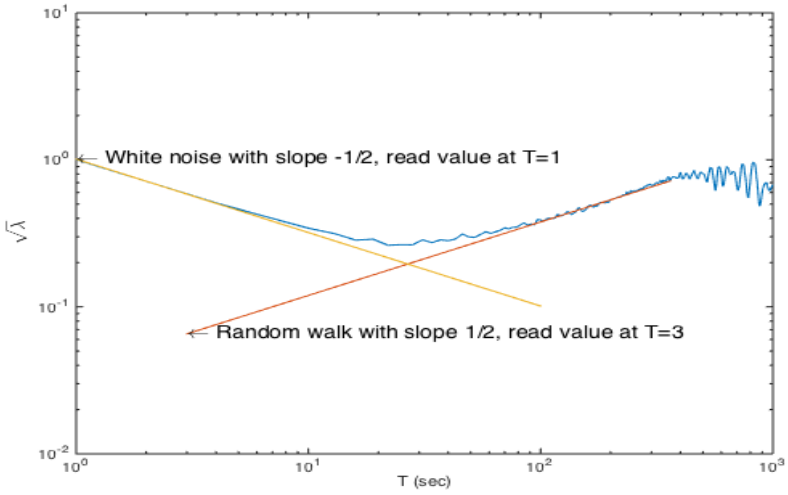


Figure 2.4: *Allan variance of white and brown noise*

2.4.2 Calibration of Camera and IMU Coordinate systems

The IMU and camera are not placed at the same position in the phone which will cause them to accelerate differently when turning. For accurate tracking using both camera and IMU it is important to know the relative pose (orientation and translation) between the two sensors to get an accurate measurement model. This section will go through some theory that can be used to estimate the translation $t_C^{I/C}$ and orientation q^{IC} used in (2.31) and (2.32).

In [32] they first estimate the rotation in a calibration step, using a vertical chessboard target, where both sensors observe the vertical direction. Then they use the information that the accelerometer only measures gravity when stationary which can be used as a vertical reference for the IMU and the camera can extract a vertical reference from the vertical chessboard target. These vertical references can then be compared to find the relative rotation between the two sensors. The translation can be estimated using a turntable, by adjusting the system such that it is turning about the intertial sensor's null point.

[26] takes a system identification approach and identifies the problem as a gray-box problem. They try to minimize the innovations from the measurement update in an Extended Kalman Filter (EKF) by adjusting the relative orientation and translation between the camera and IMU. [35] also estimates the relative pose using an EKF by augmenting the state vector with the unknown transformation between the camera and IMU. They initiate an approximate value of the transformation between the camera and IMU and then they let the EKF refine the estimate. These methods do not need any additional hardware as a turntable

but needs an Extended Kalman Filter. They both use a tightly coupled EKF and there is no guarantee that a loosely coupled would give the same accuracy since the number of innovations from the measurement update are far less.

[18] formulates a maximum a posteriori problem in continuous time using B-splines. This is done because estimation problems in discrete time does not scale very well with high-rate sensors like IMU, since the the states typically grow with the number of measurements [17]. The maximum a posteriori problem breaks down into an optimization problem solved with the Levenberg-Marquardt (LM) algorithm. [18] does both estimate the temporal and spatial displacement of the camera and IMU, which is convenient if the two sensors use different clocks. The temporal and spatial displacement are uncorrelated. When simultaneously estimating two uncorrelated quantities, the inaccuracy of the estimation for one quantity risks to impair the estimation of the other quantity. However, it is shown in [18] that using all information in a unified estimate increase the accuracy of the estimation.

Since no tightly coupled EKF was developed, no turntable was available and that the two sensors in the phone uses different clocks the toolbox, called KALIBR [19], was chosen for this thesis, which is the implementation of [18].

KALIBR Toolbox

Instead of estimating the time-varying states in discrete time [18] approximate them as a B-spline function,

$$x(t) = \Phi(t)c, \tag{2.38}$$

where $x(t)$ are the states, $\Phi(t)$ is a known matrix of B-spline basis functions and c are parameters defining $x(t)$. The estimation can be seen as curve fitting. What is the value of c that best fits the measurements of the sensors?

The toolbox uses three coordinate systems. The coordinate systems for the camera and IMU are the same as in Section 2.3.1 and the third is similar to N and S. The third is called the world frame, W, and is fixed with respect to the earth and all the 3D points in the scene are expressed in this coordinate system. The 3D points are assumed known. The non-time-varying quantities estimated in the toolbox [18] are

1. the gravity direction, g_W
2. the transformation matrix between the camera and the IMU, T^{CI}
3. the time offset between the IMU and camera, d

and the time-varying quantities are

1. the transformation matrix between the IMU and the world frame, $T^{WI}(t)$
2. accelerometer bias, $\delta_I^a(t)$

3. gyroscope bias, $\delta_I^{\omega}(t)$.

The transformation matrix $T^{WI}(t)$ is formed from a subset of the states represented by B-spline functions,

$$T^{WI}(t) = \begin{pmatrix} R(\Phi_\gamma(t)c_\gamma) & \Phi_{p_W^{I/W}}(t)c_{p_W^{I/W}} \\ 0_{1x3} & 1 \end{pmatrix}, \tag{2.39}$$

where R form a rotation matrix from the B-spline function $\Phi_\gamma(t)c_\gamma$, which represent the orientation, γ. $\Phi_{p_W^{I/W}}(t)c_{p_W^{I/W}}$ represent the position of the IMU to the world frame W, $p_W^{I/W}$. [18] uses Cayley-Gibbs-Rodrigues parametrization to represent orientation and the transformation to a rotation matrix can be found in [17]. It is important to notice that the rotation matrix is not defined as (2.9) because Cayley-Gibbs-Rodrigues parametrization is used instead of quaternions. Different parametrization results in the same rotation matrix, but is formed in a different way depending on the parametrization.

Since $\Phi_{p_W^{I/W}}(t)$ is a known function and $c_{p_W^{I/W}}$ is not a function of time, the acceleration of the IMU can easily be found by computing the second derivative of the position,

$$\ddot{p}_W^{I/W}(t) = \ddot{\Phi}_{p_W^{I/W}}(t)c_{p_W^{I/W}}. \tag{2.40}$$

The angular velocity can be found as,

$$\omega_W^{WI}(t) = S\left(\Phi_\gamma(t)c_\gamma\right)\dot{\Phi}_\gamma(t)c_\gamma, \tag{2.41}$$

where S is a matrix relating angular parametrization to angular velocity similar to (2.12) and the definition can be found in [17]. The toolbox uses similar measurement models as in Section 2.3. The accelerometer measurements y_k^a are sampled at time t_k for $k = 1 \dots K$. The measurement model for sample k can be written as,

$$y_k^a = R^{IW}(\Phi_\gamma(t_k)c_\gamma)(\ddot{p}_W^{I/W}(t_k) - g_W) + \delta_I^a(t_k) + v_I^a(t_k), \tag{2.42}$$

where $v_I^a(t_k)$ is modelled as white Gaussian noise,

$$v_I^a(t_k) \sim \mathcal{N}(0, R_a).$$

The gyroscope measurement model can be written as,

$$y_k^\omega = R^{IW}(\Phi_\gamma(t_k)c_\gamma)\omega_W^{WI}(t_k) + \delta_I^\omega(t_k) + v_I^\omega(t_k), \tag{2.43}$$

where $v_I^\omega(t_k)$ is modelled as white Gaussian noise,

$$v_I^\omega(t_k) \sim \mathcal{N}(0, R_\omega).$$

The measurement model for the camera are the correspondences between 2D points and 3D points. The 3D points are called landmarks and the location of the

landmarks are known in the world frame since a calibration pattern is used, see Figure 2.5. There are M landmarks and landmark m is denoted p_W^m. There are J images and j denotes image j. The correspondence in image j for landmark m and time t_j can be written as,

$$y_{mj} = P\left(T^{CI}T^{IW}(t_j + d)p_W^m\right) + v_C^{y_{mj}}(t_j), \tag{2.44}$$

where $v_C^{y_{mj}}(t_j)$ is modelled as white Gaussian noise,

$$v_C^{y_{mj}}(t_j) \sim \mathcal{N}(0, R_{y_{mj}})$$

and P is a camera projection matrix.

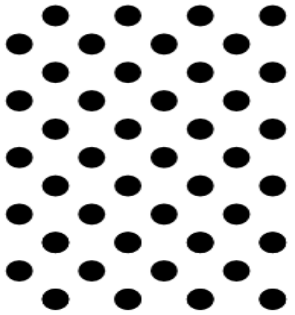


Figure 2.5: *Calibration pattern for the toolbox*

The biases are modelled as random walks. The accelerometer bias is modelled as,

$$\dot{\delta}_I^a(t) = v_I^{\delta^a}(t), \tag{2.45}$$

where $v_I^{\delta^a}(t)$ is modelled as white Gaussian noise

$$v_I^{\delta^a}(t) \sim \mathcal{N}(0, R_{\delta^a}).$$

The gyroscope bias is modelled as,

$$\dot{\delta}_I^\omega(t) = v_I^{\delta^\omega}(t), \tag{2.46}$$

where $v_I^{\delta^\omega}(t)$ is modelled as white Gaussian noise

$$v_I^{\delta^\omega}(t) \sim \mathcal{N}(0, R_{\delta^\omega}).$$

The toolbox minimizes the error between estimated variables and the measurements. The errors e are formed from the measurement models above which are then used to form cost functions J in the following way [18],

$$e_{y_{mj}} = y_{mj} - P(T^{CI}T^{IW}(t_j + d)p_W^m), \tag{2.47a}$$

$$J_y = \frac{1}{2}\sum_{j=1}^{J}\sum_{m=1}^{M} e_{y_{mj}}^T R_{y_{mj}} e_{y_{mj}}, \tag{2.47b}$$

$$e_{y_k^a} = y_k^a - R^{IW}(\Phi_\gamma(t_k)c_\gamma)(\ddot{p}_W^{I/W}(t_k) - g_W) + \delta_I^a(t_k), \tag{2.47c}$$

$$J_a = \frac{1}{2}\sum_{k=1}^{K} e_{y_k^a}^T R_a e_{y_k^a}, \tag{2.47d}$$

$$e_{y_k^\omega} = y_k^\omega - R^{IW}(\Phi_\gamma(t_k)c_\gamma)\omega_W^{WI}(t_k) + \delta_I^\omega(t_k), \tag{2.47e}$$

$$J_\omega = \frac{1}{2}\sum_{k=1}^{K} e_{y_k^\omega}^T R_\omega e_{y_k^\omega}, \tag{2.47f}$$

$$e_{\delta^a}(t) = \dot{\delta}_I^a(t), \tag{2.47g}$$

$$J_{\delta^a} = \int_{t_1}^{t_k} e_{\delta^a}(\tau)^T R_{\delta^a} e_{\delta^a}(\tau) d\tau, \tag{2.47h}$$

$$e_{\delta^\omega}(t) = \dot{\delta}_I^\omega(t), \tag{2.47i}$$

$$J_{\delta^\omega} = \int_{t_1}^{t_k} e_{\delta^\omega}(\tau)^T R_{\delta^\omega} e_{\delta^\omega}(\tau) d\tau. \tag{2.47j}$$

The Levenberg-Marquardt algorithm are then used to minimize the sum of all cost functions $J_y + J_a + J_\omega + J_{\delta^a} + J_{\delta^\omega}$.

3

Method

This section is divided into two parts. The first part, "Calibration", describes how different calibrations were done, while the second part, "Movements tests", describes the tests done to evaluate the EKF. An overview of which order the different calibrations are conducted can be seen in Figure 3.1.

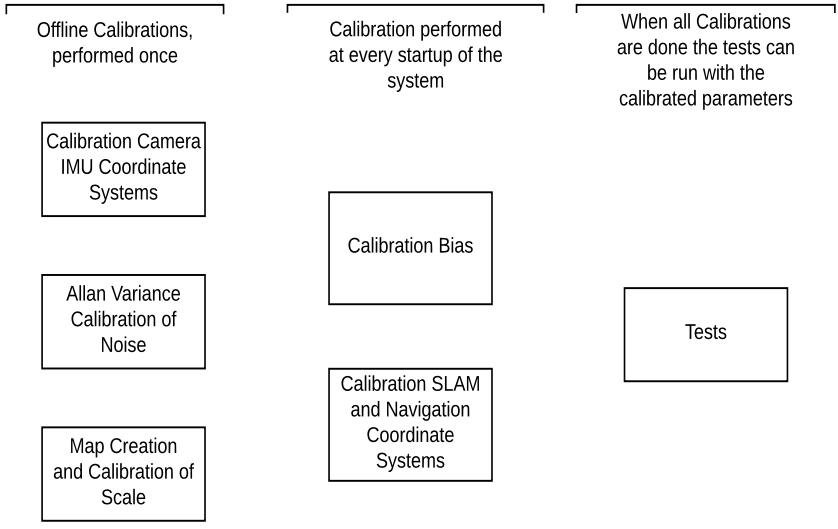


Figure 3.1: *Illustration of when all the calibrations are conducted*

3.1 Calibration

This section will go through all calibrations that was done to use the EKF, starting with the noise of the IMU.

3.1.1 Calibration of Noise using Allan variance

For the EKF to work well it needs good estimates of the noise parameters of the IMU, Q_k. An accurate estimation of these parameters makes it possible for the filter to fuse the measurements of the IMU and camera correctly. The noise terms of Q_k can be estimated using the Allan variance, see Section 2.4.

The noise parameters are also needed in the toolbox that estimates the relative pose of the camera and IMU. They are needed in the optimization problem for the parameters R_a and R_ω in (2.47d) and (2.47f).

To calculate the Allan variance three datasets approximately of 3.5 hours from the IMU were sampled with a sampling frequency of $f_0 = 200$ Hz in a stationary position. The length of the averaging interval, T, was chosen as [3] suggests, $T = \frac{2^j}{f_0}$, $j = 0, 1, 2...\lfloor \frac{M*t_0}{2} \rfloor$, where M is the length of the dataset.

The slope vector, k, of the Allan variance was computed as,

$$k_i = \frac{(\log(\lambda(T_{i+1})) - \log(\lambda(T_i)))}{\log(T_{i+1}) - \log(T_i)}, \qquad T_i = \frac{2^i}{f_0} \quad \text{for } i = 0, 1, 2...\lfloor \frac{M}{2} \rfloor. \tag{3.1}$$

If two consecutive intervals with equal inclination of either $\frac{-1}{2}$ or $\frac{1}{2}$ and the adjacent intervals' inclinations were similar, the Allan variance was considered to have that slope. If a slope was accepted a line could be fitted to the segment where the slope of either $\frac{-1}{2}$ or $\frac{1}{2}$ was found and the noise parameters could be recorded at $T = 1$ respectively $T = 3$, as described in Section 2.4.

3.1.2 Calibration of bias

For the EKF to be able to use the IMU optimally the bias estimates $\delta^a_{t,I}$ and $\delta^\omega_{t,I}$ have to be accurate at all time. Since the biases are different at every start up of the system there is no possibility to have an accurate initial value of the biases. Thus they have to be calibrated every time the system is started. This was done by holding the phone in a stationary position (i.e not moving) of approximately 8 minutes and letting the EKF:s estimates of the biases become stable.

In the normal EKF, SLAM poses are used in the measurement update, see Section 2.3.5. The SLAM poses include errors and might affect the bias estimation. The information that the phone is hold stationary during the calibration can be used in the measurement update. A modified measurement update was tried where a constant value was used in the measurement update during the calibration, indicating that the phone is not moving at all. The modified measurement

update looks like,

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_0 - h(\hat{x}_{t|t-1}, u_k, \hat{e}_t), \tag{3.2}$$

where y_0 is the constant value, which is the first output from SLAM, compared to the normal measurement update (2.4b).

If the bias estimates are good, the dead reckoning [11] of the IMU should work well. Dead reckoning of the IMU is the same as skipping the measurement update and only use the time update, see Section 2.2.1. The time update uses only the IMU for tracking and no aiding sensor, i.e camera in this case. Over time the errors originating from the IMU will be accumulated and result in a error in the pose.

To evaluate the bias estimation of the normal and modified EKF, a comparison of the dead reckoning was done. In the test the phone was held stationary. After approximately 8 minutes, when the biases have converged, the measurement update in the EKF was turned off and only the time update was used, while the phone was kept in a stationary state. The error was recorded after 35 seconds of dead reckoning. In Figure 3.2 it can be seen how the error of the z-component of $p^{I/N}_{t,N}$, e^z_N, grows after 500 s when the measurement update was turned off and the error was recorded at time 535 s.

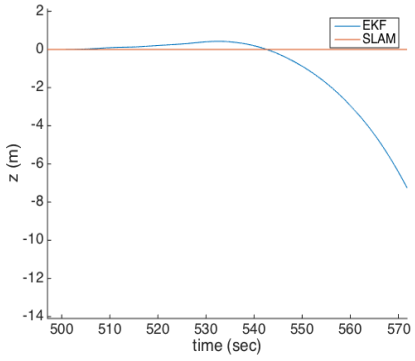


Figure 3.2: *Illustration of how the EKF starts to diverge from SLAM, when only dead reckoning was used.*

Four datasets were compared. In the first three datasets the coordinate systems of the IMU, I, and the navigation coordinate system, N, were aligned, such that y_I was aligned to y_N etc. Also a forth dataset was tested where the phone was rotated 90° around z_N, before the test was started. This makes x_I to be aligned with $-y_N$ and y_I to be aligned with the x_N. This was done to see if gravity affected the result.

3.1.3 Calibration of IMU and Camera Coordinate Systems

In this thesis an already existing open source toolbox called KALIBR [19] was used which is the implementation of [18], discussed in Section 2.4.2. A calibration target is used when recording the dataset used as input to the toolbox. A calibration target is a pattern with distinctive features and known distances between the features. The calibration pattern can be seen in Figure 2.5.

To estimate unknown parameters of a dynamic system it is necessary that the system is observable. Observability ensures that it is possible to estimate the parameters given measurements from the system. For the system of a camera and IMU to be observable the phone has to be rotated around at least two axes [35]. The speed of the movement also affects the observability. Faster rotation increases the disparity of the camera's and IMU's acceleration and make the relative translation more observable. The phone was rotated and moved along all axes to ensure observability.

3.1.4 Calibration of SLAM and Navigation Coordinate Systems

The relative orientation between the two global coordinate systems S and N, q^{NS}, was set during initialization using (2.32), i.e

$$q^{NS} = q_{t_0}^{NI} q^{IC} y_{t_0}^{q}, \tag{3.3}$$

where t_0 indicates the time when the filter is started. To get an accurate value of q^{NS}, good estimates of the terms in the right hand side of (3.3) are required. $q_{t_0}^{NI}$ was estimated with help the of GearVr's algorithm [20] to find the orientation between the IMU and a vertically aligned coordinate system. To calculate the orientation, the algorithm uses the fact that the accelerometer only measures gravity when stationary. $y_{t_0}^{q}$ was estimated by the SLAM algorithm. To improve this value the camera was held still leading to no rolling shutter and motion blur effects and pointed to an area with a lot of features. q^{IC} was calculated according to Section 3.1.3. All the values in the right hand side of (3.3) are then known and can be used in (3.3) to get an estimate of q^{NS}.

3.1.5 Calibration of Scale

To fuse monocular camera and IMU measurements it is necessary to know the scale of SLAM. The SLAM algorithm itself can not determine the scale without any knowledge of the true distances in the images. One way of deciding the scale is to walk a known distance and scale the distance from the SLAM algorithm to match the known distance travelled. This does not work if done when a map is under construction since bundle adjustments will move around map points, hence rescaling the map, see Section 2.3.5. Therefore a map was always first constructed and then a one meter step was then taken, which was used to define the scale of the map. The map could then be loaded during initialization when different tests were carried out.

3.2 Movement tests

This section will describe the tests performed to evaluate how well the EKF could track motions. A lot of tests have been performed, but only ”fast translation”, ”rotation” and ”rotation lost” are presented. These tests captures the pros and cons and illuminates the current problems. For the two first tests, fast translation and rotation, the measurement update was also turned off during motion to see how well the dead reckoning of the IMU worked.

All tests were started with a calibration step of the bias, since it changes at every start up of the system, see Section 3.1.2. After the calibration step the movements were done, e.g. fast translation. Also and predefined map was loaded, since the scale of the SLAM has to be known to fuse IMU and camera.

For all tests the camera was sampled with a sampling frequency of 30 Hz and the IMU was sampled in 200 Hz. Even if the camera was sampled in 30 Hz, SLAM only provided EKF with poses with a a lower frequency, typically as seen in Figure 3.3. This is due to it takes longer time for SLAM to process the images than new images arrives. If a new image arrives before SLAM has finish processing the earlier image, the new image is discarded.

For many sensors the measurements are used in the EKF directly when they arrive, without accounting for that the measurements might correspond to an earlier time instance. This is often enough since the effect of the delay is small compared to other simplifications or errors. However, the camera measurement delays are significant and the results will show that they can not be ignored. In this thesis, the EKF did not account for this, and the big impact of this simplification is highlighted in Section 4.2.1.

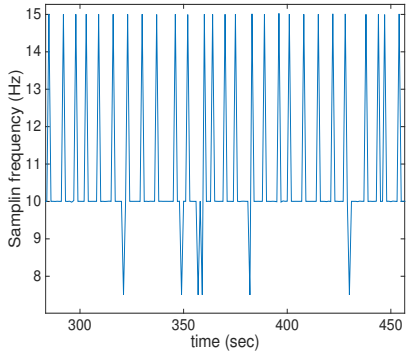


Figure 3.3: *Sampling frequency of SLAM.*

3.2.1 Fast translation

This test was performed to see if the EKF could increase the performance during fast motion when SLAM is supposed to not work as well. The phone was moved rapidly 38.5 cm along the negative z_N. To see how well the dead reckoning of the IMU worked during fast translation, the measurement update was also turned off during the motion and compared to the result when used.

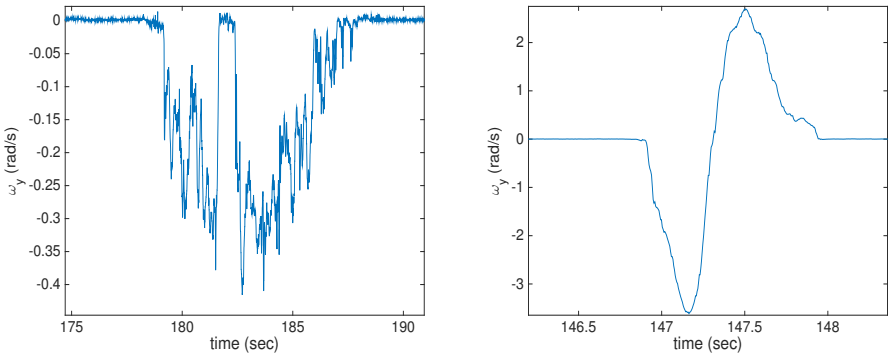
3.2.2 Rotation

Rotation is a more challenging task than translation because even a slow rotation can change the scene of the surroundings relatively much compared to a slow translation. A faster change of the scene leads to more rolling shutter and motion blur effects and the accuracy of the SLAM algorithm decreases.

In this test the phone was rotated 90° around y_N. The phone was not rotated around the IMU's origo, thus the rotation will also lead to translation. To see how well the dead reckoning of the IMU worked during rotation, the measurement update was also turned off during the rotation and compared to the result when used. The angular velocity of y_N is seen in Figure 3.4a.

3.2.3 Rotation lost

SLAM easily lose tracking when rotating fast. A fast rotation test was done to see if the EKF could be used to keep track of the orientation even when SLAM gets lost. The phone was rotated 45° around y_N and back again to its start position. The angular velocity of the y_N can be seen in Figure 3.4b and compared to Figure 3.4a the rotation was much faster in this test.



(a) *angular velocity of the y-axis in rad/s from test Rotation*

(b) *angular velocity of the y-axis in rad/s from test Rotation lost*

Figure 3.4: *Angular velocity of the rotation tests*

4

Results and Discussion

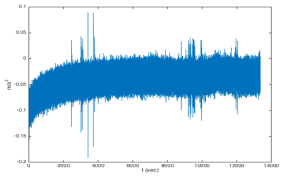
This chapter will present all the results along with discussions. The chapter is divided into two sections. The first section presents the results from all the calibrations and the second section compares the EKF with the SLAM algorithm.

4.1 Calibration

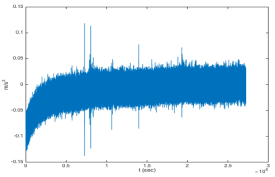
In this section the results from the calibration of the noise, the calibration of IMU and camera and the calibration of the bias are presented, starting with the noise.

4.1.1 Calibration of Noise using Allan variance

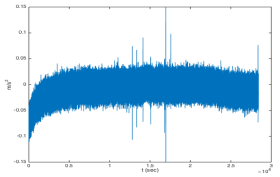
The IMU was started cool for all datasets in this calibration. Temperature is one factor that changes the bias and the accelerometer data has a "temperature transient" in the beginning of the dataset seen in Figure 4.1. Even if no temperature data was available, the conclusion was that the temperature steadily increased the first hour, when the accelerometer values also steadily increased. The random walk model is used to model random fluctuations in the bias. The steadily increased bias because of the steadily increased temperature should not be modelled by this model and thus the first hour was removed from the datasets. It can be pointed out that this is the behaviour of the particular IMU used in this thesis and it might differ from others. Other IMUs might not have this significant "temperature transient" and some might also have a built in temperature compensation to reduce this undesirable behaviour.



(a) accelerometer data of the x-axis from dataset 1



(b) accelerometer data of the x-axis from dataset 2

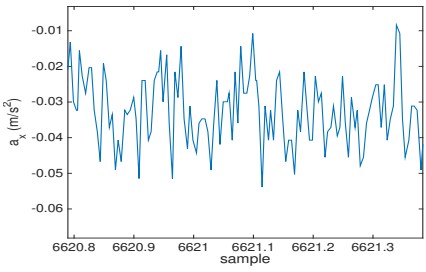


(c) accelerometer data of the x-axis from dataset 3

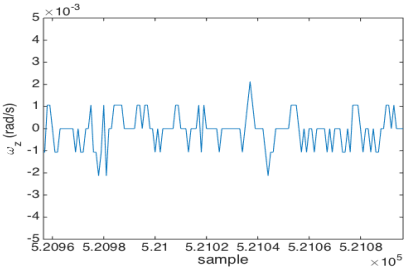
Figure 4.1: *Accelerometer data of the x-axis from 3 datasets*

The Allan variance plots are very similar for small averaging intervals ,T, for all datasets, e.g. seen in Figure 4.3a. The slope of $\frac{-1}{2}$ can be found for small averaging intervals, T and is found for all axes and for all datasets. This means that the sensors suffer from white noise. In Figures 4.2a and 4.2b data from the gyroscope and accelerometer can be seen when zoomed in. The data rapidly changes thus indicating white noise.

In Figure 4.2b the data from the gyroscope shows a lot of identical values. This is due to quantization. The precision of the A/D converter seems to be such that a lot a values from the gyroscope are converted to the same. This could potentially corrupt the Allan variance since quantized noise is not completely white. However, the Allan variance should also be able to detect errors originating from quantization, which are represented in the Allan variance plot with a slope of -1. No slope of -1 could be found for either the gyroscope or accelerometer and the conclusion was that the quantization errors were negligible.



(a) zoomed in accelerometer data.



(b) zoomed in gyroscope data. Many identical values due to quantization

Figure 4.2: *Data zoomed in from the gyroscope and accelerometer. The rapid changes indicate white noise*

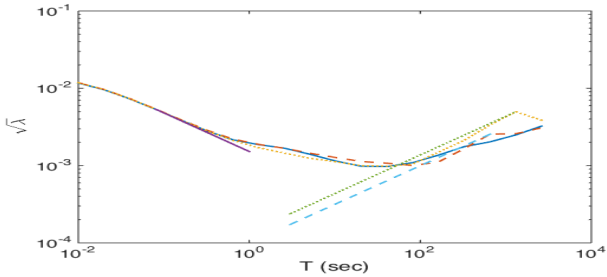
It is much harder to draw any conclusion about the random walk of the sen-

sors since the plots differ much more for larger T where the slope of $\frac{1}{2}$ normally can be found. This can be seen in Figure 4.4a. It is not possible to find a slope of $\frac{1}{2}$ for all axes and the datasets shows different results. Also, the uncertainty of the Allan variance also grows with the averaging interval T where the slopes of $\frac{1}{2}$ are found [14]. In light of these observations it might be questionable if the bias should be modelled as random walk or just a constant value. If chosen to be modelled as random walk how should the value be chosen for the variance of the random walk? There is no obvious answer to this question. The Allan variance gives a guideline of the variance of the random walk, but does probably not reflect reality perfectly. The run time of the application also affects the choice. If the run time is short, a constant bias model might be more suitable. The run time of Virtual Reality applications are relatively long. One could at least expect half an hour run time and then a random walk model might give performance advantages.

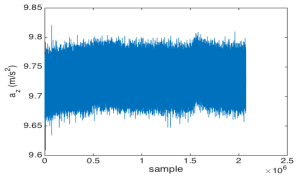
As mention earlier the accelerometer has a long "temperature transient", approximately one hour. In reality the VR application is likely to be started when the phone is cool and then the "temperature transient" has to be accounted for. This could be done by a model that takes the temperature of the IMU into account. However, this was not investigated in this thesis and instead the tests were performed when the IMU was in a warm state.

Axis	White Noise	Units	Random Walk	Units
Gyro X dataset 1	0.000098	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	-	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro X dataset 2	0.000092	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	0.0000072	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro X dataset 3	0.000092	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	0.0000052	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro X mean	0.000094	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	0.0000062	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro Y dataset 1	0.000091	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	0.0000038	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro Y dataset 2	0.000085	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	-	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro Y dataset 3	0.000085	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	0.0000052	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro Y mean	0.000087	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	0.0000045	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro Z dataset 1	0.000091	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	-	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro Z dataset 2	0.000091	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	-	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro Z dataset 3	0.000091	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	-	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Gyro Z mean	0.000091	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	-	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$
Acc X dataset 1	0.00096	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	-	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc X dataset 2	0.00095	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	-	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc X dataset 3	0.00096	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	0.0001	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc X mean	0.00096	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	0.0001	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc Y dataset 1	0.00098	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	-	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc Y dataset 2	0.00099	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	-	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc Y dataset 3	0.00097	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	-	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc Y mean	0.00098	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	-	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc Z datset 1	0.0015	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	-	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc Z dataset 2	0.0015	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	0.00017	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc Z dataset 3	0.0015	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	0.00024	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Acc Z mean	0.0015	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	0.00020	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$

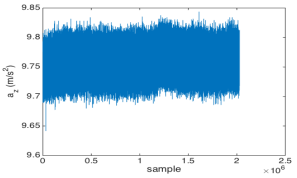
Table 4.1: *Noise parameters from Allan variance*



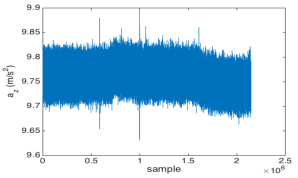
(a) *Allan variance of the z-axis of the accelerometer.*



(b) *accelerometer data of the z-axis from dataset 1*

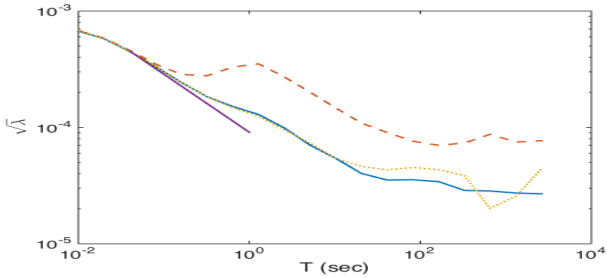


(c) *accelerometer data of the z-axis from dataset 2*

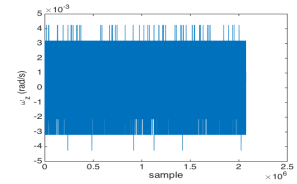


(d) *accelerometer data of the z-axis from dataset 3*

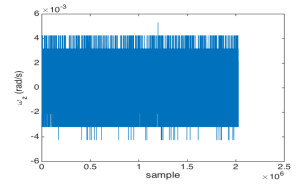
Figure 4.3: *Accelerometer of the z-axis for the three first datasets*



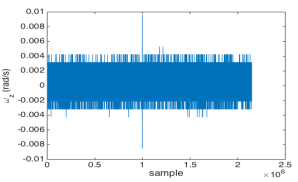
(a) *Allan variance of the z-axis of the gyroscope.*



(b) *gyroscope data of the z-axis from dataset 1*



(c) *gyroscope data of the z-axis from dataset 2*

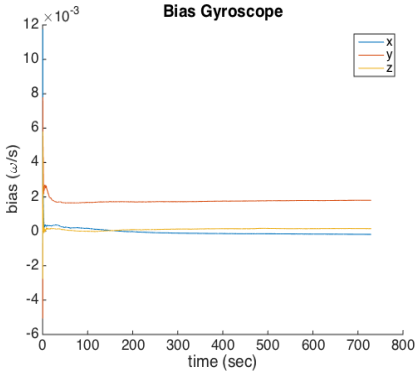


(d) *gyroscope data of the z-axis from dataset 3*

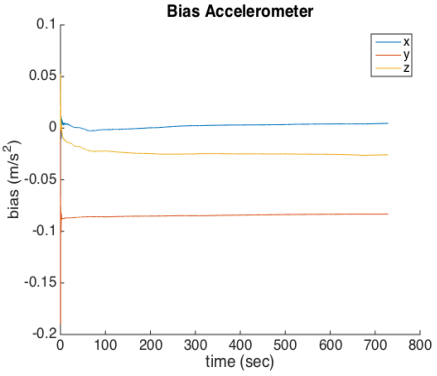
Figure 4.4: *Gyroscope data of the z-axis for the three first datasets*

4.1.2 Calibration of bias

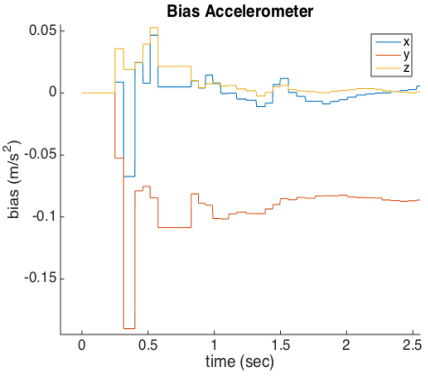
In this test the bias is evaluated by looking at the error after dead reckoning the IMU. In table 4.2 the column ”error normal” is the result when the normal measurement update was used and the third column ”error modified” is the result when the modified measurement update was used. A plot of the bias estimates from one dataset is also shown to see how the bias estimates typically behaves. This is seen in Figure 4.5.



(a) Bias estimates of the gyroscope. The biases are stable but not completely constant.



(b) Bias estimates of the accelerometer. The biases are stable but not completely constant



(c) Bias estimates of the accelerometer zoomed in. It is possible to see how they change rapidly in the beginning to then get more stable.

Figure 4.5: *Typical behaviour of the bias estimates.*

The tables show very similar results for the two methods. The conclusion is that using the extra information did not result in a better estimate of the bias.

Axis	error normal	error modified
e_N^x	25.8 m	24.9 m
e_N^y	50.8 cm	49.3 cm
e_N^z	68.7 cm	40 cm
angle error	0.6 °	0.6 °

Table 4.2: *Errors from dataset 1, when dead reckoning the IMU. e_N^a is the error of the a-axis of the position $p_{t,N}^{I/N}$ after 35 seconds of dead reckoning.*

Axis	error normal	error modified
e_N^x	14.9 m	14.5 m
e_N^y	30.7 cm	28.9 cm
e_N^z	5.1 m	5.6 m
angle error	0.43 °	0.46 °

Table 4.3: *Errors from dataset 2, when dead reckoning the IMU*

Axis	error normal	error modified
e_N^x	19.8 m	19.4 m
e_N^y	22 cm	24 cm
e_N^z	4.0 m	5.7 m
angle error	0.39 °	0.39 °

Table 4.4: *Errors from dataset 3, when dead reckoning the IMU*

Axis	error normal	error modified
e_N^x	16.8 m	16.5 m
e_N^y	29 cm	27 cm
e_N^z	8.0 m	8.5 m
angle error	0.31 °	0.29 °

Table 4.5: *Errors from dataset 4, when dead reckoning the IMU*

SLAM is as mention earlier very accurate when standing still and this could be one reason why the extra information did not give much.

By looking at the errors of the three first datasets, it is clear that the errors e_N^x and e_N^z are bigger than e_N^y. The first thought might be that the y-component of bias estimate $d_{t,I}^a$ is much better, since I and N were aligned according to 3.1.2. However, the result from dataset 4 does not support this conclusion, since x_I was aligned with the negative y_N. If the x-component of $d_{t,I}^a$ was much harder to esti-

mate, the error e_N^y of dataset 4 would be more similar to the error e_N^x for the three first datasets. This is not the case and the error e_N^y in dataset 4 is still low. A more likely explanation is that the angle error leads to the gravity being projected onto x_N and z_N. Since a vertical coordinate system is used, see Section 2.3.1, gravity is supposed to be completely projected onto y_N. Figure 4.6 illustrates the situation, where an angle error of θ is introduced and no other acceleration than gravity is present. Double integrating the acceleration for 35 seconds gives the error in the position after 35 seconds. The double integration yields,

$$e_N^y = \int_0^{35}\int_0^{\tau} (cos(\theta)g_N - g_N)d\tau dt = /\theta = 1°/ = -0.9m, \tag{4.1}$$

$$e_N^x = \int_0^{35}\int_0^{\tau} -sin(\theta)g_N d\tau dt = /\theta = 1°/ = -10.7m. \tag{4.2}$$

As seen, the angle error influences the error e_N^x much more than the e_N^y. Figure 4.7 shows the gravity resolved in N for the third dataset and the plots support the conclusion.

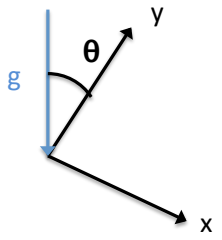
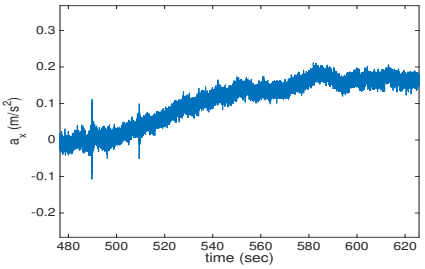


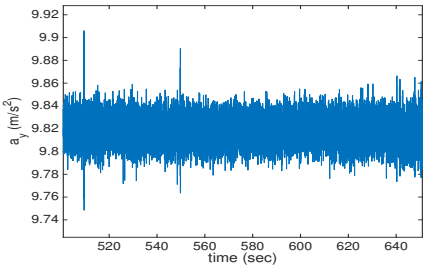
Figure 4.6: *Illustration of an angle error θ*

4.1.3 Calibration of IMU and Camera Coordinate Systems

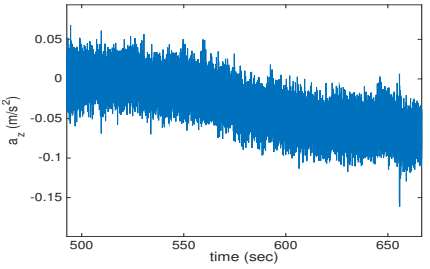
The prior information was that the rotation between the IMU and camera were a rotation of 180° around the x-axis based on the information given from the supplier of the phone. These values are not always accurate since they are not calibrated for every single phone produced at a factory, which makes it important to calibrate your specific phone. However the value can be used to see if the results of the calibration toolbox seems reasonable. The translation of the IMU and camera can be roughly estimated by looking at the location of the sensors. This is not always easy as in the case for the Samsung Galaxy S6 where you can



(a) *Measurements of the x-axis of the accelerometer resolved in N. Supposed to be 0*



(b) *Measurements of the y-axis of the accelerometer resolved in N. Supposed to be 9.82*



(c) *Measurements of the z-axis of the accelerometer resolved in N. Supposed to be 0.*

Figure 4.7: *Gravity resolved in coordinate system N frame for dataset 3.*

not disassemble the phone without special tools. A source on the internet suggests that the translation is $(\,1.7\ \ -1.9\ \ 0\,)$ cm [15].

The rotation q^{IC} were estimated to,

$$\begin{pmatrix} 0.003 & 1 & 0.005 & 0.004 \end{pmatrix}. \tag{4.3}$$

The value seems reasonable given the prior information. The estimate of the rotation of the toolbox also seemed robust since the same value was obtained from many tests and also with different movements.

The translation $t_C^{I/C}$ was estimated to,

$$\begin{pmatrix} 0.0013 & 0.0018 & 0.0012 \end{pmatrix} \text{ cm}, \tag{4.4}$$

which seemed unreasonable given the prior information. The estimation was also not robust and a couple of different results were obtained in different tests, but the one presented was the most occurring one.

The reason why the translation could not be obtained could be because of a couple of things. The models in the toolbox does not account for the rolling shutter effects which affects the estimation. These effects could be reduced by slow movements. However slower movements did not give any better results. This is likely due to lower movements reduces the observability and the optimization converge to the wrong value. The average absolute angular velocity measured by the gyroscope was $10.5°s^{-1}$ during the test presented. This was not close to average presented in [18] which had a average of $55°s^{-1}$ who used a global shutter camera. Increasing the angular velocity introduces too large errors and could not be done. The conclusion is that the toolbox has to be modified to account for rolling shutter effects to get an accurate value of the translation.

4.2 Results movement

This section will present all the results from the motion tests. Fast translation, Rotation and Rotation lost were tested.

4.2.1 Fast translation

This section is divided into two subsections. The same dataset was used for both tests, but in the first subsection the measurement update was used during the whole translation. In the other test the measurement update was turned off during the translation, to see how well the dead reckoning of the IMU works during translation.

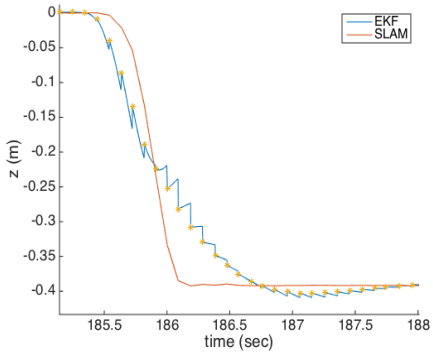
Fast translation with measurement update

This section will present the results from fast translation in negative z_N-direction. First the position in z-direction, $p^{I/N}_{t,N,z}$, is analysed and then the two other axes are discussed.

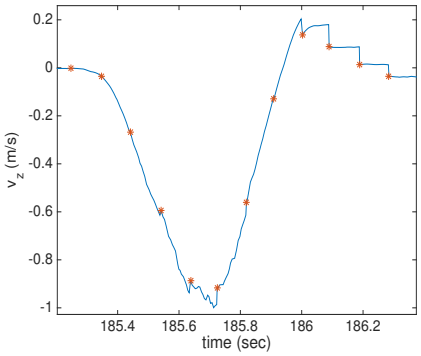
For the z_N-axis the delay of SLAM is apparent and leading to the measurement update doing more harm than good. In the beginning in Figure 4.8a it is seen how EKF starts to move before SLAM and is ahead until roughly the middle of the movement. Since the poses of SLAM are delayed, the pose of the EKF is compared to an old pose in the measurement update. This is more apparent during fast motions when the phone moves relatively much during the time equal to the time delay. The states of the EKF are then corrected towards an old position.

The velocity $\dot{p}^{I/N}_{t,N,z}$ is also a state in the EKF and will also be affected by the measurement update. The error in velocity, originating from the delayed measurements are easiest seen in the plot of the position at time 186 in Figure 4.8a. At time 186 the position starts to go upwards in positive z-direction, but in reality the phone has already been stopped and the velocity is supposed to be 0.

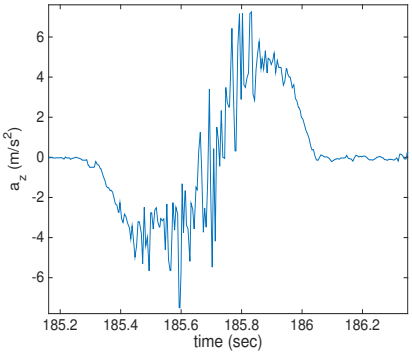
Two things stand out when looking at the x_N and y_N position. Firstly, the ac-



(a) *z_N position of EKF and SLAM, with '*' indicating time for measurement update. The time delay of SLAM can be seen and how it affects the measurement update*



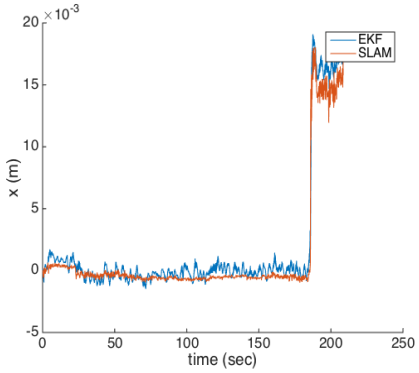
(b) *Velocity of z resolved in the N estimated by the EKF. '*' indicates time for measurement update*



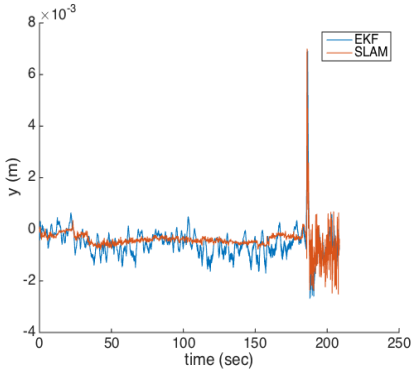
(c) *acceleration of z-axis resolved in the N*

Figure 4.8: *position, velocity and acceleration of the z-axis from fast translation*

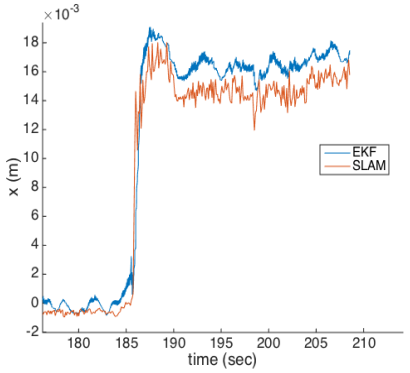
curacy of SLAM seems to be much better before the movement. Before the movement SLAM is less noisy than the EKF seen in Figure 4.9. This can be because the quality of the map is better at the position prior the motion than afterwards. Secondly, SLAM seems to be more noisy during the motion, especially for the y_N-axis seen in Figure 4.9d. This is expected, since SLAM is less accurate during fast motions.



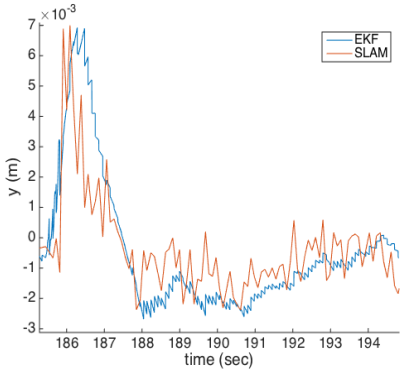
(a) x_N *position of EKF and SLAM*



(b) y_N *position of EKF and SLAM. The accuracy of SLAM is much better before the motion.*



(c) x_N *position of EKF and SLAM zoomed in.*



(d) y_N *position of EKF and SLAM zoomed in. EKF is a bit more smooth.*

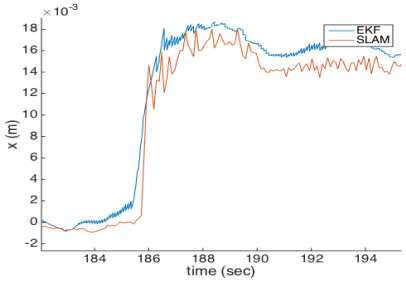
Figure 4.9: *position of the* x_N *and* y_N*-axes from fast translation*

Translation without measurement update

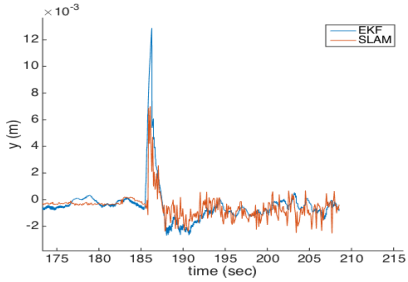
In this test the measurement update was turned off during the translation. Comparing Figures 4.10c and 4.8a, the estimate of z_N seems better without the measurement update. It looks like SLAM, but is ahead in time because of the delay of the SLAM algorithm. The position of y_N for the EKF starts to drift seen in Figure 4.10b and the error after 1 second is around 5-6 mm.

Discussion translation

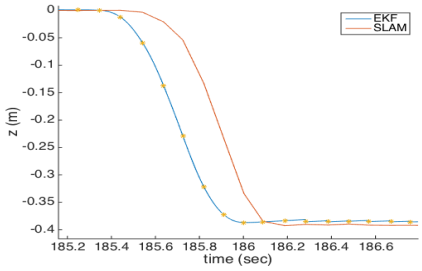
The delays of SLAM can negatively affect the virtual reality experience and it is therefore desirable to mitigate these effects. The EKF could estimate the pose



(a) x_N *position of EKF and SLAM*



(b) y_N *position of EKF and SLAM. The y-position is starting to drift*



(c) z position of EKF and SLAM.

Figure 4.10: *position of the x−, y− and z-axes from fast translation without measurement update.*

with negligible delay since it uses an IMU. However, for the EKF to work well the delays of the SLAM has to be accounted for in the measurement update, otherwise the measurement update will do more harm than good.

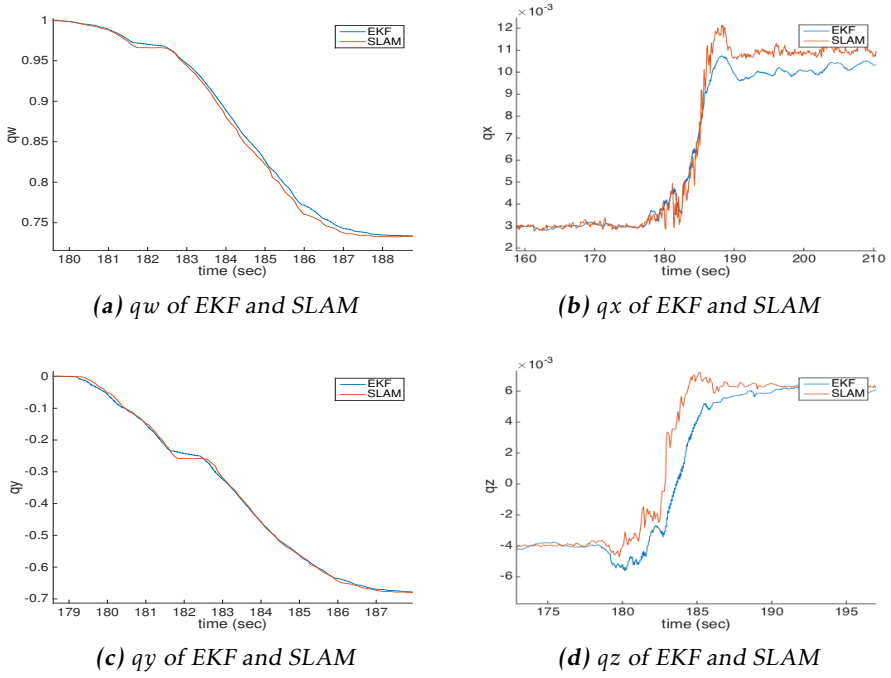
During the test a constant value of R, the measurement noise has been used. This might not completely reflect the reality. The accuracy of the output of SLAM depends on the quality of the map which varies. Also, the accuracy of SLAM depends on the motion of the phone, since fast motions will lead to more rolling shutter effects and motion blur. It is therefore reasonable to assume that a dynamic R should be used for accurate tracking.

4.2.2 Rotation

As mention in Section 3.2.2 two rotation tests were performed. One where the measurement update was used the whole time and one where the measurement update was turned off during the rotation.

Rotation with measurement update

The quaternion components of the SLAM and EKF can be seen in Figure 4.11 and are very similar. Quaternions are rather hard to interpret and to easier understand the difference, (2.14) was used to plot the angle difference. This is seen in Figure 4.12.



(a) qw *of EKF and SLAM*

(b) qx *of EKF and SLAM*

(c) qy *of EKF and SLAM*

(d) qz *of EKF and SLAM*

Figure 4.11: *quaternion components of the EKF and SLAM*

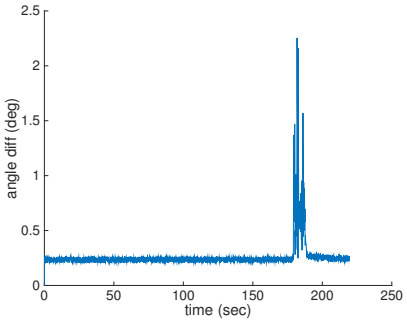


Figure 4.12: *Angle difference SLAM and EKF in degrees.*

In Figure 4.13 the position can be seen zoomed around the time of the rota-

tion. The position in x_N-direction is supposed to steady increase as SLAM does. EKF seems to follow well the first 2 seconds from time 180 to 182 but then starts to diverge from SLAM. Even after time 186, when the rotation has ended, EKF does not converge to SLAM. The saw shaped behaviour of x_N-position of the EKF after time 186 is due to the measurement update trying to correct the position to SLAM and then the time update drifts away in the wrong direction between the measurement updates. The same behaviour can be seen for the z_N-axis in Figure 4.13c. This behaviour originates from the gravity being projected on the wrong axis. The accelerometer measurements resolved in the N-frame can be seen in Figure 4.14. After the rotation the acceleration of the z_N-axis and x_N-axes differ from 0 which is not true when standing still. The conclusion is that the orientation of SLAM that is used to support the EKF is not good enough. Even if the orientation is not off much, small angle errors influence the acceleration very much, mention earlier, see (4.2) and (4.1). As discussed earlier the angle error does not affect the y_N-axis as much and no saw shape behaviour in Figure 4.13b is seen.

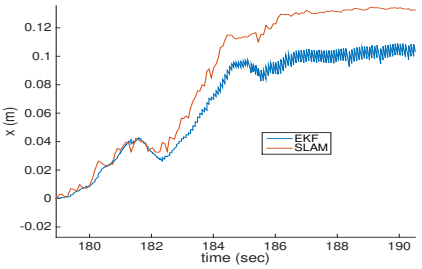
Difficulties for SLAM during rotations can be seen for the z_N-axis and y_N-axis. The position of the z_N-axis is supposed to decrease to 3.5 cm without any valley around time 183 in Figure 4.13c. However, the EKF follows SLAM too much and can not offer any improvements. The position of the y_N-axis around time 182 jitters substantially for SLAM, seen in Figure 4.13b. In virtual reality jitter is unwanted since it makes the user nauseous. The jitter can be avoided by the EKF, which offers a much more smooth movement around this time.

Rotation without measurement update

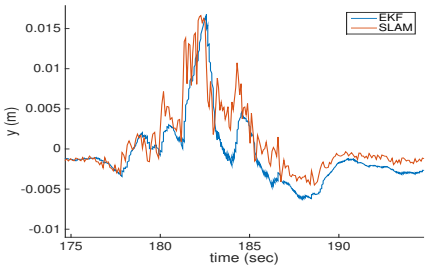
In this test the measurement update was turned off during the rotation from time 179.8 to 186. The orientation of the EKF is still very good, but the position drifts away fast. The result shows promising result for dead reckoning of the gyroscope, but the position drifts away fast and it would not be possible to use the accelerometer for dead reckoning for this long, seen in Figure 4.17. However, the drift after 1 second is not especially large seen at time 180.8 in Figure 4.17.

4.2.3 Rotation lost

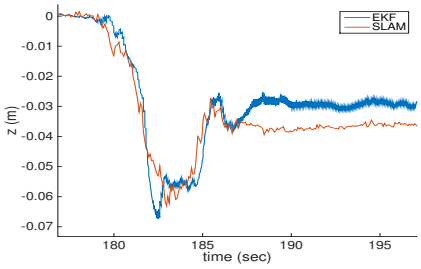
When rotating the phone 45° around the negative y_N-axis the quaternion is supposed to be $q = (\,0.92\ 0\ -0.38\ 0\,)$, see Section 2.3.2. As seen in Figure 4.18 the EKF tracks the rotation good. SLAM however, loses tracking around time 147.2 seen in Figure 4.18c and the user would not get any information about the rotation after that. The estimated position is not evaluated since the EKF is not able to track the position for lower angular velocity tested in the test above. In this test, when rotating fast the delay of SLAM is also visible in Figure 4.18c, where SLAM starts to decrease a while after EKF seen around time 147.



(a) x_N *of EKF and SLAM. EKF does not converge to SLAM, due to error in the orientation of SLAM.*



(b) y_N *of EKF and SLAM. SLAM jitters more around time 182 and is avoided by the EKF*

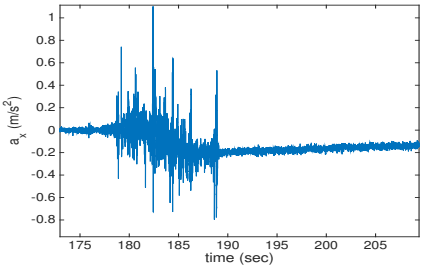


(c) z_N *of EKF and SLAM.*

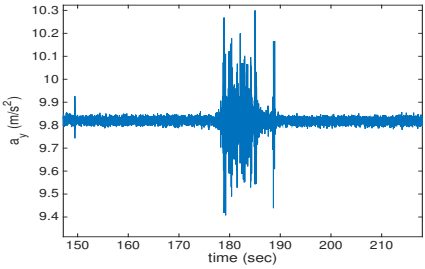
Figure 4.13: *position of the SLAM and EKF*

Discussion rotation

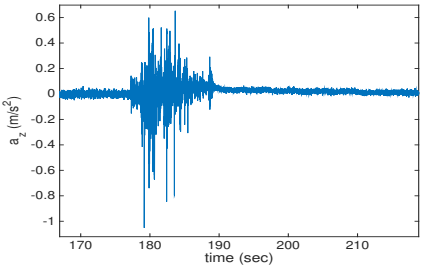
The EKF seems to be able to track orientation well. Even the dead reckoning of the gyroscope seems accurate, since it could still track the orientation when the measurement update was turned off for 6 seconds and when SLAM was lost. However, the position is very hard to track when rotating and could not offer much improvements. Sometimes a more smooth motion could be seen when SLAM was very shaky. The bad tracking of the position after the rotation was completed was because of gravity being projected onto wrong axes. Using the EKF to track the orientation could also avoid delays which would the make the user to experience the virtual world to not lag behind the user's motion.



(a) *x_N component of gravity. Supposed to be 0.*

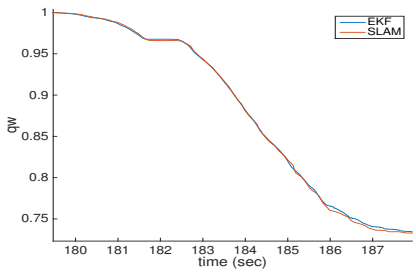


(b) *y_N component of gravity. Supposed to be 9.82*

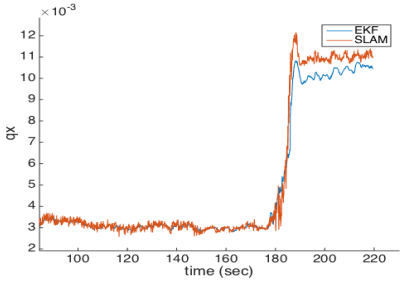


(c) *z_N component of gravity. Supposed to be 0.*

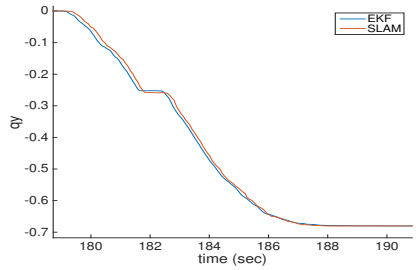
Figure 4.14: *Gravity resolved in coordinate system N*



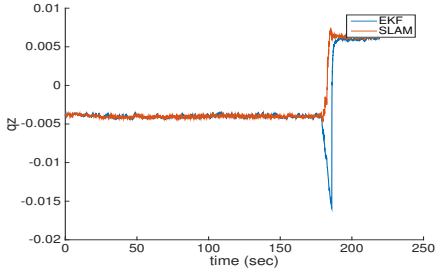
(a) q_w *of EKF and SLAM*



(b) q_x *of EKF and SLAM*



(c) q_y *of EKF and SLAM*



(d) q_z *of EKF and SLAM. Starts to drift, but the angle difference is still low, see Figure 4.16*

Figure 4.15: *quaternion components of the EKF and SLAM when not using the measurement update*

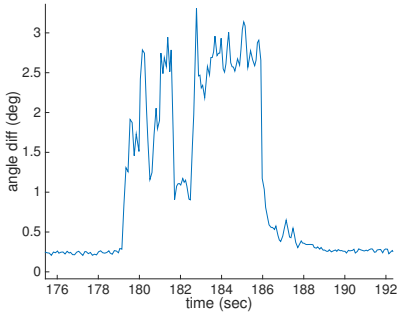
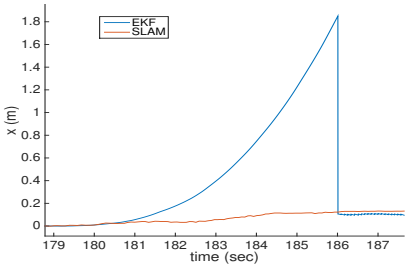
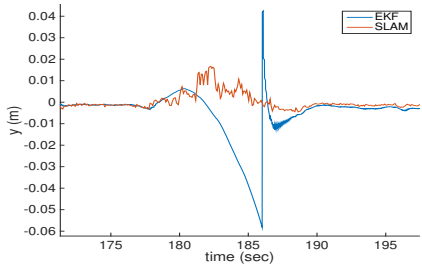


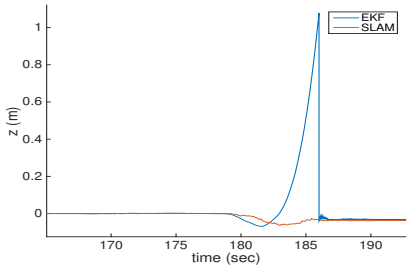
Figure 4.16: *Angle difference SLAM and EKF in degrees when not using the measurement update*



(a) x_N *of EKF and SLAM*

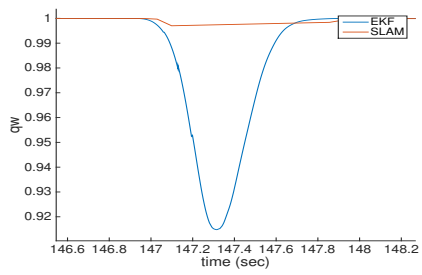


(b) y_N *of EKF and SLAM*

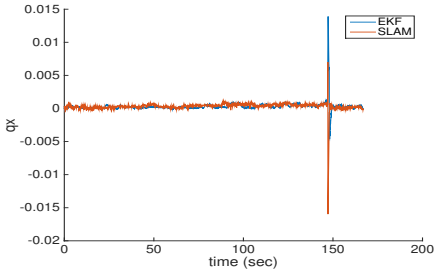


(c) z_N *of EKF and SLAM*

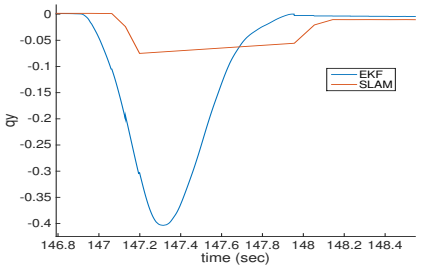
Figure 4.17: *Position of the SLAM and EKF when not using the measurement update. The position of all axes quickly starts to drift.*



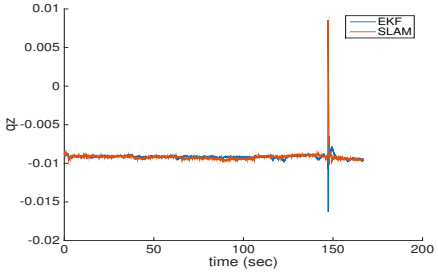
(a) qw of EKF and SLAM



(b) qx of EKF and SLAM



(c) qy of EKF and SLAM. SLAM starts to notice the rotation, but then get lost. The delay of SLAM is also visible in the figure.



(d) qz of EKF and SLAM

Figure 4.18: *Quaternion components of the EKF and SLAM when rotating fast such that SLAM get lost*

5
Conclusion

This section will summarize the results and the conclusions. Future work is also discussed as a consequence of the conclusions drawn.

5.1 Allan variance

The Allan variance was a good method to estimate the white noise of the IMU. It showed consistent results for all datasets, which was not the case for the random walk. The Allan variance did not seem to be as reliant when estimating the random walk. It showed different results for different datasets. Also, the Allan variance is less accurate for long averaging intervals T, where the random walk is found. The accuracy for long averaging intervals is low since the number of independent clusters are small [14]. For future work the length of the datasets could be extended to see if it affects the results.

The values of the noise from the Allan variance could not be directly used in the EKF in order for the EKF to work well. These parameters had to be tuned manually and in general the parameters for the white noise had to be increased while the parameters of the random walk had to be decreased. In several tests the parameters of random walk were set to 0, resulting in a constant model for the bias instead of a random walk model.

5.2 Calibration of camera and IMU coordinate system

The toolbox can accurately find the relative orientation. However the relative translation is much harder to find. For future work the author recommends adding a rolling shutter rectification model, to decrease the rolling shutter effects.

This would not only increase the accuracy of the pose estimation, but also allow for faster rotation increasing the observability of the relative pose between the sensors. The rolling shutter rectification model could also be used in the SLAM algorithm to increase its accuracy.

When rotating, different locations in the phone accelerates differently. Thus, an inaccurate estimate of the relative translation makes the motion model inconsistent when rotating. This could be a reason of worse tracking when rotating, but from the results there are hard to draw any conclusions about this affect. To analyse this better, datasets with groundtruth are probably necessary.

5.3 Tracking

This section brings up the pros and cons of the EKF and recommendations of future work to solve current problems.

5.3.1 Orientation error

The EKF has more problem to track the translation than rotation. The tracking of translation gets more problematic when rotating, since small errors in orientation affects the translation estimate. This is because the gravity vector needs to be known since accelerometers can not distinguish accelerations of body from gravity [25]. Thus, the system is very sensitive to errors in the orientation. [25] discusses the sensitivity of this error, but does not thoroughly explain if the problem have been handled in a special way. However, [25] says ”a preferable method is to record accelerometer measurements while scanning the scene and include this data in the model building procedure to align the scene model vertically.”, implying that a tightly coupled solution could handle this problem.

Another method would be to include either the gravity vector [30] or the drift of the map's orientation in the states of the EKF. This would lead to information only of the current estimate and could lead to decreased performance of the EKF when quickly moved to a new location in the map. If moved quickly, it is likely that the EKF's new estimate of the gravity vector of the drift has not converged to a good value. This could be avoided if the estimate of the gravity vector or drift is continuously saved to the map and can be loaded in these cases, instead of using the EKF's current estimate. This is only possible if the location is revisited otherwise no old estimate is available.

5.3.2 Time delay

The results shows that the EKF reacts faster to motions than SLAM. Thus, the EKF can potentially lower the feeling of lagging behind reality, which can cause motion sickness. However, at the current state the EKF does not handle the delays of SLAM in the measurement update, which causes problem. To avoid this

problem the recommendation of future work is to implement the solution proposed by [21] in the EKF.

The camera and IMU runs on different clocks and to use the solution in [21], the clocks first needs to be synchronized. This has actually already been done in the calibration of the relative pose of the camera and IMU, see Section 2.4.2. However, the time offset, d, is different at every start up of the system and has to be calibrated online. The relative pose of the camera and IMU, was done offline using the toolbox KALIBR. The toolbox, can of course be used in a calibration step before running the EKF, without turning off the phone. Since the toolbox could not estimate the translation correctly, the temporal and spatial calibration might preferably be separated to avoid that error in one estimate to impair the other. This is done in [34], which uses cross-correlation to align the absolute angular velocity estimate of the camera and IMU. This is actually also done in KALIBR to get a good start value of d. Absolute angular velocity is used since it is independent of which coordinate systems it is resolved in. Another property is that a rigid body, like a phone, rotates with the same angular velocity across the whole body. This is important since the camera and IMU are not located at the same position. An IMU measures angular velocity directly, while a camera estimates orientation which can be differentiated to get angular velocity, see [34]. This requires the estimated orientation to be good. Both KALIBR and [34] uses calibration targets to achieve this. In this way, the user would need an own calibration pattern and perform the calibration on his/her own. For consumer products it is desirable to simplify for the user and avoid cumbersome calibration steps, which the user has to conduct himself. A calibration pattern could be avoided, by letting SLAM estimate the position with either a loaded map or like in the most general case with a map that SLAM builds continuously. Using a calibration pattern would likely improve the result and it has to be investigated if such a pattern could be omitted. A drawback of cross-correlation is that the time offset needs to be short, otherwise the overlap of the camera and IMU data sequences might become too small [34].

5.3.3 Dynamic accuracy of SLAM

Since the accuracy of SLAM varies with different motions and with the quality of the map the recommendation is to use a dynamic R. This would make the EKF "trust" the sensors more correctly and increase the performance of the EKF. [41] gives a solution to this problem, using the Jacobian of the optimization problem (2.22) to calculate the covariance matrix R dynamically. The solution is a standard approach in non-linear parameter optimisation [41].

5.3.4 Computational complexity

Since the EKF could track the orientation very well, one idea is to use the EKF to only track the orientation and let SLAM track the position. The EKF could track the orientation very well during longer periods of time without aid from

SLAM. Thus, SLAM could concentrate on estimating the position and only sometimes calculate the orientation to aid the EKF to prevent the EKF's estimate to drift away. SLAM could use EKF's estimate of the orientation in the optimization problem 2.23 and only sometimes perform the full optimization problem for both the orientation and position. This would decrease the complexity of the optimization problem and lead to lower usage of the CPU, lowering the emission of heat and the battery consumption.

5.3.5 Temperature transient

If the accelerometer should be used to track the position it is also important to account for the "temperature transient", while the gyroscope seems to not be affected of the temperature. To use the accelerometer a more complex model that includes the temperature dependency has to be applied. Since different cellphones can be used for virtual reality the temperature model is likely to be specific for each model, leading to a laborious process. Another complication is that many cellphones do not have a thermometer. If a thermometer is available, the relative distance between the thermometer and CPU and the relative distance between CPU and IMU also has to be accounted for, since it is warmer closer to the CPU.

5.3.6 IMU performance

The results indicates that the performance of the gyroscope is very good. The dead reckoning of the gyroscope seems accurate given the results from the bias calibration and the movements tests.

The accelerometer does not seems as accurate given the results. The position error from the bias calibration is higher and also the position from the movement tests is harder to track than the orientation. However, as mention earlier small orientation error affects the position estimate which makes it hard to know if the error is due to the accelerometer performance or other error sources. The drift error of the position in "fast translation without measurement" and "rotation without measurement" after 1 second are not that large and it seems that the accelerometer might be good enough for dead reckoning for shorter periods of time. This requires accurate estimates of the other quantities, like the bias and orientation estimate.

The current solution is not good enough to get accurate estimates at all time. The suggestions of future work, mention above, could possibly fix this and then a more thorough investigation could be done to see if the IMU is good enough to enable tracking with camera measurements at lower frame rates.

5.4 Summary

The EKF could at this stage not offer much improvements for tracking of the position. My advise would be to start looking on a tightly coupled solution which also should account for the delays of SLAM.

The orientation estimate of the EKF is good and makes it possible to still keep track of the user's orientation even during fast rotation, when SLAM gets lost.

If using an EKF to only estimate the orientation, SLAM would not need to calculate the orientation as often, thus lowering the CPU and battery consumption.

Including an IMU offers tracking without latencies, which is one of the more important things to avoid motion sickness.

Bibliography

[1] HTC homepage, 2016. URL https://www.htcvive.com/eu/. Cited on page 4.

[2] Virtual reality society, 2016. URL http://www.vrs.org.uk/virtual-reality-applications/. Cited on page 2.

[3] David W. Allan. Time and frequency (the-domain) characterization, estimation, and predictionof precision clocks and oscillators. *IEEE transanctions on ultrasonics. ferroelectrics, and frequency constrol*, UFFC-34 NO. 6, 1987. URL http://ieeexplore.ieee.org.e.bibl.liu.se/stamp/stamp.jsp?tp=&arnumber=1539968&tag=1. Cited on page 26.

[4] Leopoldo Armesto, Josep Tornero, and Markus Vincze. Fast ego-motion estimation with multi-rate fusion of inertial and vision. *International Journal of Robotics Research*, 2007. URL http://www.cs.ucf.edu/~jjl/pubs/laviola_acc2003.pdf. Cited on page 7.

[5] M. Sanjeev Arulampalam, Simon Maskell, and Neil Gordon. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transcation on signal processing*, 50(2), 2002. URL https://people.eecs.berkeley.edu/~pabbeel/cs287-fa12/optreadings/Arulampalam_etal_2002.pdf. Cited on page 7.

[6] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13 (3), 2006. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1678144. Cited on page 13.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3), 2008. URL https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf. Cited on page 15.

[8] Gabriele Bleser and Didier Stricker. Advanced tracking through efficient image processing and visual-inertial sensorfusion. *IEEE Virtual Reality Conference*, 2008. URL http://ieeexplore.ieee.org.e.bibl.liu.se/stamp/stamp.jsp?tp=&arnumber=4480765. Cited on page 7.

[9] Gabriele Bleser and Didier Strickery. Using the marginalised particle filter for real-time visual-inertial sensor fusion. *International Symposium on Mixed and Augmented Reality*, pages 3–12, 2008. URL http://ieeexplore.ieee.org.e.bibl.liu.se/stamp/stamp.jsp?tp=&arnumber=4637316. Cited on page 7.

[10] Vernon Chi. Quaternions and rotations in 3-space: How it works. *Microelectronic Systems Laboratory, Department of Computer Science, University of North Carolina at Chapel Hill*, 1998. URL http://gamma.cs.unc.edu/courses/planning-f07/PAPERS/quaternions.pdf. Cited on pages 11 and 12.

[11] Bong-Su Cho, Woo sung Moon, Woo-Jin Seo, and Kwang-Ryul Baek. A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding. *Journal of Mechanical Science and Technology*, 2011. URL http://www.j-mst.org/On_line/admin/files/22-J2011-1223.pdf. Cited on page 27.

[12] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors, 2006. Cited on page 11.

[13] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2), 2006. URL https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf. Cited on page 13.

[14] 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), 2008. URL http://ieeexplore.ieee.org.e.bibl.liu.se/xpl/articleDetails.jsp?tp=&arnumber=4404126&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4404126. Cited on pages 17, 18, 33, and 51.

[15] Fixit. Samsung Galaxy S6 edge teardown. URL https://www.ifixit.com/Teardown/Samsung+Galaxy+S6+Edge+Teardown/39158. Cited on page 39.

[16] Inc. Freescale Semiconductor. Allan variance: Noise analysis for gyroscopes, 2015. URL http://cache.freescale.com/files/sensors/doc/app_note/AN5087.pdf. Cited on page 18.

[17] Paul Furgale, Timothy D. Barfoot, and Gabe Sibley. Continuous-time batch estimation using temporal basis functions. *IEEE International Conference on Robotics and Automation*, 2012. URL http://ieeexplore.ieee.org.e.bibl.liu.se/stamp/stamp.jsp?tp=&arnumber=6225005. Cited on pages 20 and 21.

[18] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan.*, 2013. URL https://3234f89137bccf2ede29cc86e315c75116020d70.googledrive.com/host/0B64GJ60h3Ai1MVVwWTZwekhtcFU/publications/bib/furgale_iros13.pdf. Cited on pages 20, 21, 23, 28, and 40.

[19] Paul Furgale, Jerome Maye, and Jörn Rehde. Kalibr toolbox, 2014. URL https://github.com/ethz-asl/kalibr. Cited on pages 20 and 28.

[20] GearVR. Repository gearvr, 2016. URL https://github.com/Samsung/GearVRf/blob/1f036297acd0a2734ba349c4429bdda44cfc5e26/GVRf/Framework/jni/sensor/ksensor/k_sensor.cpp. Cited on page 28.

[21] Fredrik Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur AB, 2:1 isbn 978-91-44-07732-1 edition, 2012. Cited on pages 6, 12, 16, 17, and 53.

[22] C. G. Harris and J. M. Pike. 3d positional integration from image sequences. *In Proceedings of the Alvey Vision Conference*, 1987. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.380.8604&rep=rep1&type=pdf. Cited on page 1.

[23] Richard I. Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2), 1997. URL https://perception.inrialpes.fr/Publications/1997/HS97/HartleySturm-cviu97.pdf. Cited on page 15.

[24] Jeroen Hol. *Sensor Fusion and Calibration of Inertial Sensors, Vision, Ultra-Wideband and GPS*. Linköping studies in science and technology. dissertations. no. 1368, Linköping University, The Institute of Technology, June 2011. Cited on pages 1, 6, 7, 12, 13, 14, and 16.

[25] Jeroen D. Hol, Thomas B. Schön, Henk Luinge, Per J. Slycke, and Fredrik Gustafsson. Robust real-time tracking by fusing measurements from inertial and vision sensors. *Journal of Real-Time Image Processing*, 2007. URL http://user.it.uu.se/~thosc112/pubpdf/holslsg2007-2.pdf. Cited on page 52.

[26] Jeroen D. Hol, Thomas B. Schön, and Fredrik Gustafsson. Modeling and calibration of inertial and vision sensors. *The International Journal of Robotics Research*, 2010. URL http://liu.diva-portal.org/smash/get/diva2:301427/FULLTEXT01.pdf. Cited on page 19.

[27] Joseph J. LaViola Jr. A comparison of unscented and extended Kalman filtering for estimating quaternion motion. *American Control Conference*, 3:2435–2440, 2003. URL http://www.cs.ucf.edu/~jjl/pubs/laviola_acc2003.pdf. Cited on page 7.

[28] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. Cited on page 7.

[29] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960. Cited on pages 7 and 8.

[30] Jonathan Kelly and Gaurav S Sukhatme. Visual–inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 2011. URL http://ijr.sagepub.com.e.bibl.liu.se/content/30/1/56.full.pdf+html. Cited on page 52.

[31] Stefan Leutenegger, Paul Furgale, and Vincent Rabaud. Keyframe-based visual-inertial SLAM using nonlinear optimization. *The International Journal of Robotics Research*, 2014. URL http://ijr.sagepub.com/content/early/2014/12/12/0278364914554813. Cited on page 6.

[32] Jorge Lobo and Jorge Dias. Relative pose calibration between visual and inertial sensors. *The International Journal of Robotics Research*, 2007. URL http://ap.isr.uc.pt/archive/155.pdf. Cited on page 19.

[33] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004. URL https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf. Cited on page 15.

[34] Elmar Mair, Michael Fleps, Michael Suppa, and Darius Burschka. Spatio-temporal initialization for imu to camera registration. *In 2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 557–564, 2011. URL http://www6.in.tum.de/Main/Publications/mair2011b.pdf. Cited on page 53.

[35] Faraz M. Mirzaei and Stergios I. Roumeliotis. A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. *IEEE TRANSACTIONS ON ROBOTICS*, 24, 2008. URL http://www-users.cs.umn.edu/~stergios/papers/TRO_08-IMU-Camera-calibration.pdf. Cited on pages 16, 19, and 28.

[36] M.M.Tehrani. Ring laser gyro data analysis with cluster sampling technique. *Proc. SPIE 0412, Fiber Optic and Laser Sensors I, 207*, 0412, 1983. URL http://proceedings.spiedigitallibrary.org.e.bibl.liu.se/proceeding.aspx?articleid=1235651. Cited on page 17.

[37] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147 – 1163, 2015. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7219438. Cited on page 4.

[38] Thomas Schön, Fredrik Gustafsson, and Per-Johan Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transcation on signal processing*, 53(7), 2005. URL http://users.isy.liu.se/en/rt/fredrik/reports/04SPschon.pdf. Cited on page 7.

[39] Knvul Sheikh and Staff Writer. Beyond gaming: 10 other fascinating uses for virtual-reality tech. *Live Science*, 2016. URL http://www.livescience.com/53392-virtual-reality-tech-uses-beyond-gaming.html. Cited on page 2.

[40] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Real-time monocular SLAM: Why filter? *IEEE International Conference on Robotics and Automation*, 2010. URL https://www.doc.ic.ac.uk/~ajd/Publications/strasdat_etal_ivc2012.pdf. Cited on page 1.

[41] Jean-Philippe Tardif, Michael George, and Michel Laverne. A new approach to vision-aided inertial navigation. *IEEE/RSJ International Conference, Intelligent Robots and Systems (IROS)*, 2010. URL http://www.fieldrobotics.org/users/alonzo/pubs/papers/iros2010final.pdf. Cited on pages 1, 6, 7, and 53.

[42] David Titterton and John Weston. *Strapdown Inertial Navigation Technology*. IEE radar, sonar, navigation and avionics series. Peter Peregrinus Ltd., Stevenage, UK, 1997., second edition, 1997. Cited on page 12.

[43] Alexander A. Trusov. Allan variance analysis of random noise modes in gyroscopes, 2011. URL http://www.alexandertrusov.com/uploads/pdf/2011-UCI-trusov-whitepaper-noise.pdf. Cited on page 17.

[44] The free encyclopedia Wikipedia. Rolling shutter, 2007. URL https://en.wikipedia.org/wiki/Rolling_shutter. Cited on page 3.

[45] The free encyclopedia Wikipedia. Motion blur, 2008. URL https://en.wikipedia.org/wiki/Motion_blur. Cited on page 3.