Großer Beleg

Real-Time Structure from Motion Using Kalman Filtering

By Jeannette Bohg
Born on 15th March 1981 in Cottbus

Submitted on 21st March 2005
Overseeing Professor: Steffen Hölldobler
Supervisor: Tobias Pietzsch

Abstract

Simultaneous localization and mapping *(SLAM) is one of the core problems in robotics. Its central theme is the estimation of the location and motion of a mobile robot situated in an unknown environment and the simultaneous construction of a map of this environment.*

In this work, the problem is tackled by using vision sensors. Therefore, classical Structure From Motion *(SFM) techniques, which have a long history in computer vision, are applied to SLAM. In general, we can distinguish two major approaches to SFM. Firstly, we have techniques based on the recursive Kalman Filter where time dependend sequences of images are required. The second kind of SFM-methods utilizes the constraints in multiple view geometry. The sequence of the images considered is independent of time.*

One of our main objectives is to obtain real-time performance which is required in the field of robotics rather than offline processing.

We build upon the results by Tobias Pietzsch [24] who presented promising results using the second kind of SFM algorithms specifically tailored for real-time performance.

In this work we will analyze the usage of Kalman Filtering concerning the task of SLAM and compare it with methods that utilize the multiple view constraints. As a sensor we will first use a single camera and then compare the results with the performance of a stereo camera. We track 3D point features as landmarks.

Contents

Chapter 1

Introduction

One of the most important abilities for mobile robots is navigation. It enables them to perform complex tasks in real-world environments. Consider for example a mobile robot in an office environment who has to deliver mail as presented in [19]. Without the ability to navigate, it will not be able to fulfil its task.

Nehmzow [34] proposed to devide navigation into three subtasks:

- Self-localisation
- Mapping
- Routing

Self-localisation is the ability of an autonomous robot to estimate its own position within a coordinate frame. Generally speaking, localisation techniques can be divided into two basic categories: *Landmark-based methods* and *Dense Sensor Matching*. The latter uses all available sensor information and compares a dense sensor scan with an *a priori* given surface map of the environment. In contrast, landmark methods rely on the recognition of landmarks in the environment. In this work, we will concentrate on this technique.

Landmarks need to be stationary and preferably distinguishable so that the robot recognises them after periods of neglect. The robot's position is estimated with respect to them. Therefore, the position of these landmarks within a global coordinate frame needs to be known as well. They can either be given in advance or approximated by the robot itself. The latter implies that measurements of features serving as landmarks inside this environment have to be taken and are used to estimate their position, which means to build a map. Thus, the problem of self-localisation is directly linked to mapping. Here, the term map describes the mapping of entities in the real world to entities in an internal representation. Concerning vision sensors this internal representation might consist, for example, of point or line features.

Routing can be seen as an extension of localisation because the robot must determine the start as well as the target position inside the environment.

In this work we will focus on the first two problems of navigation, summarised in the term of *Simultaneous Localisation and Mapping* (SLAM) or also known as *Concurrent Mapping and Localisation* (CML). The classical solution

to this problem is based on the *Extended Kalman Filter* (EKF SLAM) which was suggested by Smith et al. [28]. Here, it is assumed that the robot moves in an environment with fixed landmarks that can be measured by distance sensors. The position of the landmarks as well as of the robot at a particular point in time are summarised in the term of the *system's state*. The task is to estimate the new state for the next point in time given the last motion of the robot and new observations provided by the sensor.

The last movement is usually estimated by odometry data as for example in [10].[1] Traditionally, sensors to measure the landmarks were range-only sensors such as *sonar rings* [25] or *laser range finders* (LRF) [14]. Usually, we need to model the robot's motion and its sensors to be able to provide accurate estimates of the system's state. EKF SLAM is based on a stochastic model that involves the error concerning the position of the robot as well as of the landmarks. These errors are assumed to be independent and to have a Gaussian profile.

Generally, the classic approaches to SLAM using range-only sensors, are restricted to 2D planar robot motion and mapping. This is due to the fact, that sonar rings and LRFs just provide measurements all situated on the same height level. Hence, movements over rough or undulated terrain are not directly supported. Recently, we find approaches attempting to build a map in three dimensions (3D-SLAM) using either two planar laser scanners [15] (one pointing at the horizontal direction, the other at the vertical direction) or a single 2D laser scanner equipped with a servo motor [4]. With the increase in computing power we also find 3D-SLAM approaches using vision sensors. Here, we can distinguish between monocular cameras and stereo cameras.

In contrast to LRF's, vision systems do not perform well under certain environmental conditions such as regions with an uniform appearance, large metallic or transparent surfaces and poor lightning conditions.[2] This is caused by the fact, that an image not just depends on the *geometry* of the scene considered, but also on its *photometry*. Besides the negative aspect of this dependency, there are also advantages. With a vision sensor, we are able to percept the surface properties of objects such as colour and material. LRFs just provide spatial information and, inferred from the intensity of the reflected rays, also rough information about the surface [5].

The whole complexity of the physical world cannot be captured with either of these sensors. We need to concentrate ourselves on the properties that are of interest for the specific task we want to solve. For the properties not perceptible, assumptions and hypotheses should be introduced. The result of this process is either a model or an internal representation of the world. What kind of model we infer, depends on whether for example we want to move within the environment, visualise it from several viewpoints or recognise objects and materials. With regard to the future goal of developing an autonomous mobile robot that is able to solve complex tasks in real-world environments, we need to achieve that it is able to navigate as well as to recognise objects of interest. Thus, in the long term the usage of a camera as a sensor to solve the 3D-SLAM problem is more promising with regard to the additional ability of object recognition.

[1] Odometry data contains information about the covered distance from a certain starting point based on the number of wheel revolutions.

[2] An LRF is also not able to recognise a glass surface

Kalman Filter based techniques to Structure from Motion have the advantage that they are proven to be real-time compliant and that they take the noise in the measurements into account. Here, not just the position of the robot and the landmarks is estimated but additionally the uncertainty about this current estimate is calculated. This information is essential to know for the robot to be able to move without risk through an unknown environment. To reconstruct the position of the robot and the landmarks at the current point in time the previous estimate and current measurements are involved in the calculations to form the new estimate. Thus, all subsequently taken measurements are also indirectly involved at every point in time by the previous estimate. In 2003, Andrew Davison [9] applied this approach to a hand-waved single camera and achieved remarkable results.

In this work, we will first review and re-implement Davison's approach with a single camera. The problem with using a monocular vision sensor is that the 3D geometry and ego-motion can just be recovered up to a scale factor. Instead of real distances, we just have relative positions. This problem can either be solved by calibrating the camera with an object of known size at the beginning of the navigation process or by using a stereo camera. With the latter, we are able to determine an estimate of the absolute depth of a landmark just from the two grabbed images at a certain point of view. After substituting the stereo camera for the monocular sensor, we will compare the results obtained by Tobias Pietzsch, who proposed a Structure from Motion technique based on Multiple View Geometry that was specifically tailored for real-time processing.

To constrain the complexity of the problem some assumptions are made.

Corresponding projections of 3D points are determined. The problem of identifying corresponding projections of a world point between several monocular images or between stereo image pairs is solved by a feature tracker. This feature tracker is no subject of this work and will therefore be treated as a black box for the monocular case as well as for the usage of a stereo camera. In [24], you can find detailed information about the implementation of a feature tracker for a single camera.

The pinhole camera model is valid. Most cameras suffer from radial lens distortion. We assume here, that it is negligible or already removed in a preprocessing step.

The world is static. For the robot the 3D point features anchor the surrounding. Its position is related to them in the environment. Therefore, if a features moves, the position of the robot will also assumed to have changed although it has not moved in reality. To avoid the task of distinguishing between static and non-static point feature, we assume that there are no moving objects in the scene.

The initial position of the camera and a number of features is known. In this work, we assume that the robot initially knows its position within the environment and a certain number of features. In our future work we

will analyse initialisation techniques for the case of SLAM with a single and stereo camera.

This work is structured as follows: In the Chapter 2, we will give an overview about existing solutions for the problem of SLAM. In Chapter 3 basics of projective geometry and camera models are presented. The Kalman Filter as well as the Extended Kalman Filter (EKF) are explained in Chapter 4. Applications of the EKF to SLAM with a single camera and with a stereo camera are given in Chapter 5. In Chapter 6 we will analyse the complexity of the EKF and methods to reduce it. In Chapter 7 the results of several experiments on simulated data are presented. The conclusions are drawn in Chapter 8.

Throughout this work, we expect the reader to be familiar with basic concepts of stochastics such as *standard deviation, variance* or *covariance*.

Chapter 2

Related Work

The first mobile robot that used a vision sensor to act in its environment was SHAKEY, developed in 1969 by Nils Nilsson at the Stanford Research Institute. It operates in highly constrained environments just containing isolated prismatic objects, as it can be seen in Figure 2.1.

A good impression of the performance of SHAKEY is provided by the statement from Moravec in the book *Robots* [21]:

> On a very good day it could formulate and execute, over a period of hours, plans involving moving from place to place and pushing blocks to achieve a goal.

Actually, most of the execution time was needed for vision processing. At the time of SHAKEY, real-time performance in the field of vision based navigation was unthinkable facing the massive computational costs.

The problem of inferring 3D structure from a set of 2D images has a long history in the field of computer vision and is known as *Structure from Motion* (SFM). In general, we can distinguish two approaches to SFM. Both have in common that they track features serving as landmarks, regardless in which form (e.g., points or lines), through multiple images.

Firstly, there are techniques that utilise the constraints in multiple-view geometry. Generally speaking, we can determine the relative orientation and distance between two camera positions if we have given a number of corresponding points in each image grabbed at the considered camera positions.

Secondly, we have recursive Kalman Filter based methods referred to as *SFM filtering*. Here, a 3D model is constantly updated over time by the currently grabbed image of a sequence.

In the field of Computer Vision, the unfavourable computational circumstances led to a focus on offline or batch SFM techniques where three-dimensional models of objects or rooms are built *after* the camera has captured images of them.They fell in the first category of SFM-techniques. Global optimisation methods are applied where all pictures are taken into account. These images had been grabbed and saved on a computer during the robot had moved and before they were processed. The result is optimal in the sense that it is consistent with all available measurements at every point in time. An offline

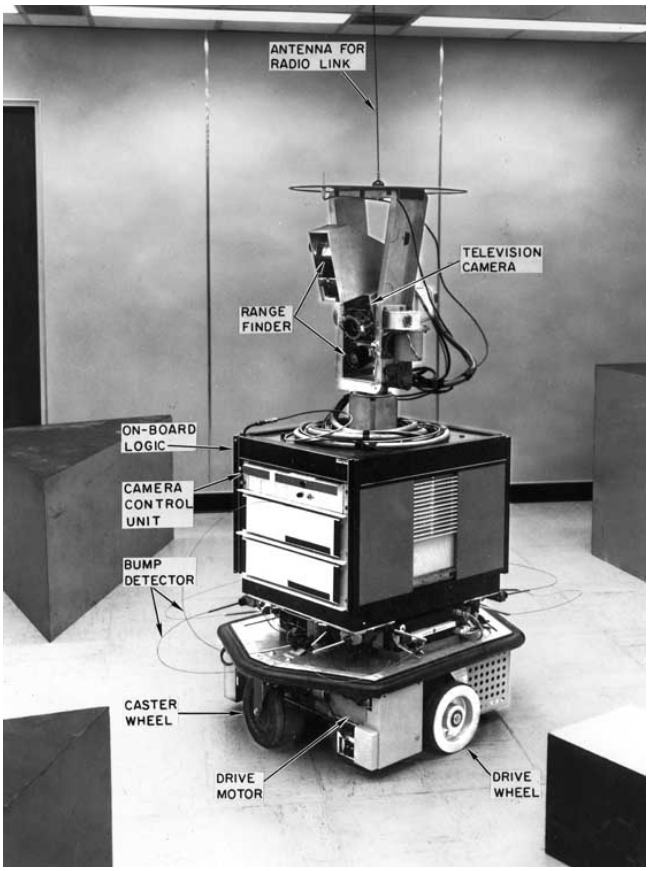

Figure 2.1: SHAKEY, *the robot. Developed in 1969 by Nils Nilsson at the Stanford Research Institute. Picture adapted from [21].*

technique which is mostly applied in the field of photogrammetry is Bundle Adjustment ([18], [33]). Triggs et al. [33] defined it as follows:

> Bundle Adjustment is the problem of refining a visual reconstruction to produce *jointly optimal* 3D structure and viewing parameter (camera pose and/or calibration) estimates. *Optimal* means that the parameter estimates are found by minimising some cost function that quantifies the model fitting error, and *jointly* that the solution is simultaneously optimal with respect to both feature and camera positions.

Here, also the noise in the previously obtained measurements is involved.

In the field of robot navigation, more specifically in the field of SLAM, traditional approaches used basically range-only sensors such as sonar rings and laser range finders (LRFs). The large amount of image processing to obtain spatial information is unnecessary because LRFs directly provide depth measurements. Assuming a horizontal alignment of the considered sensor, the measurements are all located on the same height level and thus, restricted conditions like 2D planar robot motion and mapping were used, e.g., in [14] or [10]. We summarise these methods in the term *2D-SLAM*.

The classic solution for SLAM is based on the *Extended Kalman Filter* (EKF-SLAM) and was suggested by Smith et al. [28]. In 2002, another approach using LRF's was developed. *FastSLAM* by Montemerlo et al. [20] is based on *Monte Carlo localisation* (MCL) and does away with some of the assumptions of the EKF based approaches to be able to cope with a larger number of landmarks. FastSLAM scales better than EKF-SLAM. Nevertheless, depending on the number of known landmarks both 2D-SLAM techniques are real-time compliant.

Recently, there is a general tendency in robot navigation to 3D maps. In contrast to 2D-SLAM, where we have three Degrees of Freedom (DoF), 3D-SLAM allows us to estimate six DoF.[1] This enables to cope with uneven terrain or to detect obstacles that are smaller than the height of the 2D-LRF measurements.

To be able to percept landmarks in three dimensions, we need to apply 3D sensor devices. First of all, we have 3D laser scanners. They either consist of two 2D LRFs (one vertically aligned, the other horizontally aligned) as for example in [15] or of a single 2D LRF equipped with a servo motor. Using the latter, Nüchter presented imposing results in his diploma thesis [4].

Secondly, vision sensors are also able to percept 3D landmarks. Thanks to the rapid development in imaging and computing hardware, nowadays it is possible to transfer a full-resolution image into the memory of a computer at a frame rate of 30 Hz and process the data in real-time. Also the geometry of vision has been understood to the point that full spatial information can be reconstructed by using a stereo camera or a sequence of pictures taken by a monocular camera.

To enable a mobile robot to navigate purely vision-based, the formerly

[1] 2D-SLAM:2D translation and yaw; 3D-SLAM:3D translation, roll, pitch and yaw

offline processing SFM-techniques to reconstruct 3D information from images, have to be adapted in order to obtain real-time performance. The collection of a batch of images before processing them, requires motion of the considered robot. This compromises the safety of navigation. How should the robot know how to avoid an obstacle when it obtains the necessary information after it has already collided? Thus, the data-gathering delay cannot be forgiven. If we assume that just one image after each time step is taken into account, we will also not achieve real-time compliance. Every time a new image had been grabbed the whole sequence of images has to be reconsidered for the re-calculation of the scene. Thus, the longer the period of navigation the longer is the calculation process. Instead of batch methods, we prefer "windowed" or recursive versions of offline SFM-techniques.

"Windowed" methods are characterised by just processing a small and constant number of images instead of the whole sequence such as in Bundle Adjustment. They are also based on the geometry of multiple views. For example, in the diploma thesis of Tobias Pietzsch [24] a method, namely "Sequential Alignment of Image Triples", was analysed and improved by exploiting additional information from redundant 2-view motion estimates. The appropriate application fulfils the request for real-time performance. It is based on the 8-point algorithm by Longuet-Higgins [13] for reconstructing a scene from two arbitrary projections by just using eight corresponding points in each image. This algorithm is extended to enable the reconstruction of the whole trajectory of a mobile robot by a sequence of images captured at different positions en route. The main problem is, that the error in each estimate for distance and relative orientation between two views accumulate over time and causes a motion drift.

Error in the estimates is basically caused by inaccurate or respectively noisy measurements. To diminish this problem we can try to reduce the noise, e.g., by using a stereo camera to derive absolute 3D positions of the considered points, and we can handle the noise explicitly by using the second form of SFM-techniques: the recursive Kalman Filter based methods.

In contrast to the multiple-view based approaches, the latter exploits the chronological ordering of the obtained images and the fact that motion is subjected to a certain degree of regularity. If, after time step k, the vision sensor is situated at a known position somewhere in the room, it will not jump to a completely different location after the next time step $k+1$. Rather, it will have moved just slightly to a position near the old one. Motion depends on forces, inertia and other physical constraints. The same we have with the process of obtaining images from a 3D scene. Perspective projection is not arbitrary but underlies certain laws.
Known or measurable forces or accelerations can be used in a model in order to obtain an approximation of 3D structure and motion. Constraints that are unknown, unmeasurable or too complex are treated as uncertainty in the model. Uncertainty in forces or accelerations is represented as the realisation of a normally distributed random vector with zero mean.

To sum up, in Kalman Filter based SFM-methods, we have a dynamical system whose unknown *state* describes the structure and motion at a certain point in time. Its "output" are the measurable images. The task is to infer the system's state given its output in the presence of uncertainty. Just one picture by a monocular camera or one stereo image by a stereo camera is needed each point in time to apply this technique.

	AIS 3D laser scanner	**bumblebee™**
Field of Range	180° (horizontal) 120° (vertical)	100° (horizontal and vertical)
min. Resolution	16,200 point measurements in 1.35 seconds	640 × 480 pixels at 30 Hz frame rate
Size	350 × 240 × 240 mm	160 × 40 × 50 mm
Weight	4.5 kg	375 g
Power Consumption	20 W + 12 W Servomotor	2.1 W
Price	8,500 €	2,000 €

Table 2.1: *AIS 3D laser scanner versus bumblebee™. Specifications for the LRF adapted from [31] and for the stereo camera from [2].*

Here, we have a similarity between SFM-techniques and classical SLAM-methods: the Kalman or Extended Kalman Filter.

Recently, Chiuso et al. [3] presented a real-time sequential Kalman Filter based SFM approach that was more focused on the mapping aspect of the SLAM problem. Therefore, 2D projections of 3D points serving as landmarks were tracked over time using a monocular camera. The main drawback of this work is a motion drift that occurs because features will not be re-identified after periods of neglect.

An impressive and successful application of vision based SLAM for a single and a stereo camera using the *Extended Kalman Filter* has been developed very recently by Davison [8], [9]. In contrast to the work of Chiuso, 3D point features were tracked in the environment and can be re-recognised. Therefore, measurements of these "old" features can be used to correct the possible drift in motion.

The question may arise what the differences between 3D LRFs and vision sensors are and where the advantages or disadvantages can be found for each device with respect to 3D-SLAM. In Table 2.1 we listed technical specifications for the *AIS 3D laser scanner* used by Nüchter [4] and for the bumblebee™ stereo camera by Point Grey Research [2]. The LRF is depicted in Figure 2.2, the stereo camera in Figure 2.3. Just considering these technical details, we can state that the camera has advantages over the laser apart from a minor field of range.

Firstly, there is the time aspect. In about the time the laser range finder is able to return the distance for a single direction, bumblebee™provides the colour values of 786 pixels. Of course, we should not forget, that the image data need to be processed to derive spatial information, what can be done in real-time. Another drawback of the laser range finder is its weight and power consumption.

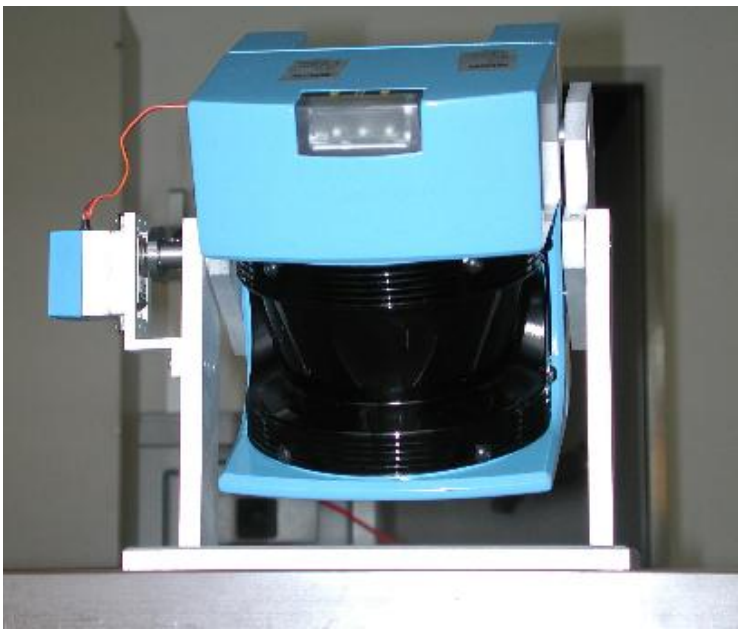

Figure 2.2: *AIS 3D laser scanner. Based on a simple 2D LRF, it is additionally equipped with a servo motor. Picture adapted from [1].*

Figure 2.3: *bumblebee™stereo camera by Point Grey. Picture adapted from [2].*

The LRF actively scans the surface by emitting pulsed laser beams for which mechanical processes are necessary. In contrast to that, a camera just passively measures light intensities. Thus, the usage of a stereo camera will benefit to a longer period of navigation due to a lower power consumption.

Not contained in Table 2.1 are information about the accuracy of each sensor device. Based on the used 2D-LRF the precision of the 3D-LRF amounts to 1 or 5 cm. Additionally, we have half a degree deviation due to the time-critical control of the servo motor. The accuracy of the camera is not measurable that precisely. This is due to the fact, that it depends on the light and surface constraints in the environment. As already mentioned in the introductive chapter, a vision sensor does not perform well under certain environmental conditions such as uniform regions, metallic and transparent surfaces or poor lightning conditions. Then, finding corresponding projections in multiple images becomes harder. This is known as the *data association problem* [32].

On the other hand, the cause for this problem when using vision sensors can also be seen as an advantage over the LRF: an image grabbed by a camera not just depends on the geometry of a scene but also on its photometry. This means that we not just have spatial information as derived by a laser scanner but also colour information. The pictures based on the intensity of the reflected laser beam as presented in the work of Nüchter [4] can not compete with a coloured image of the environment in terms of reproduction of reality. Some special lasers are additionally equipped with digital cameras that contribute colour information to the distance measurements, e.g., in [7]. But this will hike the already higher price of an LRF in comparison to the stereo camera.

To sum up, we can say that the choice of the appropriate sensor device depends on the task to be fulfilled. If we just need to navigate safely in a more regular environment such as an office hall, a 3D-LRF or even a 2D-LRF is sufficient and more accurate than a vision sensor. Colour or surface information are simply not necessary but might rather be disturbing. Facing the future goal of dealing with complex task in a real-time environment, object recognition might be required. Consider for example a mobile robot who is equipped with a vacuum cleaner and a gripper as in [22]. If its task is to clean up the environment, navigating the environment will not be sufficient. The robot will also need to recognise the objects on the floor in order to pick them up and take them to the desired place.

In the near future, cameras can be seen as more flexible and versatile than distance sensors such as laser range finders. The advantageous technical handling and price of such a camera will also contribute to this development.

Chapter 3

Foundations

In this chapter we will provide a basis for the understanding of the following chapters. Thus, in the first section we will introduce the notational conventions used in this work. To understand the application of the two different vision sensors for the problem of SLAM, knowledge about the camera systems is essential. Section 3.2 will provide these information.

3.1 Notation

Throughout this work, the following notational conventions are used.

- Scalars are denoted by italic symbols, e.g., a, b, c.
- Vectors, regardless in which dimension, are denoted in bold lower case symbols, e.g., $\mathbf{x} = (x, y)^\top$. Usually, the dimension of a vector is clear from the context. Nevertheless, this is sometimes declared by subscripts in brackets, e.g., $\mathbf{x}_{(2)}$.
  Sometimes, geometric relations are presented in a projective framework and homogeneous coordinates are used. They are labeled with a tilde superscript, e.g., $\tilde{\mathbf{x}} = (x, y, w)^\top$.
- In this work we will deal with estimates of vectors. They are indicated by a hat superscript, e.g., $\hat{\mathbf{x}}$. They also can be referred to as *a posteriori* estimates in contrast to estimates that are additionally labeled with a minus superscript, which are *a priori* estimates, e.g., $\hat{\mathbf{x}}^-$. The reason for this distinction is given in Chapter 4.
- Matrices, regardless in which dimension, are denoted in bold-face capital letters, e.g., $\mathbf{A}, \mathbf{B}, \mathbf{C}$. Usually, their dimension is clear from the context, but sometimes it will be denoted by a subscript, e.g. $\mathbf{A}_{3\times 3}$.
- *Frames of reference* are denoted by sans-serif capital letters, e.g., W, C. They are imaginary alignments of a set of axes in 3D space also referred to as *coordinates frames*. These axes are denoted by sans-serif lower case symbols, e.g., $\mathsf{x}, \mathsf{y}, \mathsf{z}$.
  In this work we have to deal with four reference frames: the *world coordinate frame* W, the *camera coordinate frame* C, the *image coordinate frame*

I and the *pixel coordinate frame* P. Vectors that are related to a specific reference frame, e.g., to W, are denoted by appropriate superscripts like $\mathbf{x}^{\mathsf{W}}$.
We frequently will need to convert vectors, related to a frame of reference, to another, e.g., by rotating them. Therefore, we need a transformation matrix converting the components of a vector in frame C to components in frame W. This is denoted by

$$\mathbf{x}^{\mathsf{C}} = \mathbf{R}^{\mathsf{CW}}\mathbf{x}^{\mathsf{W}}.$$

The inverse transformation is then formulated easily as

$$\mathbf{x}^{\mathsf{W}} = \mathbf{R}^{\mathsf{WC}}\mathbf{x}^{\mathsf{C}}.$$

- Instead of the conventional 3×3 rotation matrices we will use quaternions as a representation of rotations. This will ease the linearization needed to derive the appropriate jacobian matrices in Appendix A. Quaternions are four-dimensional vectors denoted by $\mathbf{q} = (q_0, q_1, q_2, q_3)^\top$ with $\|\mathbf{q}\| = \mathbf{1}$.
If we have given a roation as an angle θ around some axis given as a unit vector $\mathbf{x}$, the according quaternion can be calculated by

$$\mathbf{q} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2})\mathbf{x} \end{bmatrix} \tag{3.1}$$

In Appendix B the handling of quaternions is explained in detail.

- We also have to deal with the geometry of two views. To distinguish between the left and the right camera system we label vectors referring to the left one with superscript l and vectors referring to the right one with superscript r, e.g.,$\mathbf{x}_l, \mathbf{x}_r$.

3.2 Camera Models and Calibration

To obtain measurements of feature points in the environment we will use vision sensors. To relate these measurements of 2D points in a picture to 3D Points in the world, we need to understand the process of projection and model it.

In the following we will introduce the *pinhole camera model* which describes how a 2D image of a 3D scene is obtained by a monocular camera. Secondly, the geometry of two views is explained to obtain a model for a stereo camera.

This section is chiefly based on the book *Computer Vision* by Klette et al. [29].

3.2.1 Pinhole Camera Model

In the pinhole model it is assumed that an image is derived by a central projection of points in space onto the image plane. As depicted in Figure 3.1, the intersection of a ray from the *camera projection centre* C to the 3D position of a point $\mathbf{x}^{\mathsf{C}} = (x^{\mathsf{C}}, y^{\mathsf{C}}, z^{\mathsf{C}})$ with the *image plane* is defined as the image $\mathbf{x}^{\mathsf{I}} = (x^{\mathsf{I}}, y^{\mathsf{I}})$ of this point. The camera coordinate frame is three-dimensional with the axes x, y and z. The coordinate frame of the image plane is two-dimensional with the

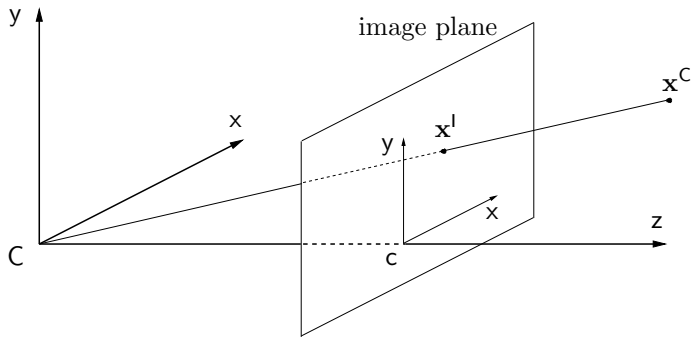

Figure 3.1: *The pinhole camera model.*

axes x and y. The z-axis is also referred to as the *principal axis* and perpendicularly pierces the image plane at the *principal point* $\mathbf{c} = (0, 0)$ which at the same time represents the origin of the image coordinate frame. The distance between the camera projection centre and the principal point is referred to as the *focal length* f. Therefore, the image coordinates $(x^{\mathsf{I}}, y^{\mathsf{I}})$ of the projected 3D camera point $(x^{\mathsf{C}}, y^{\mathsf{C}}, z^{\mathsf{C}})$ can be also represented in in camera coordinates: $(x'^{\mathsf{C}} = x^{\mathsf{I}}, y'^{\mathsf{C}} = y^{\mathsf{I}}, z'^{\mathsf{C}} = f)$. We can summarize this relation as:

$$x^{\mathsf{I}} = f\frac{x^{\mathsf{C}}}{z^{\mathsf{C}}} \tag{3.2}$$

$$y^{\mathsf{I}} = f\frac{y^{\mathsf{C}}}{z^{\mathsf{C}}}. \tag{3.3}$$

If we use homogeneous coordinates $\tilde{\mathbf{x}}^{\mathsf{I}} = (x^{\mathsf{I}}, y^{\mathsf{I}}, w^{\mathsf{I}})$ and $\tilde{\mathbf{x}}^{\mathsf{C}} = (x^{\mathsf{C}}, y^{\mathsf{C}}, z^{\mathsf{C}}, w^{\mathsf{C}})$ instead of Euclidian coordinates, we can express this fact in a matrix

$$\tilde{\mathbf{x}}^{\mathsf{I}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{x}}^{\mathsf{C}}. \tag{3.4}$$

This matrix is referred to as the *projection matrix* $\mathbf{F}^{\mathsf{IC}}$.

If we set $f = 1$ the image coordinates are *normalized*.

External Parameters

In the description above, it is assumed that a 3D point is already given in camera coordinates. This is quite simplified. More common is the case that 3D points are related to the world reference frame and have to be converted into camera coordinates before projecting them onto the image plane as already explained above.

Assume that the camera coordinate frame is somehow shifted and rotated with respect to the world coordinate frame like depicted in Figure 3.2. The translation can be described by a three-dimensional vector $\mathbf{t}^{\mathsf{W}} = (t_x, t_y, t_z)^\top$. The rotation is fully described by three angles α, β and γ around the x−, y− and z−axes which can be augmented into a rotation matrix $\mathbf{R}$. The whole transformation of world into camera coordinates can be represented by a 4×4

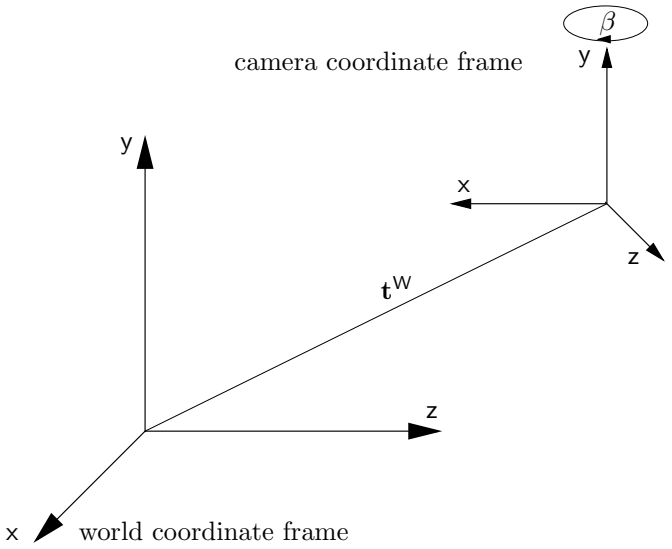

Figure 3.2: *The camera coordinate frame shifted about* $\mathbf{t}^{\mathsf{W}}$ *and rotated about* β *around the* $\mathsf{y}-$*axis with respect to the world coordinate frame.*

matrix $\mathbf{D}^{\mathsf{CW}}$ containing the translation vector as well as the rotation matrix. We have

$$\tilde{\mathbf{x}}^{\mathsf{C}} = \mathbf{D}^{\mathsf{CW}}\tilde{\mathbf{x}}^{\mathsf{W}} \tag{3.5}$$

$$\tag{3.6}$$

$$= \begin{bmatrix} \mathbf{R}^{\mathsf{W}}_{3\times 3} & \mathbf{t}^{\mathsf{W}}_{(3)} \\ \mathbf{0}_{(3)} & 1 \end{bmatrix} \text{ with } \mathbf{0} = (0\ 0\ 0). \tag{3.7}$$

Internal Parameters

Up to now, we just have consideres the image coordinate frame I. In addition to that, there is the pixel coordinate frame P, where the pixels are adressed in instead of image points. Its axes are named as the $\mathsf{u}-$ and $\mathsf{v}-$axis.

Its origin is not necessarily the same as the one of the image coordinate frame. Usually, it is the upper left corner of the image plane. The origin of the image coordinate frame is referred to as the principal point and is given as $\mathbf{c}^{\mathsf{P}} = (\mathbf{u}^{\mathsf{P}}_{\mathbf{0}}, \mathbf{v}^{\mathsf{P}}_{\mathbf{0}})$ in pixel coordinates. The $\mathsf{u}-$ and $\mathsf{v}-$ axis are maybe not perpendicular to each other. The angle between them is referred to as θ. Here, we assume that $\theta = 90°$ and therefore, $\cos\theta = 0$. The size of the pixels also has to be taken into consideration. The pixel dimensions are given as k_x and k_y.

We can express the relation between the image and pixel coordinate frame as a matrix converting image into pixel coordinates.

$$\tilde{\mathbf{x}}^{\mathsf{P}} = \mathbf{K}^{\mathsf{PI}}\tilde{\mathbf{x}}^{\mathsf{I}}$$

$$= \begin{bmatrix} \frac{f}{k_x} & 0 & u_0^{\mathsf{P}} \\ 0 & \frac{f}{k_y} & v_0^{\mathsf{P}} \\ 0 & 0 & 1 \end{bmatrix} \tilde{\mathbf{x}}^{\mathsf{I}}$$

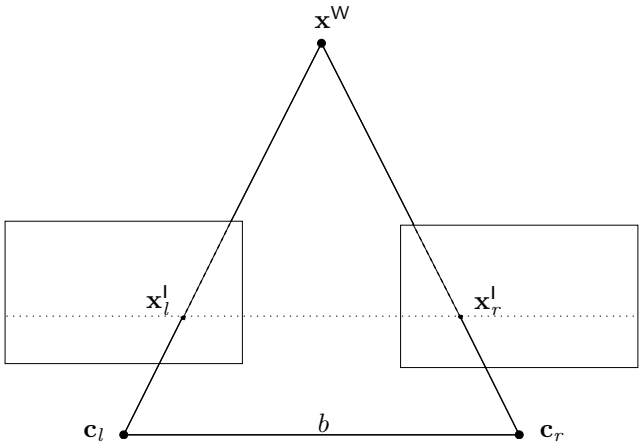

Figure 3.3: *Entities in standard stereo geometry.*

The parameters contained in the so called *calibration matrix* $\mathbf{K}^{\mathsf{PI}}$ are referred to as *internal parameters*.

3.2.2 Stereo Camera Model

In the last section the projection of a 3D point onto the image plane of a camera was explained. In the case of a stereo camera we have two image planes, two projection centres and therefore two images of one 3D point. In general, the image planes are shifted and rotated to each other. Here we assume what is known as *standard stereo geometry*. This constellation is characterized by an identical focal length f of the left and right camera , parallel principal axes and row-identical orientation of the image planes. The distance between the projection centers $\mathbf{c}_l$ and $\mathbf{c}_r$ is denoted as b, the so called *baseline*. A stereo image derived by a camera system in standard stereo geometry is referred to as *rectified*.

If we have a 3D point $\mathbf{x}^{\mathsf{C}} = (x^{\mathsf{C}}, y^{\mathsf{C}}, z^{\mathsf{C}})$ in camera coordinates the projections $\mathbf{x}_l^{\mathsf{I}} = (x_l^{\mathsf{I}}, y^{\mathsf{I}})$ and $\mathbf{x}_r^{\mathsf{I}} = (x_r^{\mathsf{I}}, y^{\mathsf{I}})$ will have the same $y^{\mathsf{I}}-$coordinate but different $x^{\mathsf{I}}-$coordinates. This situation is depicted in Figure 3.3. The difference between the left and right $x^{\mathsf{I}}-$coordinate is referred to as the *disparity* d.

We can determine the 3D coordinates of the considered point related to the left or right camera system if we have given the two projections $\mathbf{x}_l^{\mathsf{I}}$ and $\mathbf{x}_r^{\mathsf{I}}$ of a world point $\mathbf{x}^{\mathsf{W}}$, the baseline b and the focal length f. Here, we assume that the left camera system is equal to the world coordinate frame. Then we have

$$x^{\mathsf{W}} = x_l^{\mathsf{C}} = \frac{b * x_l^{\mathsf{I}}}{x_l^{\mathsf{I}} - x_r^{\mathsf{I}}} = \frac{b * x_l^{\mathsf{I}}}{d} \tag{3.8}$$

$$y^{\mathsf{W}} = y_l^{\mathsf{C}} = \frac{b * y^{\mathsf{I}}}{x_l^{\mathsf{I}} - x_r^{\mathsf{I}}} = \frac{b * y^{\mathsf{I}}}{d} \tag{3.9}$$

$$z^{\mathsf{W}} = z_l^{\mathsf{C}} = \frac{b * f}{x_l^{\mathsf{I}} - x_r^{\mathsf{I}}} = \frac{b * f}{d} \tag{3.10}$$

If we assume that we have already given the two projections of one point we ignored one of the main problems in stereo analysis: the search for correspond-

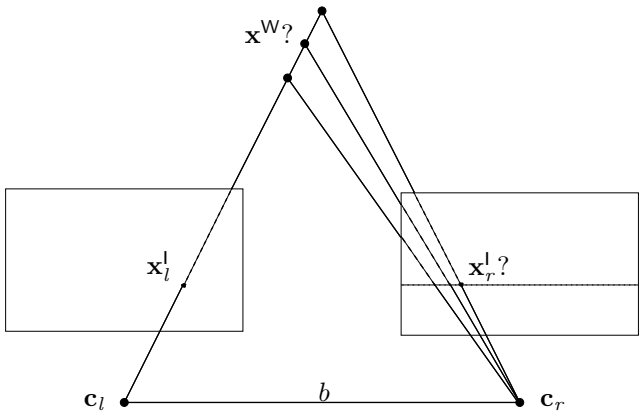

Figure 3.4: *The epipolar line constraint. The world point* $\mathbf{x}^{\mathsf{W}}$*, its projections* $\mathbf{x}^{\mathsf{I}}_l$ *and* $\mathbf{x}^{\mathsf{I}}_r$*, both projections centers* $\mathbf{c}_l$ *and* $\mathbf{c}_r$ *span the* epipolar plane*. The intersection of this plane with an image plane is referred to as an* epipolar line*. The projections of all world points situated on the ray from the left camera centre* $\mathbf{c}_{\mathsf{I}}$ *to some world coordinates will lie on the same epipolar line on the right image plane.*

ing image points concerning one 3D point. As already stated above, we assume that the camera system is in standard stereo geometry. Thus, corresponding projections of one point will always have the same $\mathsf{y}-$ coordinate. This coordinate corresponds to a horizontal line in each image and is refered to as *epipolar line*. If we have given the image coordinates of the left projection of a world point, we just need to search the according epipolar line in the right image to determine the corresponding right projection. We reduced the dimension of the search space from 2D to 1D. This relation is referred to as the *epipolar constraint* and is depicted in Figure 3.4.

Chapter 4

The Kalman Filter Approach

Imagine you are sitting in a car waiting at a crossroad to pass it. The visibility is poor due to parked cars at the roadside. But there are some gaps between them so that you have the possibility to observe these openings to decide whether you can cross the street without causing an accident or not. You have to guess the number, position and velocity of potential vehicles moving on the road from just a few information derived by watching these gaps over time.
Let us integrate the mentioned attributes of the street into the concept of a *state* of the street. The observations can also be seen as *measurements* and are noisy because of the poor visibility.
An estimation of the state of the street is just possible if you know how vehicles move on a road and how the measurements are related to this motion. Due to the noise in the measurements and to not directly observable aspects like acceleration there will not be absolute certainty in your estimation.

This task is one instance of the problem known as the *observer design problem*. In general, you have to estimate the unknown internal state of a dynamical system given its output in the presence of uncertainty. The output depends somehow on the system's state. To be able to infer this state from the output you need to know the according relation and the system's "behaviour". In such situations, we have to construct a model. In practise it is not possible to represent the system considered with absolute precision. Instead, the according model will stop at some level of detail. The gap between it and reality is filled with some probabilistic assumption referred to as *noise*. The noise model introduced in this chapter will be applied throughout this work.

An optimal solution for this sort of problems in the case of linear models can be derived by using the *Kalman Filter* which is explained in the first section of this chapter based on [12]. Most of the interesting instances of the observer design problem, e.g. the SLAM problem, do not fulfil the condition of linearity. To be able to apply the Kalman Filter approach to this non-linear sort of tasks, we have to linearise the models. The according algorithm is referred to as *Extended Kalman Filter*. We will introduce it in the second section.

4.1 The Discrete Kalman Filter

In this section we introduce the Kalman Filter chiefly based on its original formulation in [17] where the state is estimated at discrete points in time. The algorithm is slightly simplified by ignoring the so called *control input* which is not used in this specific application of purely vision based SLAM. Nevertheless, in a robotic application it might be useful to involve e.g. odometry data as control input. A complete description of the Kalman Filter can be found in [17] and [12].

In the following, we will firstly introduce the models for the system's state and the process model which describes the already mentioned system's "behaviour". Here, also the noise model is presented. After that, we introduce the model for the relation between the state and its output. The section closes with a description of the whole Kalman Filter algorithm.

4.1.1 Model for the Dynamical System to Be Estimated

The Kalman filter is based on the assumption that the dynamical system, which should be estimated, can be modelled as a normally distributed random process $\mathbf{X}(k)$ with mean $\hat{\mathbf{x}}_k$ and covariance matrix $\mathbf{P}_k$ where index k represents time. The mean $\hat{\mathbf{x}}_k$ is referred to as the estimate of the unknown real state $\mathbf{x}_k$ of the system at the point k in time. This state is modelled by an n dimensional vector:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix}$$

For the simplicity of the notation we did not use the subscript k, here. Throughout this work, we will continue omitting k when the components of a vector or matrix are presented even if they are different at each point in time.

Our main objective is to derive a preferably accurate estimate $\hat{\mathbf{x}}_k$ for the state of the observed system at time k.

The covariance matrix $\mathbf{P}_k$ describes the possible error between the state estimate $\hat{\mathbf{x}}_k$ and the unknown real state $\mathbf{x}_k$, in other words - the uncertainty in the state estimation after time step k. It can be modelled as an $n \times n$ matrix.

$$\mathbf{P} = \begin{bmatrix} x_1x_1 & \dots & x_1x_i & \dots & x_1x_n \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_ix_1 & \dots & x_ix_i & \dots & x_ix_n \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_nx_1 & \dots & x_nx_1 & \dots & x_nx_n \end{bmatrix}$$

where the main diagonal contains the variances of each variable in the state vector and the other entries contain the covariances of pairs of these variables. Covariance matrices are always symmetric due to the symmetric property of

covariances.[1]

If we want to derive an accurate estimate of the system's state, the corresponding uncertainty should obviously be small. The Kalman filter is optimal in the sense, that it minimises the *error covariance matrix* $\mathbf{P}_k$.

4.1.2 Process Model

Examined over time the dynamical system is subject to a transformation. Some aspects of this transformation are known and can be modelled. Others, e.g., acceleration as in the example above (also influencing the state of the system) are unknown, not measurable or too complex to be modelled. Then, this transformation has to be approximated by a *process model* $\mathbf{A}$ involving the known factors. The "classic" Kalman filter expects that the model is linear. Under this condition the normal distribution of the state model is maintained after it has undergone the linear transformation $\mathbf{A}$. The new mean $\hat{\mathbf{x}}_k$ and covariance matrix $\mathbf{P}_k$ for the next point in time are derived by

$$\hat{\mathbf{x}}_k = \mathbf{A}\hat{\mathbf{x}}_{k-1} \tag{4.1}$$

$$\mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top. \tag{4.2}$$

Due to the approximative character of $\mathbf{A}$, the state estimate $\hat{\mathbf{x}}_k$ is also just an approximation of the real state $\mathbf{x}_k$. The difference is represented by a random variable $\mathbf{w}$:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}. \tag{4.3}$$

The individual values for $\mathbf{w}$ are not known for each point k in time but need to be involved to improve the estimation. We assume these values to be realisations of a normally distributed white noise vector with zero mean. In the following, this vector $\mathbf{w}$ is referred to as *process noise*. It is denoted by

$$p(\mathbf{w}) \sim N(0, \mathbf{Q}) \tag{4.4}$$

where zero is the mean and $\mathbf{Q}$ the *process noise covariance*. The individual values of $\mathbf{w}$ at each point in time can now be assumed to be equal to the mean, to zero. Thus, we stick to Equation (4.1) to estimate $\mathbf{x}_k$ as $\hat{\mathbf{x}}_k$.

The process noise does not influence the current state estimate, but the uncertainty about it. Intuitively we can say, the higher the discrepancy is between the real process and the according model, the higher is the uncertainty about the quality of the state estimate.
This can be expressed by extending the computation of the error covariance $\mathbf{P}_k$ in Equation (4.2) with the process noise covariance matrix $\mathbf{Q}$.

$$\mathbf{P_k} = \mathbf{A}\mathbf{P_{k-1}}\mathbf{A}^\top + \mathbf{Q} \tag{4.5}$$

The choice of the values for the process noise covariance matrix reflects the quality we expect from the process model. If we set them to small values, we are quite sure that our assumptions about the considered system are mostly right. The uncertainty regarding to our estimates will be low. But then, we will be unable or hardly able to cope with large variations between the model and

[1] The covariance value x_1x_n is the same as x_nx_1. In practise this means, that x_1 is correlated to x_n like x_n to x_1

the system. Setting the variances to large values instead means to accept that there might be large differences between the state estimate and the real state of the system. We will be able to cope with large variations but the uncertainty about the state estimate will increase stronger than with a small process noise. A lot of good measurements are needed to constrain the estimate.

4.1.3 Output of the System

As already mentioned earlier, the output of the system is related to the state of the system. If we know this relation and the estimated state after the current time step, we are able to predict the according measurement of the system's output. In this section, we will introduce the model for the measurement of the output. In the next section the relation between state and output is examined.

As well as the state of the considered dynamical system, also its output is modelled as a normally distributed random process $\mathbf{Z}(k)$ with mean $\hat{\mathbf{z}}_k$ and covariance matrix $\mathbf{S}_k$ where index k indicates time. The mean $\hat{\mathbf{z}}_k$ represents the estimated and predicted measurement of the output depending on the state estimate $\hat{\mathbf{x}}_k$ at the point k in time. The real measurement $\mathbf{z}_k$ of the output is obtained by explicitly measuring the system's output. $\mathbf{z}_k$ is modelled as an m dimensional vector

$$\mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_i \\ \vdots \\ z_m \end{bmatrix}$$

The so called innovation covariance matrix $\mathbf{S}_k$ describes the possible error between the estimate $\hat{\mathbf{z}}_k$ and the real measurement $\mathbf{z}_k$, in other words - the uncertainty in the measurement estimation after time step k. It can be modelled as an $m \times m$ matrix

$$\mathbf{S} = \begin{bmatrix} z_1 z_1 & \dots & z_1 z_i & \dots & z_1 z_m \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_i z_1 & \dots & z_i z_i & \dots & z_i z_m \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_m z_1 & \dots & z_m z_1 & \dots & z_m z_m \end{bmatrix}$$

where the main diagonal contains the variances of each variable in the measurement vector and the other entries contain the covariances of pairs of these variables.

Note, that in contrast to the system's real state, the real measurement can be obtained and we are therefore able to compare predicted and real measurement. The precisely known difference between estimation and reality constitutes the basis to correct the state estimate used to predict the measurement. This will be explained in detail in Section 4.1.5.

4.1.4 Measurement Model

In the previous sections we mentioned, that the system's output is somehow related to the system's state. In this sections the relation is modelled.

We have the same situation as for the process model. The connection between the output and the state can just be modelled up to a certain degree. Known factors are summarised in the *measurement model* $\mathbf{H}$. After we have obtained a new state estimate for the current point in time, we can apply $\mathbf{H}$ to predict the according measurement $\hat{\mathbf{z}}_k$ and covariance matrix $\mathbf{S}_k$. If this measurement model is linear, the normal distribution of the state model is maintained after applying this linear transformation.

$$\hat{\mathbf{z}}_k = \mathbf{H}\hat{\mathbf{x}}_k \tag{4.6}$$
$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_k\mathbf{H}^\top. \tag{4.7}$$

Because measurements of the system's output are mostly noisy due to inaccurate sensors, the difference between the estimate $\hat{\mathbf{z}}_\mathbf{k}$ and the real measurement $\mathbf{z}_\mathbf{k}$ is not just caused by the dependency on the state estimate but also by a random variable $\mathbf{v}$:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k. \tag{4.8}$$

As already mentioned for the process noise, the individual values of $\mathbf{v}$ are not known for each point k in time. We apply the same noise model and approximate these unknown values as realisations of a normally distributed white noise vector with zero mean. In the following, $\mathbf{v}$ is referred to as *measurement noise*. It is denoted by

$$p(\mathbf{v}) \sim N(0, \mathbf{R}) \tag{4.9}$$

As $\mathbf{v}$ is now assumed to be equal to the mean of its distribution at each point in time, it does not influence the measurement estimate but the uncertainty about it. This is modelled by extending the computation of the measurement innovation covariance matrix $\mathbf{S}_k$ in Equation (4.7) with the measurement noise covariance matrix $\mathbf{R}$.

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_k\mathbf{H}^\top + \mathbf{R} \tag{4.10}$$

Again, the values chosen for the measurement noise covariance matrix indicate how sure we are about the assumptions we made in our measurement model.

More information about the influence of the measurement noise are given below in connection with the *Kalman Gain*.

4.1.5 Predict and Correct Steps

In the last sections we introduced the model for the process the system is subject to and the model for the relation between the system's state and its output. These models are used in the Kalman Filter algorithm to determine an optimal estimate of the unknown state of the system.

As already mentioned in Section 4.1.3, we use the known difference between the predicted measurement $\hat{\mathbf{z}}_k$ and real measurement $\mathbf{z}_k$ as basis to correct the state estimate derived by the application of the process model $\mathbf{A}$. The filter can be divided into two parts. In the *predict step* the process model and the current state and error covariance matrix estimates are used to derive an *a priori* state estimate for the next time step. Next, in the *correct step*, a (noisy) measurement is obtained to enhance the *a priori* state estimate and derive an improved *a posteriori* estimate.

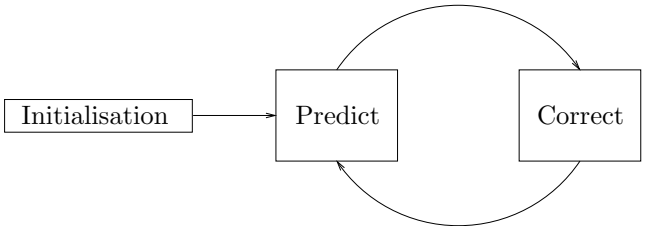

Figure 4.1: *The Predict-Correct Cycle of the Kalman Filter Algorithm.*

Before this predict-correct cycle as depicted in Figure 4.1 can be started, the state and its error covariance matrix have to be initialised. In the following we will assume that this is already the case.

Predict Step

We are situated at the point k in time and the state and error covariance matrix estimates at time $k-1$ are given. By using Equations (4.1) and (4.5) we predict the state and error covariance matrix for k:

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \mathbf{A}\hat{\mathbf{x}}_{k-1} \\ \mathbf{P}_k^- &= \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}. \end{aligned}$$

The minus superscript labels the predicted state and error covariance matrix as *a priori* in contrast to *a posteriori* estimates.

Correct Step

Assume that we have already obtained an actual measurement $\mathbf{z}_k$ of the system's output. With the help of this, we first want to calculate the *a posteriori* state estimate $\hat{\mathbf{x}}_k$. This is a linear combination of the *a priori* estimate $\hat{\mathbf{x}}_k^-$ and a weighted difference between $\mathbf{z}_k$ and the predicted measurement $\hat{\mathbf{z}}_k$. According to Equation (4.6), $\hat{\mathbf{z}}_k$ is calculated by $\mathbf{H}\hat{\mathbf{x}}_k^-$. Summarised, we have:

$$\begin{aligned} \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k) \\ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-). \end{aligned}$$

The difference $\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-$ is called *measurement innovation* or *residual*. If the value is zero, the prediction and the actual measurement are in complete agreement and the *a priori* state estimate won't be corrected. If it is unequal to zero, $\hat{\mathbf{x}}_k$ will be unequal to $\hat{\mathbf{x}}_k^-$.

The weight $\mathbf{K}_k$, the so called *Kalman Gain*, is represented by a $n \times m$ matrix and minimises the *a posteriori* error covariance estimate $\mathbf{P}_k^-$. It can be calculated by

$$\mathbf{K}_k = \frac{\mathbf{P}_k^-\mathbf{H}^\top}{(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{R})} \tag{4.11}$$

Note, that the denominator equals Equation (4.10), representing the uncertainty in the predicted measurement. If we look closely at Equation (4.11), we can

see that, if the measurement error covariance error $\mathbf{R}$ approaches zero, the measurement innovation is weighted more heavily.

$$\lim_{\mathbf{R}\to 0} \mathbf{K}_k = \frac{1}{\mathbf{H}}$$

In other words, the smaller the measurement error, the more reliable is the actual measurement $\mathbf{z}_k$.
On the other hand, if the predicted error covariance matrix $\mathbf{P}_k^-$ approaches to be zero the residual is weighted less.

$$\lim_{\mathbf{P}_k^-\to 0} \mathbf{K}_k = 0$$

This means, the smaller the uncertainty in the *a priori* state estimate $\hat{\mathbf{x}}_k^-$, the more reliable is the predicted measurement $\hat{\mathbf{z}}_k$.

Secondly, we have to correct the *a priori* error covariance matrix estimate to derive the *a posteriori* estimate.

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$$

For details of the derivation of the filter algorithm see [26].

In Figure 4.2 the whole algorithm is given again step by step.

4.1.6 A Simple Example

To clarify the effectiveness of the Kalman Filter we will examine a simple example. To stick to the central theme of this work right from the beginning, this example will be an instance of the SLAM problem. The section will be structured as follows: Firstly, we will give a short description of the problem. After that, the process and measurement model are formulated. The section closes with some experiments on simulated data.

Problem Description

In Chapter 5, we will analyse how to apply the Kalman Filter approach to the problem of SLAM with using a vision sensor mounted on a robot. This firstly means to track the position and orientation of the camera within the 3D environment (localisation) and secondly to estimate the position of some landmarks situated in the world (mapping).

In the following we will simplify this task to SLAM in one dimension. The camera is represented by a point moving randomly in 1D. There also is a static landmark with a position known up to a certain degree. The process model of this example should describe the motion of the camera. We will assume that it moves smoothly so that fast changes in its velocity are unlikely. We are able to measure the distance between the landmark and the moving point at discrete points in time. The measurement model should relate this distance to the state of the considered system.

The situation is depicted in Figure 4.3.

1. Predict Step

   (a) Predict the state

   $$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1}$$

   (b) Predict the error covariance matrix

   $$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}$$

2. Correct Step

   (a) Calculate the Kalman Gain

   $$\mathbf{K}_k = \frac{\mathbf{P}_k^-\mathbf{H}^\top}{(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{R})}$$

   (b) Correct the *a priori* state estimate

   $$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-)$$

   (c) Correct the *a priori* error covariance matrix estimate

   $$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$$

Figure 4.2: *Equations of one Kalman Filter Cycle. We assume that the state, its covariance and the noise values are already initialised.*

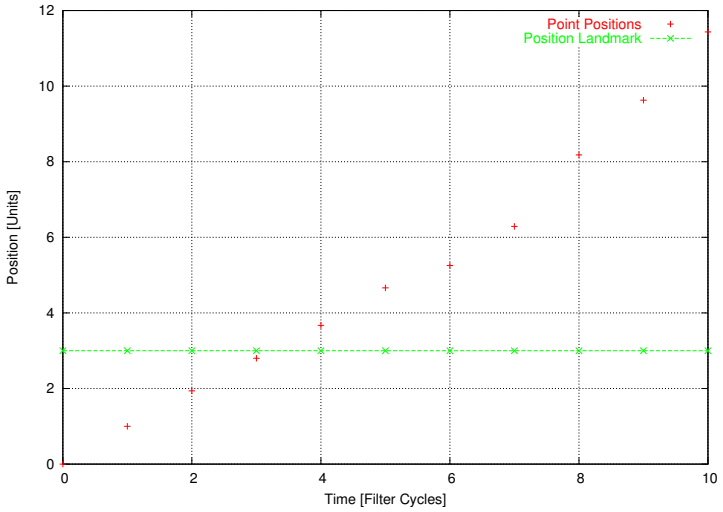

Figure 4.3: *An Example for a Point Moving Randomly in 1D. A static landmark is situated at $x = 3$. The distance between the current point position and this landmark is measurable at each time step.*

Process and Measurement Model

At first we have to model the state $\mathbf{x}$ which has to be estimated. Three important entities have to be taken into account. Firstly, there is the position of the point in a point in time. It is fully described by an one-dimensional coordinate in x-direction. Secondly, we choose a constant velocity to describe the motion of the point. [2] This does not mean that we assume the point is moving constantly over all time but that this value is the average velocity between two points in time and changes occur with a Gaussian profile. These changes are modelled beneath as process noise. At last, the position of the landmark has to be augmented into the state.

$$\mathbf{x} = \begin{bmatrix} x_p \\ v_p \\ x_f \end{bmatrix} = \begin{bmatrix} \text{Position of the point} \\ \text{Velocity of the point} \\ \text{Position of the landmark} \end{bmatrix}$$

The error covariance matrix is then a 3×3 matrix of the following form

$$\mathbf{P} = \begin{bmatrix} x_p x_p & x_p v_p & x_p x_f \\ v_p x_p & v_p v_p & v_p x_f \\ x_f x_p & x_f v_p & x_f x_f \end{bmatrix}.$$

The task of the process model $\mathbf{A}$ is to approximate the transformation of the considered system over time. Here, this is the motion of the point between time k and $k-1$. This constant time period is denoted as $\triangle k$. $\mathbf{A}$ is used to predict the state of the system for the current point k in time from the old state estimate at time $k-1$ by calculating $\hat{\mathbf{x}}(k) = \mathbf{A}\hat{\mathbf{x}}(k-1)$.

$$\begin{array}{rcl} \hat{x}_p(k) & = & \text{old point position + old velocity per } \triangle k \\ & = & \hat{x}_p(k-1) + \hat{v}_p(k-1)\triangle k \\ \hat{v_p}(k) & = & \text{constant velocity due to assumed smooth motion} \\ & = & \hat{v}_p(k-1) \\ \hat{x}_f(k) & = & \text{static landmark} \\ & = & \hat{x}_f(k-1) \end{array} \tag{4.12}$$

As already mentioned, the constant velocity value just describes the average velocity in the time period $\triangle k$. Therefore, it is just an approximation. Variations are caused by random unmeasurable accelerations a.[3] We involve it in the process noise vector $\mathbf{w}$. If we would know the individual values of $\mathbf{w}$ at each k, we could derive the real state:

$$\mathbf{x}(k) = \mathbf{A}\mathbf{x}(k-1) + \mathbf{w}(k-1)$$

Because the process noise is an additive constant, $\mathbf{w}$ is modelled as a three-dimensional vector $\mathbf{w} = (w_0, w_1, w_2)^\top$. Noise is just added to the velocity component of the state. Thus, the first and third component, w_0 and w_2, referring to the position of the moving point and to the position of the landmark, are set to zero. Just the second value carries a different random value after each time step: $\mathbf{w} = (0, a\triangle k, 0)^\top$. Adding the noise term to the process model, we

[2] A velocity v_p describes the distance $\triangle x$ covered in a certain time intervall $\triangle k$.

[3] An acceleration a is a change in velocity $\triangle v_p$ in a certain time intervall $\triangle k$. Thus, $w_1 = a\triangle k = \triangle v_p$, the change in velocity.

have:

$$\begin{aligned} x_p(k) &= x_p(k-1) + (v_p(k-1) + a(k-1)\triangle k)\triangle k \\ v_p(k) &= v_p(k-1) + a(k-1)\triangle k \\ x_f(k) &= x_f(k-1) \end{aligned}$$

We do not know the individual values for a at each point in time. Therefore, we model the process noise as a realization of a normally distributed white noise random vector with zero mean and a covariance matrix $\mathbf{Q}$.

$$p(\mathbf{w}) \sim N(0, \mathbf{Q})$$

Now, we can assume $\mathbf{w}$ to be equal to the mean of its distribution, which is zero. We derive the process model already formulated in Equation (4.12). Expressed in a linear transformation with $\triangle k$ assumed to be 1, this is

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$\mathbf{Q}$ is of the following form:

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma_p^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The constant value of σ_p as the standard deviation of the noise in the velocity value indicates the amount of smoothness in the motion we expect. If we choose it to be small, we expect the point to move with a nearly constant velocity. Then, we will not be able to cope with sudden accelerations. If we choose large values instead, we will be able to track the point well, if it acts in another way than expected by the process model. On the other hand, the uncertainty about a state estimate is higher than with small values for σ_p.

The measurement model approximates the relation between the actual measurement z_k and the current state $\mathbf{x}_k$. In our example the measurement consists of just one value representing the distance d_k between the moving point and the static landmark at the current point k in time. Expressed in a linear equation, we have

$$\hat{z}(k) = d_k = x_p(k) - x_f(k) \tag{4.13}$$

The sensor used to measure the distance is assumed to provide just noisy measurements. If we would know the value for this measurement noise exactly, we could determine the real measurement and not just an estimate. If we denote the measurement noise by the random variable v, the real measurement can be computed by:

$$z(k) = d_k = x_p(k) - x_f(k) + v(k).$$

But we do not know the individual values of the random variable v. Therefore, we apply our noise model such that the values of v are a realization of a normally distributed white noise with zero mean and the variance σ_m^2

$$p(v) \sim N(0, \sigma_m^2).$$

The measurement noise has the same dimension as the measurement and its distribution is therefore modelled by specifying a variance instead of a covariance matrix.

We can now assume the value v to be equal to the mean of its distribution, to zero. Then, we derive the measurement model already formulated in Equation (4.13). Note, that the difference between the estimate of the measurement $\hat{z}_k$ and the real measurement is not just caused by the unknown noise, but also by the fact that in reality we just have an estimate of the state to predict the measurement. The final measurement model for this problem is:

$$\hat{z}(k) = d_k = \hat{x}_p(k) - \hat{x}_f(k).$$

Expressed in a linear transformation we have

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}.$$

The constant value σ_m as the standard deviation of the measurement noise distribution indicates how sure we are about the correctness of the real measurements. Large value show that we do not trust them that much and we will weight the measurement innovation less. Small values indicate that the measured values are accurate. The residual will be weighted more heavily.

Experiments on Simulated Data

In the previous section, we derived the basis for the application of the Kalman Filter on our problem: the appropriate process and measurement model. In this section, we will test these models on simulated data. The simulation was initialized with the state:

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix}$$

The subsequent real positions of the point moving in 1D were generated by applying the process exactly described in the according model and adding some random values. The standard deviation of the random values is set to 0.2. The real measurements were also generated as described in the measurement model. Measurment noise is simulated by adding random values with a standard deviation of 0.2.

To start the predict-correct cycle of the Kalman Filter, we have to initialize the state and its error covariance matrix as well as the process and measurement noise values. Let us set the state to the real initial values. We assume an uncertainty about the initial position of the moving point as well as about the position of the landmark and velocity at time 0. Let the error covariance be

$$\mathbf{P}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The real noise in the measurements can usually be determined prior to the application of the filter. To determine the process noise covariance is more complicated, because we generally do not have the ability to measure the process, we want to estimate, directly. Anyway, we set the standard deviation of the

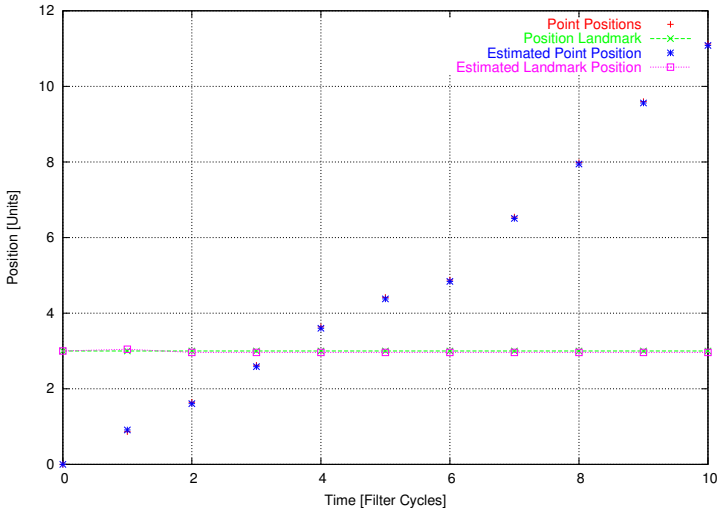

Figure 4.4: *The Simulation of the Problem of Estimating a Moving Point's Position by Orientating at a Single Landmark. The deviation between the estimation and real position of the point is very small as well as between the estimated and real position of the landmark.*

noise in the velocity σ_v and in the measurement σ_m to the real value used in the simulation: 0.2

We will run the filter on ten simulated measurements. The results are depicted in Figure 4.4. In Figure 4.5, the behaviour of the error covariance **P** during the ten filter cycles is visualized.

4.2 The Extended Kalman Filter

As we saw in Section 4.1.6, the Kalman Filter algorithm works quite well for the estimation of a linear system with linear related measurements depending on the quality of the appropriate models for the process and measurement of the output. Moreover, the Kalman Filter is optimal in the sense that it minimizes the error covariance representing the uncertainty in the estimate of the state.

To come back to the main theme of this work, estimating the position of a moving robot and of static landmarks using a camera sensor, we need to be able to cope with nonlinear motion and a nonlinear relationship between measurements and the system's state. The nonlinear motion is caused by possible rotational movements, the robot is able to do. Measurements of landmarks in the surrounding of the robot are projections of them onto the image plane of the camera sensor. The process of projection is nonlinear.

In Section 4.1.2, it is stated that a Gaussian distribution is maintained by a linear transformation. This is not the case if we use a nonlinear transformation instead. Thus, we cannot apply the Kalman Filter equations in its original formulation to estimate a nonlinear system. A solution for this problem is to linearize the transformation via Taylor Expansion. A Kalman Filter that uses Taylor Expansion to linearize the process and measurement models is called *Extended Kalman Filter*, in the following abbreviated as *EKF*.

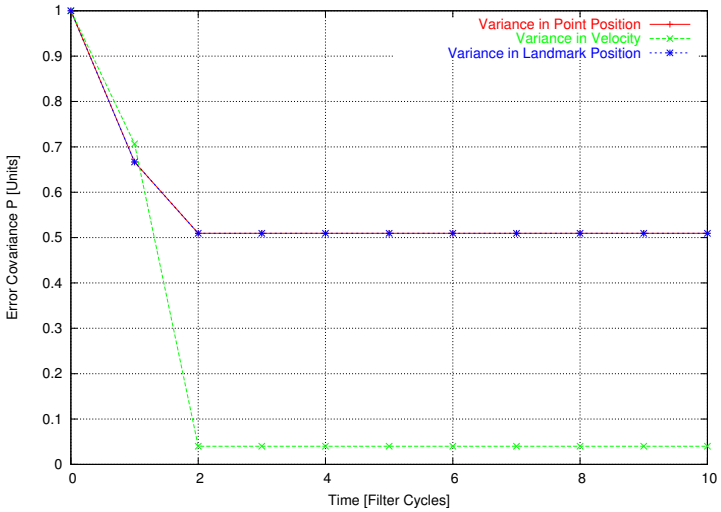

Figure 4.5: *The Error Covariance matrix* $\mathbf{P}$. *After two iterations, the initial value of* 1 *for the variances has settled at approximatly* 0.5 *for the estimation of the point's position and of the landmark's position and at approximatly* 0.04 *for the estimation of the velocity.*

Like in Section 4.1.1 we assume that the considered system can be modelled as a normally distributed random process $\mathbf{X}(k)$ with mean $\hat{\mathbf{x}}_k$ as the estimation of the real system's state $\mathbf{x}_k$ and covariance matrix $\mathbf{P}_k$. Its output can be modelled as well as a normally distributed random process $\mathbf{Z}(k)$ with mean $\hat{\mathbf{z}}_k$ as the prediction of the real measurement $\mathbf{z}_k$ and covariance matrix $\mathbf{S}_k$. In the following sections the EKF is derived for nonlinear process and measurement models.

Right from the beginning, we will stick to the "super minus" notation labeling *a priori* estimates.

4.2.1 Process Model

Let us assume that our system to be estimated, represented by a state vector $\mathbf{x}_k$ at time k, is now governed by the nonlinear funtion

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}) \tag{4.14}$$

relating the previous state $\mathbf{x}_{k-1}$ at the point $k-1$ in time to the next state $\mathbf{x}_k$ at the current point k in time. The random value $\mathbf{w}_{k-1}$ represents the process noise as in Equation (4.4).

$$p(\mathbf{w}) \sim N(0, \mathbf{Q})$$

We assume $\mathbf{w}$ to be equal to the mean of its distribution, which is zero. The result of the function f will then be an approximation $\hat{\mathbf{x}}_k^-$ of the real state $\mathbf{x}_k$.

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, 0) \tag{4.15}$$

Let the difference between the real state and its estimate, namely the error in the prediction, be a random variable $\mathbf{e}$:

$$\mathbf{e}_{x_k} = \mathbf{x}_k - \hat{\mathbf{x}}_k^- .$$

To be able to estimate the result of the process represented by the non-linear Equation (4.14) via the Kalman Filter algorithm, we linearize it about the current state estimate given in Equation (4.15) by setting up a first order Taylor polynomial ([16], p.411):

$$\mathbf{x}_k \approx \hat{\mathbf{x}}_k^- + \mathbf{A}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{W}\mathbf{w}_{k-1} = \hat{\mathbf{x}}_k \tag{4.16}$$

The matrix $\mathbf{A}$ is the Jacobian matrix containing the partial derivatives of (4.15) with respect to $\mathbf{x}$, whereas the Jacobian matrix $\mathbf{W}$ is filled with the partial derivatives of f with respect to $\mathbf{w}$. Note, that we ommitted time subscript k for the Jacobians to simplify the notation. Nevertheless, they may be different at each point in time. In the following, we will stick to omitting k for the Jacobian matrices.

The *a priori* estimate $\hat{\mathbf{x}}_k^-$ in Equation (4.16) can be calculated by $f(\hat{\mathbf{x}}_{k-1}, 0)$. The remainder term approximates $\mathbf{e}_{x_k}$ as $\hat{\mathbf{e}}_{x_k}$.

$$\mathbf{e}_{x_k} \approx \mathbf{A}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{W}\mathbf{w}_{k-1} = \hat{\mathbf{e}}_{x_k} \tag{4.17}$$

With this defintion of $\hat{\mathbf{e}}_{x_k}$, we can rewrite Equation (4.16) to

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \hat{\mathbf{e}}_{x_k} \tag{4.18}$$

According to Equation (4.18), we need to estimate the random value $\mathbf{e}_{x_k}$ as $\hat{\mathbf{e}}_{x_k}$ at each point in time to achieve our actual goal: estimating $\mathbf{x}_k$ as $\hat{\mathbf{x}}_k$.

Note, that (4.17) is a linear equation. Thus, we can apply a second hypothetical "classic" Kalman Filter to estimate $\mathbf{e}_{x_k}$. We will model this dynamical *linear error system* as a normally distributed random process with mean $\hat{\mathbf{e}}_{x_k}$ and covariance matrix $\mathbf{P}_k$ representing the uncertainty about the estimated $\mathbf{e}_{x_k}$. Since $\mathbf{e}_{x_k}$ denotes the error in the state estimate, it is clear that it should always be approximatly zero. Therefore, the mean $\hat{\mathbf{e}}_{x_k}$ of the distribution is chosen to be zero.

Let's consider Equation (4.17) again. The second term $\mathbf{W}\mathbf{w}_{k-1}$ denotes the noise in the estimation of $\mathbf{e}_{x_k}$. It is the product of the process noise $\mathbf{w}$ and the Jacobian matrix $\mathbf{W}$ containing the partial derivatives of Equation (4.15) with respect to $\mathbf{w}$. Remember, that the process noise is assumed to be always equal to zero. Thus, the term $\mathbf{W}\mathbf{w}_{k-1}$ is also assumed to be equal to zero. If $\mathbf{w}$ is transformed by applying $\mathbf{W}$, the corresponding covariance matrix $\mathbf{Q}$ of the process noise is transformed by $\mathbf{W}\mathbf{Q}\mathbf{W}^\top$. The noise in the estimation of $\mathbf{e}_{x_k}$ is then modelled as

$$p(\mathbf{W}\mathbf{w}_{k-1}) \sim N(0, \mathbf{W}\mathbf{Q}\mathbf{W}^\top).$$

To involve this noise in the prediction of the error $\mathbf{e}_{x_k}$ between real and estimated state, the according error covariance $\mathbf{W}\mathbf{Q}\mathbf{W}^\top$ is added to the prediction $\mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top$ of its error covariance $\mathbf{P}$. To summarize the last statements, we have:

$$\hat{\mathbf{e}}_{x_k}^- = \mathbf{A}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) = 0 \tag{4.19}$$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{W}\mathbf{Q}\mathbf{W}^\top. \tag{4.20}$$

Equations (4.19) and (4.20) represent the process model for the linear error system.

If we substitute Equation (4.19) for $\hat{\mathbf{e}}_{x_k}$ in Equation (4.18), the process model for the nonlinear system to predict a state estimate $\hat{\mathbf{x}}_k^-$ is then

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, 0) \tag{4.21}$$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{W}\mathbf{Q}\mathbf{W}^\top. \tag{4.22}$$

The process noise covariance matrix $\mathbf{W}\mathbf{Q}\mathbf{W}^\top$ acts in the nonlinear process model as the covariance matrix $\mathbf{Q}$ in the linear process model: It represents the amount of trust in the process model. High values indicate that high variations between the state estimate and the real state are expected. Low values show a lot of confidence in the process model.

4.2.2 Measurement Model

Let us assume that the relation between the system and its output is described by the nonlinear function

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_k) \tag{4.23}$$

where $\mathbf{v}_k$ represents the measurement noise as in (4.9).

$$p(\mathbf{v}) \sim N(0, \mathbf{R})$$

As usual, we assume $\mathbf{v}_k$ to be zero which is the mean of its distribution.

$$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_k^-, 0). \tag{4.24}$$

The result $\hat{\mathbf{z}}_k$ is just an approximation of the real measurement. Let the difference between the actual and the predicted measurement be the random value

$$\mathbf{e}_{z_k} = \mathbf{z}_k - \hat{\mathbf{z}}_k.$$

In contrast to the error $\mathbf{e}_{x_k}$ between the real state and its estimate, $\mathbf{e}_{z_k}$ is accessible.

To estimate the measurement of the system's output we linearize Equation (4.23) about the current state estimate given in Equation (4.24) by setting up a first order Taylor polynomial:

$$\mathbf{z}_k \approx \hat{\mathbf{z}}_k + \mathbf{H}(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) + \mathbf{V}\mathbf{v}_k \tag{4.25}$$

The matrix $\mathbf{H}$ is the Jacobian matrix containing the partial derivatives of Equation (4.24) with respect to $\mathbf{x}$ in contrast to the Jacobian matrix $\mathbf{V}$ which contains the derivatives of the same equation with respect to the measurement noise $\mathbf{v}$.

The predicted measurement $\hat{\mathbf{z}}_k$ in Equation (4.25) can be calculated by Equation (4.24). The error $\mathbf{e}_{z_k}$ is approximated as $\hat{\mathbf{e}}_{z_k}$ by the remainder term

$$\mathbf{e}_{z_k} \approx \mathbf{H}(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) + \mathbf{V}\mathbf{v}_k = \hat{\mathbf{e}}_{z_k}. \tag{4.26}$$

With this definition of $\hat{\mathbf{e}}_{z_k}$ we can rewrite Equation (4.25).

$$\mathbf{z}_k \approx \hat{\mathbf{z}}_k + \hat{\mathbf{e}}_{z_k} \tag{4.27}$$

Note, that Equation (4.26) is a linear equation. Therefore, we also model the error in the estimation of the output as a normally distributed random process

with mean $\hat{\mathbf{e}}_{z_k}$ and innovation covariance matrix $\mathbf{S}_k$, which approximates the error between the predicted and the actual measurement. From the notion that $\mathbf{e}_{z_k}$ specifies the estimated error in the estimation of the state $\mathbf{x}_k$ of the system, it is clear that it should preferably be approximatly equal to zero. Thus, the mean $\hat{\mathbf{e}}_{z_k}$ of its distribution is assumed to be always equal to zero.

If we re-consider Equation (4.26), we can state that $\mathbf{V}\mathbf{v}_k$ is the noise term in the prediction of $\mathbf{e}_{z_k}$. Remember that the measurement noise $\mathbf{v}$ is assumed to be zero at every point in time. Thus, the product of $\mathbf{v}$ and the Jacobian matrix $\mathbf{V}$ containing the partial derivatives of Equation (4.24) with respect to the noise is zero. If $\mathbf{v}$ is transformed by applying $\mathbf{V}$, the corresponding covariance matrix $\mathbf{R}$ is transformed by $\mathbf{V}\mathbf{R}\mathbf{V}^\top$. Then, the noise involved in the estimation of the error $\mathbf{e}_{z_k}$ is modelled as follows:

$$p(\mathbf{V}\mathbf{v}_k) \sim N(0, \mathbf{V}\mathbf{R}\mathbf{V}^\top)$$

The covariance matrix of the noise $\mathbf{V}\mathbf{v}_k$ is added to the prediction of the innovation covariance matrix by $\mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top$. Summarized, we have:

$$\hat{\mathbf{e}}_{z_k}^- = \mathbf{H}(\mathbf{x}_k - \hat{\mathbf{x}}_k) = 0 \tag{4.28}$$

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{V}\mathbf{R}\mathbf{V}^\top. \tag{4.29}$$

Equations (4.28) and (4.29) represent the measurement model for the linear error system and are used to correct the *a priori* error estimate $\hat{\mathbf{e}}_{x_k}^-$ between the state and its approximation.

If we substitute Equation (4.28) for $\hat{\mathbf{e}}_{z_k}$ in Equation (4.27), the measurement model for the nonlinear system is:

$$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_k^-, 0) \tag{4.30}$$

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{V}\mathbf{R}\mathbf{V}^\top. \tag{4.31}$$

4.2.3 Predict and Correct Steps

Using the Kalman Filter for the estimation of the state of a linear system, means that we exactly know how uncertain we are about this estimate. Whereas, using the EKF for the estimation of the state of a nonlinear system means to additionally estimate the uncertainty in this state estimate. This can be done by a second hypthetical Kalman Filter, presented in the previous chapters, which estimates the error between the real state and its estimate.

Let's assume, that we already used the process model for the nonlinear system given in Equations (4.21) and (4.22) to derive an *a priori* estimate $\hat{\mathbf{x}}_k^-$ for the state and $\mathbf{P}_k^-$ for its error covariance. Then, we can predict the measurement by using Equation (4.30). After we have obtained the real measurement $\mathbf{z}_k$, we can calculate the error $\mathbf{e}_{z_k}$ between $\mathbf{z}_k$ and the predicted measurement $\hat{\mathbf{z}}_k$.

According to Equation (4.19), the predicted error estimate $\hat{\mathbf{e}}_{x_k}^-$ between the real state and its estimate is assumed to be zero in every time step.

The Kalman Filter equation to correct the *a priori* error estimate $\hat{\mathbf{e}}_{x_k}^-$ and derive an *a posteriori* $\hat{\mathbf{e}}_{x_k}$ is then

$$\begin{aligned} \hat{\mathbf{e}}_{x_k} &= \hat{\mathbf{e}}_{x_k}^- + \mathbf{K}_k\mathbf{e}_{z_k} \\ &= \mathbf{K}_k\mathbf{e}_{z_k}. \end{aligned}$$

1. Predict Step

   (a) Predict the state.

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, 0)$$

   (b) Predict the error covariance matrix.

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{W}\mathbf{Q}\mathbf{W}^\top$$

2. Correct Step

   (a) Calculate the Kalman Gain.

$$\mathbf{K}_k = \frac{\mathbf{P}_k^-\mathbf{H}^\top}{(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{V}\mathbf{R}\mathbf{V}^\top)}$$

   (b) Correct the *a priori* state estimate

$$\hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-, 0))$$

   (c) Correct the *a posteriori* error covariance matrix estimate

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$$

Figure 4.6: *Equations of one Extended Kalman Filter Cycle. We assume that the state, its covariance and the noise values are already initialized. Note, that for simplicity the superscript k is not used here for the Jacobians, although, they have to be re-calculated after each predict-correct cycle.*

If we substitute this into Equation (4.18) we get

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\mathbf{e}_{z_k}.$$

Because $\mathbf{e}_{z_k}$ is the measurement residual, we also can write

$$\begin{aligned} \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k) && (4.32)\\ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-, 0)). && (4.33) \end{aligned}$$

Equation (4.33) can be used in the correct step of the Extended Kalman Filter algorithm to derive the *a posteriori* estimate for the state of the nonlinear system. The Kalman Gain $\mathbf{K}_k$ itself is calculated as in Equation (4.11) with the appropriate substitution for the measurement error covariance matrix given in (4.31):

$$\mathbf{K}_k = \frac{\mathbf{P}_k^-\mathbf{H}^\top}{(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{V}\mathbf{R}\mathbf{V}^\top)}$$

In Figure 4.6, the Extended Kalman Filter algorithm is given step by step.

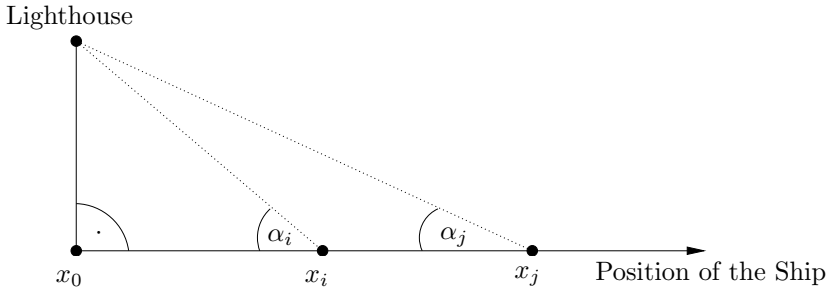

Figure 4.7: *A ship is sailing on the straight line perpendicular to the axis between x_0, the initial position of the ship, and the position of the lighthouse. x_i and x_j are sample positions of the ship which need to be estimated from the corresponding observable angles α_i and α_j.*

4.2.4 A Simple Example

The derivation of the Extended Kalman Filter presented in the previous section is a bit more complicated than the explanations of the "classic" Filter. In this section a simple example is examined to provide a better understanding of the EKF algorithm. Again, we will consider an instance of the general SLAM problem.

The section is structures as follows. Firstly, we will describe the specific problem in general. After that, the models for the system's state and process and the relation between the state and the measurement are presented. The section closes with some experiments on simulated data.

Problem Description

Imagine you are the skipper of a ship and your task is to sail a straight route of a certain length on an ocean. As you might infer from this sentence, the example deals more or less with the routing aspect of navigation. But we will focus on the localization and mapping problem. To be more concrete, as a skipper you need to localize your ship on that straight route. We assume that there is a lighthouse with an uncertainly known position to orientate at.

Your initial position is located in some distance from that lighthouse. You will sail in a perpendicular direction to the axis between the lighthouse and the initial ship position. It is obvious that the motion of a ship is smooth, so that changes in the velocity are unlikely.

You will be able to measure the angle between the current position of your ship and the lighthouse. Of course, these values will be more or less guesses than precise measurements. We assume that you are not able to measure your velocity which is normally the case.

This situation is depicted in Figure 4.7.

Process and Measurement Model

In this example we have two tasks. Firstly, we need to localize the position x of the moving ship on the straight route at every time step. Secondly, we have to refine our knowledge about the position y of the lighthouse.

Thus, the state $\mathbf{x}$ of the considered system contains three entities. The position x and velocity v_s of the ship are the first ones. Again, we choose a constant value for the velocity which represents an average value during the constant time period $\triangle k$. The third component of the state denotes the distance between the lighthouse and the initial position x_0 of the ship.

$$\mathbf{x} = \begin{bmatrix} x \\ v_x \\ y \end{bmatrix} = \begin{bmatrix} \text{Position of the Ship} \\ \text{Velocity of the Ship} \\ \text{Distance of the Lighthouse from } x_0 \end{bmatrix}$$

With this definition of the state, we have the following error covariance matrix $\mathbf{P}$ representing the uncertainty in the estimation of the state.

$$\mathbf{P} = \begin{bmatrix} xx & xv_x & xy \\ v_x x & v_x v_x & v_x y \\ yx & yv_x & yy \end{bmatrix}$$

The process, the system is subject to, is just the motion of the ship on that route. The process model f we will set up here, relates the state at time $k-1$ to k by calculating:

$$\begin{array}{rcl} \hat{x}(k) & = & \text{old position + old velocity per time intervall} \\ & = & \hat{x}(k-1) + \hat{v}_x(k-1)\triangle k \\ \hat{v}_x(k) & = & \text{constant velocity due to assumed smooth motion} \\ & = & \hat{v}_x(k-1) \\ \hat{y}(k) & = & \text{static landmark} \\ & = & \hat{y}(k-1) \end{array} \tag{4.34}$$

These equations are linear. Nevertheless, we will treat them as to be nonlinear and apply the EKF approach. We will see, that the EKF equations will reduce to the equations of the "classic" Kalman Filter.

As already mentioned, v_x just describes the average velocity between two time steps. Thus, it is just an approximation of the real velocity. The random difference between estimated and real velocity is modelled as process noise $\mathbf{w} = (w_0, w_1, w_2)^\top = (0, a\triangle k, 0)^\top$. As the state, $\mathbf{w}$ is a three-dimensional vector. Just the velocity is corrupted by noise. Therefore, just w_1 carries a value unequal to zero involving unmeasurable acceleration a:

$$p(\mathbf{w}) \sim N(0, \mathbf{Q})$$

$\mathbf{Q}$ is of the following form

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The variable σ_v denotes the standard deviation of the noise in the velocity.

If we would know the indivdual values for $\mathbf{w}$, we could derive the real state of the considered system by calculating $f(\mathbf{x}_{k-1}, \mathbf{w_{k-1}})$:

$$\begin{array}{rcl} x(k) & = & x(k-1) + (v_x(k-1) + w_1)\triangle k + w_0 \\ v_x(k) & = & v_x(k-1) + w_1 \\ y(k) & = & y(k-1) + w_2 \\ & = & x(k-1) + (v_x(k-1) + a(k-1)\triangle k)\triangle k \\ & = & v_x(k-1) + a(k-1)\triangle k \\ & = & y(k-1) \end{array} \tag{4.35}$$

Again, we assume $\mathbf{w}$ to be always equal to the mean of its distribution which is zero. Then we derive the process model $f(\hat{\mathbf{x}}_{k-1}, 0)$, as it is already formulated in Equation (4.34). To be able to predict the error covariance matrix $\mathbf{P}$ at each point in time, we need to derive the Jacobian matrix $\mathbf{A}$ containing the partial derivatives of Equation (4.34) with respect to the state $\mathbf{x}$ and the Jacobian matrix $\mathbf{W}$ containing the partial derivatives of Equation (4.34) with respect to the noise $\mathbf{w}$. Assuming that $\triangle k$ is equal to 1, as $\mathbf{A}$, we have:

$$\mathbf{A} = \begin{bmatrix} \frac{\partial x}{\partial x} & \frac{\partial x}{\partial v_x} & \frac{\partial x}{\partial y} \\ \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial v_x} & \frac{\partial v_x}{\partial y} \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial v_x} & \frac{\partial y}{\partial y} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Note, that this is the same as Equation (4.34) expressed as a linear transformation.

As $\mathbf{W}$, we have:

$$\mathbf{W} = \begin{bmatrix} \frac{\partial x}{\partial w_0} & \frac{\partial x}{\partial w_1} & \frac{\partial x}{\partial w_2} \\ \frac{\partial v_x}{\partial w_0} & \frac{\partial v_x}{\partial w_1} & \frac{\partial v_x}{\partial w_2} \\ \frac{\partial y}{\partial w_0} & \frac{\partial y}{\partial w_1} & \frac{\partial y}{\partial w_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Hence, $\mathbf{WQW}^\top = \mathbf{Q}$. Then, the equation to predict the error covariance equals the one for the standard Kalman Filter: $\mathbf{P}^-(k) = \mathbf{AP}(k-1)\mathbf{A}^\top + \mathbf{Q}$.

Now, let us consider the measurement model for our system. It provides the relation between the state $\mathbf{x}$ of the system and the measurement z of its output. Remember, as measurement we obtain the value for the angle α at each time step. If we have a look again at Figure 4.7, we can state, that the situation can be represented by a right triangle. Then, two definitions hold:

$$\begin{aligned} a^2 + b^2 &= c^2 \\ a &= c \cdot \sin\alpha \end{aligned}$$

We define the axis between the lighthouse and x_0 as a, the distance, the ship has covered till a certain point in time, as b and the connection between the lighthouse and the current position of the ship as the hypotenuse c. b is then equal to x in the state and a is the same as y. Thus, the measurement model to obtain the measurement z is

$$\hat{z}(k) = \alpha = \arcsin\left(\frac{y(k)}{\sqrt{(x(k))^2 + (y(k))^2}}\right). \tag{4.36}$$

Thus, we have a nonlinear measurement model h.

The value provided for α might be more or less a guess than a precise measurement. Therefore we have to introduce measurement noise v to model the difference between the real measurement and the predicted one. If we know the noise value for each time step, we would obtain z instead of $\hat{z}$ by calculating $h(\mathbf{x}_k, v_k)$:

$$z(k) = \alpha = \arcsin\left(\frac{y(k)}{\sqrt{(x(k))^2 + (y(k))^2}}\right) + v(k).$$

But this is not the case. Therefore, we model v as normally distributed measurement noise with zero mean and standard deviation σ_r.

$$p(v) \sim N(0, \sigma_r^2)$$

Now, we can assume v to be zero at each point in time which is the mean of its distribution. Then we obtain $h(\mathbf{x}, 0)$ as it is already formulated in Equation (4.36). The standard deviation is added to the calculation of the innovation covariance matrix $\mathbf{S}(k) = \mathbf{H}\mathbf{P}(k)^{-}\mathbf{H}^{\top}$ which is also one dimensional. Because we have a nonlinear model, the value for the variance is firstly transformed by $\mathbf{V}\sigma_\mathbf{r}\mathbf{V}^{\top}$ and then added.

As usual, the value we choose for σ_r indicates how we rate the quality of the measurement model.

Because we have a nonlinear measurement model, we need to derive the Jacobian matrices $\mathbf{H}$ and $\mathbf{V}$ for each point in time. $\mathbf{H}$ contains the partial derivatives of the measurement model $h(\mathbf{x}_k, 0)$ with respect to the state. It is of the following form:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial h}{\partial x} & \frac{\partial h}{\partial v_x} & \frac{\partial h}{\partial y} \end{bmatrix}$$

For $\frac{\partial h}{\partial x}$ we have

$$\frac{\partial h}{\partial x} = \frac{-xy}{\sqrt{1 - \frac{y^2}{x^2+y^2}}\sqrt{(x^2+y^2)^3}}$$

$\frac{\partial h}{\partial v_x}$ is equal to zero, because the velocity of the point is irrelevant in the measurement model. For $\frac{\partial h}{\partial y}$ we have

$$\frac{\partial h}{\partial y} = \frac{\frac{1}{\sqrt{x^2+y^2}} - \frac{y^2}{\sqrt{(x^2+y^2)^3}}}{\sqrt{1 - \frac{y^2}{x^2+y^2}}}$$

The covariance matrix $\mathbf{V}$ contains the partial derivatives of $h(\mathbf{x}_k, 0)$ with respect to the noise v. Thus, it is of the following form:

$$\mathbf{V} = \frac{\partial h}{\partial v}$$

Because the measurement noise v is an additive constant, $\frac{\partial h}{\partial v}$ is equal to 1.

Experiments on Simulated Data

In the previous section we derived the basis to apply the EKF approach to solve our problem: the process and measurement model. In this section, we will test these models on simulated data. We repeat the procedure from the simple example for the standard Kalman Filter. The initial values for the Filter reflect reality but are just known approximately. This is represented by an error covariance matrix $\mathbf{P}$ where the values in the main diagonal are unequal to zero. The values for the process and measurement noise are also choosen to represent the real values.

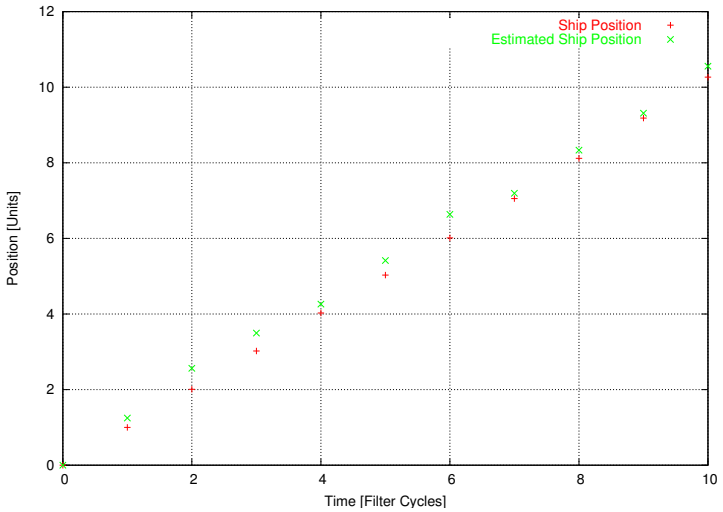

Figure 4.8: *The Simulation of the Problem of Estimating a the Position of a Ship by Orientating at a Lighthouse.*

To start the predict-correct cycle we initialize the state $\mathbf{x}$ and the error covariance matrix $\mathbf{P}$. For $\mathbf{x}$ we choose:

$$\mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ 20 \end{bmatrix}$$

These initial values are just uncertainly known. For $\mathbf{P}$ we choose:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In reality, the standard deviation of the process and measurement noise need to be determined prior to the application of the filter. Here, the values σ_v and σ_m reflect the real noise values.

$$\sigma_v = 0.02$$
$$\sigma_m = 0.02$$

We will run the filter on 10 simulated measurements. The results for the estimation of the ship's position are depicted in Figure 4.8. In Figure 4.9, the estimated lighthouse position is opposed to the real one. In Figure 4.10, the behaviour of the error covariance $\mathbf{P}$ is depicted during the ten filter cycles. We can note that the uncertainty about the position of the ship decreases first and then starts to increase slowly. This is due to the more and more influencing measurement noise. The farer the ship is getting away from its starting point the lesser the measured angle will change its value. The measurement noise stays at a constant level and will therefore increase its influence concerning uncertainty

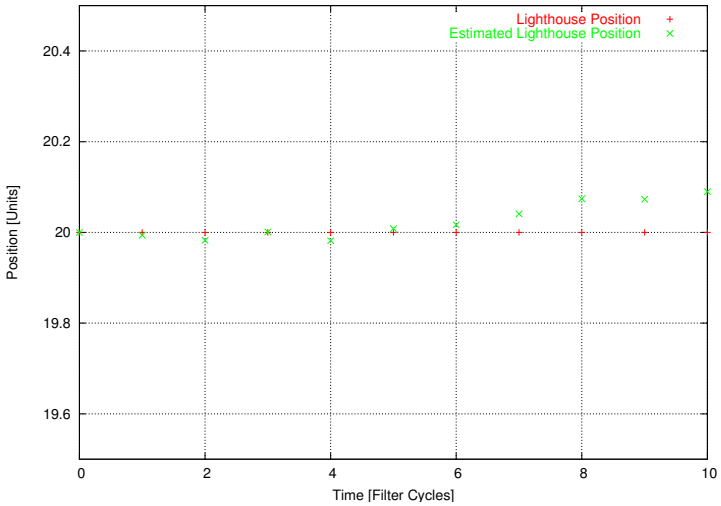

Figure 4.9: *The Results for the Mapping of the Lighthouse.*

about the correctness of the infered position of the ship. Small changes in the value of the angle will cause larger deviations in the estimation of the ship's position and therefore a large uncertainty about the state estimate.

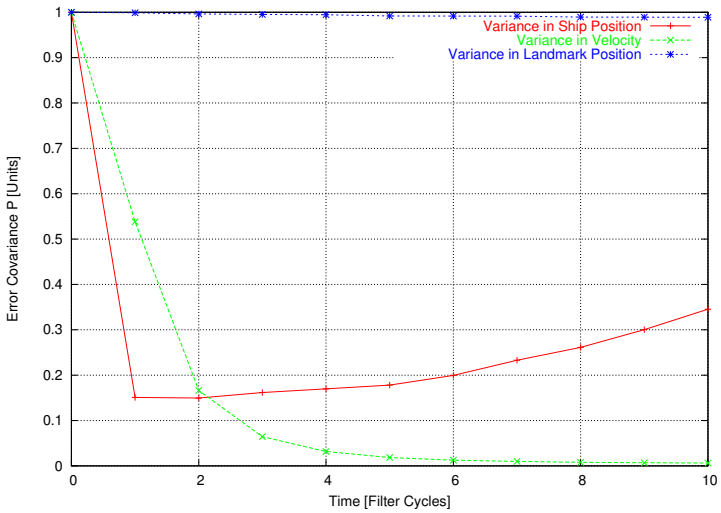

Figure 4.10: *The Error Covariance matrix* $\mathbf{P}$*. After just one iteration, the uncertainty about the ship's position has decreased massivly. Then it increases slightly. In contrast to that, the uncertainty about the velocity has nearly fallen to zero.*

Chapter 5

Real-Time SLAM

Our goal is to develop a vision system to enable a mobile robot to navigate reliably and safely in an unknown environment. Therefore, we need to determine the position and orientation of a vision sensor mounted on a mobile robot inside a certain environment and simultaneously construct a three-dimensional map of this environment. The problem should be solved in real-time so that the robot knows everything of interest at dense consecutive points in time. It is important that the accuracy of the estimation of the position etc. does not decrease over time but stays at a rather constant level to ensure a reliable detection of obstacles for a secure navigation.

In the following we present the application of the Extended Kalman Filter to the problem of SLAM with two different vision sensors. We will start with a general introduction. After that, we will generalise our task from a mobile robot situated on a plane to a camera moving freely in space. Firstly, this camera will be monocular. Secondly, we discuss the application of a stereo camera sensor. Finally, the EKF-algorithm is summarised again.

For the sake of simplification we also assume that the initial position of the camera and a fixed number of features are already initialised.

5.1 The SLAM Problem

Our approach is Kalman Filter based. Thus, the considered system, a mobile robot in an unknown environment, is represented by a state containing instances of all relevant information at the current time. After a constant period of time, in the following always denoted as $\triangle k$, the state needs to be updated to adopt the changes due to the robot's motion. Measurements of the environment are needed for this update. In our case we detect the relative position of landmarks to the location of the robot within the environment via a vision sensor. These landmarks are represented by three-dimensional points in the surrounding world, also referred to as *3D point features*. Thus, the information available from a picture is greatly reduced to just a few points: the projections of these world points. On the one hand, this means that the available solution for our problem might suffer from local inaccuracies, on the other hand, we have a massive reduction of complexity. So, we need to find a balance between the two aspects simplification and precision to obtain real-time performance *and*

reliable navigation.

The extraction of the corresponding projections of the landmarks out of the picture is done by a feature tracker. As already said at the beginning of this work, it is handled here as a black box and assumed to work correctly. It provides measurements given in image coordinates at regular intervals $\triangle k$.

Due to the motion of the robot just some landmarks will be visible at a considered point in time, the other might be simply out of view. The prerequisite to avoid a gradually increasing divergence between the state estimate and the real state of the system in question is to be able to re-recognise landmarks after periods of neglect.

This issues might be clarified by the following example: Imagine you have to draw a very long straight line on the ground but you are just equipped with a 1m ruler. At the beginning there might occur just small inaccuracies, because you are able to compare the current fragment of the line with the first one. But the longer the line gets the more it will diverge from the original orientation. This is due to the small errors in each segment which will accumulate with increasing line length. A solution would be to provide the original orientation for farther line sections by setting kind of flags placed at regular intervals which are also visible after a large distance.

The application of this concept to the localisation and mapping problem is exemplary clarified in Picture 5.1 which is adapted from [30]. A robot, equipped with any sensor device, e.g., with a monocular camera, is moving along the dashed line in an environment with eight feature points aligned in two rows of four landmarks each. Initially, the robot knows its starting position but has no idea where the landmarks are. We assume that it can determine and distinct them well.

As the robot starts moving, the uncertainty about its position increases steadily. This is indicated by the growing grey error ellipses in the Picture 5.1(a) to 5.1(c). During the robot's motion, it can obtain noisy measurements of the relative position of nearby landmarks. From these values it estimates the position of the corresponding feature points. The uncertainty in the position estimate of the landmark is closely related to the uncertainty in the robot's location. Because the first feature was measured from the known starting point of the robot, there is almost no uncertainty about the feature's position. All the other landmarks were measured from more and more uncertain robot locations. This contributes to the uncertainty in their position estimates indicated by the growing grid ellipses.

In Picture 5.1(d), the feature firstly measured from the starting point of the robot is re-recognised and measured again. Because the location of this landmark is quite certain it acts as a kind of flag for the robot's position. As a result, the uncertainty in the position estimate of the robot as well as of the other landmarks reduces enormously.

The question why the estimates concerning the other seven features are also corrected might be posed. Remember Chapter 4, where we introduced the Kalman Filter approach. The considered system is described by a state $\mathbf{x}$ and an error covariance matrix $\mathbf{P}$. This matrix represents the uncertainty in the whole state estimate as a *multivariate* Gaussian. This means that all according random variables take their values in mutual dependency. The Gaussian can be decomposed into its individual parts for each component contained in the state to derive the single error ellipses. On the main diagonal of $\mathbf{P}$ you can

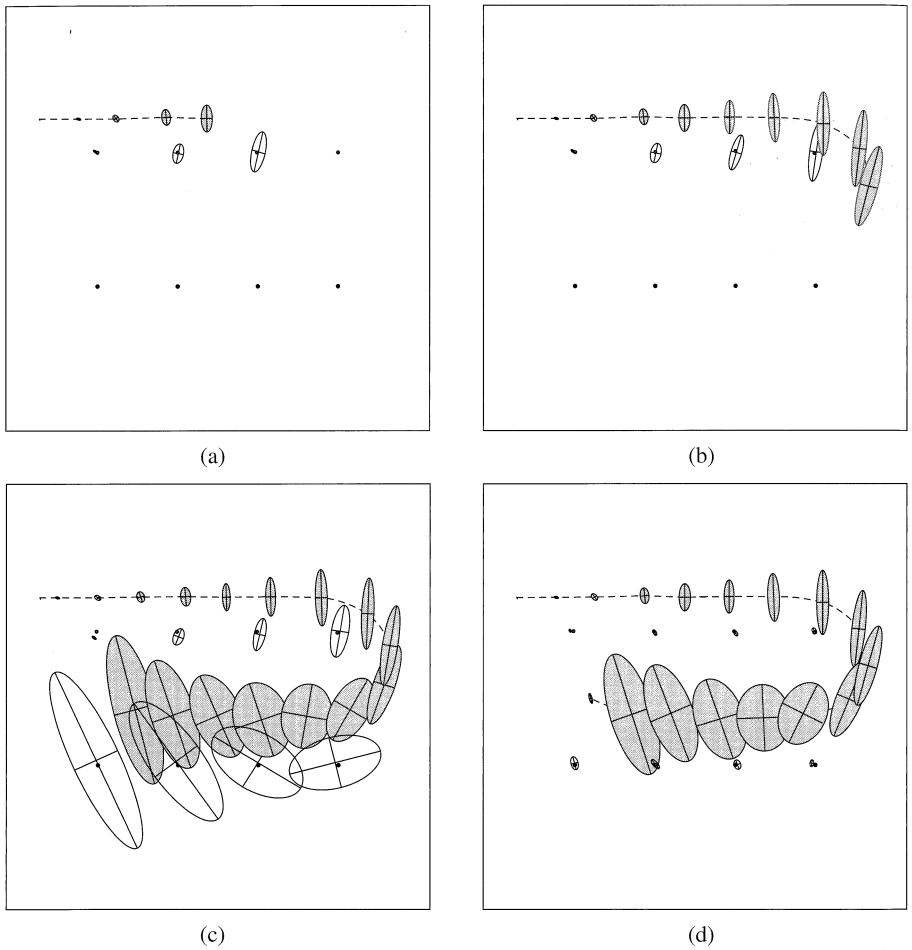

Figure 5.1: *A robot moves along the dashed line in an environment with eight feature points aligned in two rows of four landmarks each. It tries to localise itself and the features by using a sensor device, e.g., a monocular camera.(a)-(c) As the robot moves on the uncertainty about its own position (indicated by the grey ellipses) and about the position of the measured features (indicated by the grid ellipses) increases. (d) The firstly measured feature, whose position is known quite certainly, is re-recognised. The robot's position estimation and the estimates concerning the landmarks are corrected.*

find the variances of the distribution. Beyond the main diagonal, there are the covariances of pairs of variables. They indicate the amount of correlation between these variables. As already said above, the estimate of the position of the robot and of the landmarks are highly correlated. Thus, a reduction of the uncertainty in the estimate of the robot's position will also decrease the uncertainty in the estimate of the feature position.

5.2 Using a Single Camera

In this section we are going to concentrate on the more general problem of tracking a freely moving camera instead of a robot driving on a plane.

Our considered system consists of several static 3D point features in the world and a monocular camera moving with a certain velocity in this environment. Therefore, we have two frames of reference: the fixed world coordinate frame, indicated by superscript W, and the moving camera coordinate frame, indicated by superscript C. Measurements of the landmarks are taken relative to the camera frame. But they are related to fixed 3D positions in the environment based on the (uncertainly) known robot's position also in the according world frame. If the landmarks would be just related to the camera frame, the robot will not be able to re-recognise them after periods of neglect because the 3D features are not static. But as already mentioned in the introductive chapter, this ability is essential to avoid an increasing divergence between the real and estimated position of the robot.

In the following, we begin with designing the state as a description of our considered system. It will contain the relevant information for our approach to the SLAM problem. Then we formulate the process model to be able to predict the motion of the camera. This prediction is corrected with a measurement provided by the camera. For this purpose we will also set up a measurement model which relates the state to the measurements.

5.2.1 Model for the Dynamical System to be Estimated

Our considered system can be described by three entities: state (position and orientation) of the camera, motion of the camera and position of the 3D point features.

Therefore, as components of the state vector $\mathbf{x}$ we choose the 3D position $\mathbf{t}^{\mathsf{W}} = (t_x, t_y, t_z)^\top$ and the orientation $\mathbf{q}^{\mathsf{CW}} = (q_0, q_1, q_2, q_3)^\top$ of the camera projection centre with respect to the world coordinate frame.

For the representation of the motion of the camera we add a constant linear velocity $\mathbf{v}^{\mathsf{W}} = (v_x, v_y, v_z)^\top$ and angular velocity $\omega^{\mathsf{CW}} = (\omega_x, \omega_y, \omega_z)^\top$ to the state, each represented by three-dimensional vectors. This does not mean that we assume that the robot moves at constant velocities over the whole time but rather that these are the expected average values in the period $\triangle k$ of time. Accelerations are not measurable in a picture obtained at an instance of time. Therefore, they are presumed to occur randomly and are handled as uncertainty about the motion of the camera. Again we apply the noise model already introduced in Chapter 4. We assume accelerations to be realisations of a normally distributed process noise. This is discussed in more detail in Section 5.2.2.

All the last mentioned components can be summarised in the vector $\mathbf{x}_v$ with a total dimension of 13.

$$\mathbf{x}_v = \begin{bmatrix} \mathbf{t}^{\mathsf{W}} \\ \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} \end{bmatrix} = \begin{bmatrix} \text{Position of the camera} \\ \text{Orientation of the camera} \\ \text{Linear velocity} \\ \text{Angular velocity} \end{bmatrix}$$

Additionally, the landmarks in the world also represented by n three-dimensional vectors $\mathbf{y}_i^{\mathsf{W}} = (x_i^{\mathsf{W}}, y_i^{\mathsf{W}}, z_i^{\mathsf{W}})^\top$ are contained in the state.[1] Therefore we have

$$\mathbf{x} = \begin{bmatrix} \mathbf{t}^{\mathsf{W}} \\ \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} \\ \mathbf{y}_1^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_i^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_n^{\mathsf{W}} \end{bmatrix}$$

The total dimension m of the state $\mathbf{x}$ is therefore $m = 13+3n$. Because each row and column is related to one component of the state, the according error covariance matrix $\mathbf{P}$ is a $(13 + 3n) \times (13 + 3n)$ matrix.

$$\mathbf{P} = \begin{bmatrix} \mathbf{t}^{\mathsf{W}}\mathbf{t}^{\mathsf{W}} & \mathbf{t}^{\mathsf{W}}\mathbf{q}^{\mathsf{CW}} & \dots & \mathbf{t}^{\mathsf{W}}\mathbf{y}_n^{\mathsf{W}} \\ \mathbf{q}^{\mathsf{CW}}\mathbf{t}^{\mathsf{W}} & \mathbf{q}^{\mathsf{CW}}\mathbf{q}^{\mathsf{CW}} & \dots & \mathbf{q}^{\mathsf{CW}}\mathbf{y}_n^{\mathsf{W}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_n^{\mathsf{W}}\mathbf{t}^{\mathsf{W}} & \mathbf{y}_n^{\mathsf{W}}\mathbf{q}^{\mathsf{CW}} & \dots & \mathbf{y}_n^{\mathsf{W}}\mathbf{y}_n^{\mathsf{W}} \end{bmatrix}$$

As already said before, its main diagonal contains the variances of all components of the state $\mathbf{x}$ and all other values specify the correlation between pairs of these variables.

5.2.2 Process Model

The state $\mathbf{x}$, defined in the section before, is subject to a transformation over time. In our case this transformation is the motion of the camera. We are not able to model this motion up to absolute precision because we do not know the "intention" of the robot or camera. Instead, we accept a certain discrepancy between our model and reality and form this as some probabilistic entity represented by the *process noise*. In our case we assume that the camera moves with an average linear velocity $\mathbf{v}$ and angular velocity ω and ignore possible accelerations. The individual values for acceleration at each point in time, whether linear or angular, are unknown and not determinable by measurements provided by a vision sensor. So, each is assumed to occur randomly. This randomness can

[1] The number of landmarks is allowed to change dynamically over time. New landmarks might be added or "bad" features are deleted. Unless this is no subject of this work, we assume the number of landmarks in the state to be static.

be modelled as normally distributed white noise with zero mean and a specific variance.

Let $\mathbf{a} = (a_x, a_y, a_z)^\top$ be the random vector for the linear acceleration. With $\mathbf{Q}_a$ as the corresponding covariance matrix, it can be modelled as

$$p(\mathbf{a}) \sim N(0, \mathbf{Q}_a) \tag{5.1}$$

$\mathbf{Q}_a$ contains the variances of the distribution for each component of $\mathbf{a}$:

$$\mathbf{Q}_a = \begin{bmatrix} \sigma_{a_x}^2 & 0 & 0 \\ 0 & \sigma_{a_y}^2 & 0 \\ 0 & 0 & \sigma_{a_z}^2 \end{bmatrix} \tag{5.2}$$

Linear acceleration is usually defined as a change in velocity $\triangle \mathbf{v}^{\mathsf{W}}$ per time period $\triangle k$. Remember that $\triangle k$ denotes the time covered between two consecutive points in time. The average value of velocity $\mathbf{v}^{\mathsf{W}}$ is enhanced by the random value $\mathbf{a}$ at each time step and forms the new average velocity $\mathbf{v}_{new}^{\mathsf{W}}$ for the next time step. Expressed in a linear equation, we have

$$\begin{aligned} \mathbf{v}_{new}^{\mathsf{W}} &= \mathbf{v}^{\mathsf{W}} + \mathbf{a}^{\mathsf{W}} \triangle k \\ &= \mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}} \end{aligned}$$

Let the second random vector $\alpha = (\alpha_x, \alpha_y, \alpha_z)^\top$ be the angular acceleration. With the covariance matrix $\mathbf{Q}_\alpha$, it is represented by

$$p(\alpha) \sim N(0, \mathbf{Q}_\alpha) \tag{5.3}$$

$\mathbf{Q}_\alpha$ is modelled analogous to $\mathbf{Q}_a$. Also, angular acceleration is defined as a change in (angular) velocity $\triangle \omega^{\mathsf{W}}$ per time period $\triangle k$. This change contributes to the assumed average angular velocity at each point in time and forms the average value $\omega_{new}^{\mathsf{CW}}$ after the next time step. Expressed in a linear equation, we have

$$\begin{aligned} \omega_{new}^{\mathsf{CW}} &= \omega^{\mathsf{CW}} + \alpha^{\mathsf{CW}} \triangle k \\ &= \omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}} \end{aligned}$$

Thus, the process noise consist of two components: random linear and angular acceleration. We summarise them in the noise vector $\mathbf{w}$.

$$\mathbf{w} = \begin{bmatrix} \mathbf{V}^{\mathsf{W}} \\ \mathbf{\Omega}^{\mathsf{CW}} \end{bmatrix} = \begin{bmatrix} \mathbf{a}^{\mathsf{W}} \triangle k \\ \alpha^{\mathsf{CW}} \triangle k \end{bmatrix}.$$

As already mentioned above, the process noise in all is modelled as normally distributed white noise with zero mean and a certain variance, here represented by the covariance matrix $\mathbf{Q}$.

$$p(\mathbf{w}) \sim N(0, \mathbf{Q})$$

The covariance matrix $\mathbf{Q}$ of the process noise carries the appropriate covariance matrices given in Equation (5.1) and (5.3).

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_a & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_\alpha \end{bmatrix}$$

For simplification, it is assumed that linear and angular acceleration is uncoupled in contrast to reality. Therefore, the covariances are zero. The choice of the values for σ_a and σ_α indicates what kind of motion we expect. If we choose small values we estimate a smooth movement of the camera where accelerations are very unlikely. But then, we will not be able to cope with motions were sudden rapid movements occur. To track these ones, we need to set the variances to high values. The rate of growth of uncertainty about the predicted state of our system will be higher than with small values for σ_a and σ_α.

So far, we have just discussed the change of the velocity components in the state $\mathbf{x}$ between two points in time and in the course of that also about the process noise. But how are the position and orientation values influenced by these terms?

In general, an average linear velocity is defined as a certain distance $\triangle \mathbf{s}^{\mathsf{W}}$ covered in a specified period of time $\triangle k$.

If we have given the position $\mathbf{t}^{\mathsf{W}}$ of the camera and the appropriate average velocity at time $(k - \triangle k)$, we can derive the expected position at time k. In the following, estimates for k are labelled by subscript *new* whereas a subscript for the values at time $(k - \triangle k)$ is omitted to simplify the notation.

$$\mathbf{t}^{\mathsf{W}}_{new} = \mathbf{t}^{\mathsf{W}} + (\mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}})\triangle k.$$

The average angular velocity is defined as a certain amount of rotation given in radians around some axis. All these informations are contained in the three-dimensional vector ω^{CW}: Its orientation specifies the axis and its magnitude represents the amount of rotation in radians in a certain period of time $\triangle k$.

So, if we have given the orientation $\mathbf{q}^{\mathsf{CW}}$ and the average angular velocity of the camera at time $(k - \triangle k)$ we can derive the new orientation at the current time k.

$$\mathbf{q}^{\mathsf{CW}}_{new} = \mathbf{q}^{\mathsf{CW}} \times \mathbf{q}((\omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}})\triangle k)$$

The term $\mathbf{q}((\omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}})\triangle k)$ refers to the calculation of a quaternion from an angle θ and axis $\mathbf{x}$ (Equation (3.1)).

The positions $\mathbf{y}^{\mathsf{W}}_i$ of the 3D features does not change over time because they are assumed to be static.

We can combine the last equations and statements to the *process model* where the change of the state of the system between two consecutive points in time is modelled.

$$\mathbf{x}_{new} = f(\mathbf{x}, \mathbf{w}) = \begin{bmatrix} \mathbf{t}^{\mathsf{W}}_{new} \\ \mathbf{q}^{\mathsf{CW}}_{new} \\ \mathbf{v}^{\mathsf{W}}_{new} \\ \omega^{\mathsf{CW}}_{new} \\ \mathbf{y}^{\mathsf{W}}_{1,new} \\ \vdots \\ \mathbf{y}^{\mathsf{W}}_{i,new} \\ \vdots \\ \mathbf{y}^{\mathsf{W}}_{n,new} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{\mathsf{W}} + (\mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}})\triangle k \\ \mathbf{q}((\omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}})\triangle k) \times \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}} \\ \mathbf{y}^{\mathsf{W}}_1 \\ \vdots \\ \mathbf{y}^{\mathsf{W}}_i \\ \vdots \\ \mathbf{y}^{\mathsf{W}}_n \end{bmatrix}$$

We will not know the individual values for the linear and angular acceleration at each time step. Therefore, we assume it to be zero, which is the mean of their

distribution. Then, we have

$$\hat{\mathbf{x}}_{new} = f(\hat{\mathbf{x}}, 0) = \begin{bmatrix} \hat{\mathbf{t}}^{\mathsf{W}}_{new} \\ \hat{\mathbf{q}}^{\mathsf{CW}}_{new} \\ \hat{\mathbf{v}}^{\mathsf{W}}_{new} \\ \hat{\omega}^{\mathsf{CW}}_{new} \\ \hat{\mathbf{y}}^{\mathsf{W}}_{1,new} \\ \vdots \\ \hat{\mathbf{y}}^{\mathsf{W}}_{i,new} \\ \vdots \\ \hat{\mathbf{y}}^{\mathsf{W}}_{n,new} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{r}}^{\mathsf{W}} + \hat{\mathbf{v}}^{\mathsf{W}} \triangle k \\ \hat{\mathbf{q}}(\hat{\omega}^{\mathsf{CW}} \triangle k) \times \hat{\mathbf{q}}^{\mathsf{CW}} \\ \hat{\mathbf{v}}^{\mathsf{W}} \\ \hat{\omega}^{\mathsf{CW}} \\ \hat{\mathbf{y}}^{\mathsf{W}}_{1} \\ \vdots \\ \hat{\mathbf{y}}^{\mathsf{W}}_{i} \\ \vdots \\ \hat{\mathbf{y}}^{\mathsf{W}}_{n} \end{bmatrix}$$

Due to the non-linearity of the quaternion multiplication and calculation of a quaternion from an angle-axis representation of rotation, this model is nonlinear. As already explained in Section 4.2 we have to linearise it about the current estimate to apply the Kalman Filter approach to a nonlinear system. Therefore, we create the Jacobian matrix $\mathbf{A}$ containing the partial derivatives of the process model f with respect to the state $\mathbf{x}$. It is used to project the uncertainty about one estimate at the previous point in time, given in the error covariance matrix $\mathbf{P}$, to the next point in time. The Jacobian matrix $\mathbf{A}$ is given in detail in Appendix A.

To involve the noise in the process model we also need to create the Jacobian matrix $\mathbf{W}$ containing the partial derivatives of the process model f with respect to the noise vector $\mathbf{w}$. It is also given in detail in Appendix A.

Besides predicting the state $\hat{\mathbf{x}}_{new}$ by calculating $f(\mathbf{x}, 0)$, we also predict the uncertainty about the state estimate $\mathbf{P}_{new}$ by

$$\mathbf{P_{new}} = \mathbf{APA}^\top + \mathbf{WQW}^\top$$

As already done in Chapter 4, we go on omitting the time index k for the Jacobian matrices although they are different at each time step.

5.2.3 Output of the System

The camera we try to localise within the environment provides the measurements we use to infer the state of the system. As already said above, the information of one picture delivered by the camera is reduced to a set of 2D points which are projections of 3D landmarks situated in the environment. The position and orientation of the camera affect which features are projected onto the image plane. Thus, the output of the system depends on its state.

The measurement is described by a vector $\mathbf{z}$ containing coordinates $\mathbf{z}_i = (x^{\mathsf{I}}_i, y^{\mathsf{I}}_i)^\top$ of the 2D projections of the currently visible landmarks. Let l features be visible, then we have

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_i \\ \vdots \\ \mathbf{z}_l \end{bmatrix}.$$

The corresponding covariance matrix $\mathbf{S}_{2l\times 2l}$ is defined as the uncertainty about the predicted measurement. Each row and column is related to one component of the measurement vector. Its main diagonal contains the variances of the components of $\mathbf{z}$. All the other values represent the amount of correlation between the measurements.

5.2.4 Measurement Model

The measurement model specifies how the output depends on the state of the system. Here, we have projections of points in the world coordinate frame onto the image plane of a monocular camera. If we remember Section 3.2.1, the relation between the projective 3D world coordinates $\tilde{\mathbf{y}}^{\mathsf{W}}_{(\mathbf{4})}$ of a point and the image coordinates of its projection $\tilde{\mathbf{y}}^{\mathsf{I}}_{(\mathbf{3})}$ is established by two transformation matrices. First, we need to shift and rotate the world coordinates into the camera frame by using the Matrix (3.5).

$$\tilde{\mathbf{y}}^{\mathsf{C}} = \mathbf{D}^{\mathsf{CW}}\tilde{\mathbf{y}}^{\mathsf{W}}$$

The image coordinates of the projection of the point in camera coordinates are derived by applying Matrix (3.4).

$$\tilde{\mathbf{y}}^{\mathsf{I}} = \mathbf{F}^{\mathsf{IC}}\tilde{\mathbf{y}}^{\mathsf{C}}$$

Now, we need to fill these matrices with the appropriate values. The external parameters to convert world into camera coordinates are given in the state as a translation $\mathbf{t}^{\mathsf{W}}$ and a rotation $\mathbf{q}^{\mathsf{W}}$. Expressed in a linear transformation, the camera coordinates of a landmark $\mathbf{y}^{\mathsf{W}}_i$ can be obtained by

$$\mathbf{y}^{\mathsf{C}}_i = \begin{bmatrix} x^{\mathsf{C}}_i \\ y^{\mathsf{C}}_i \\ z^{\mathsf{C}}_i \end{bmatrix} = \mathbf{R}^{\mathsf{CW}}(\mathbf{y}^{\mathsf{W}}_i - \mathbf{t}^{\mathsf{W}}) \tag{5.4}$$

Here, $\mathbf{R}^{\mathsf{CW}}$ refers to the rotation matrix derived by the quaternion $\mathbf{q}^{\mathsf{CW}}$ and is used to simplify the notation. More about the relation between several representations of rotations is given in Appendix B.

As shown in Equation (3.2) and (3.3) we can easily derive the Euclidean image coordinates by a division. Thus, the measurement model h_i for a single world point $\hat{\mathbf{y}}^{\mathsf{W}}_i$ is

$$\hat{\mathbf{y}}^{\mathsf{I}}_i = h_i(\mathbf{x}, 0) = \begin{bmatrix} x^{\mathsf{I}}_i \\ y^{\mathsf{I}}_i \end{bmatrix} = \begin{bmatrix} \frac{x^{\mathsf{C}}_i}{z^{\mathsf{C}}_i} \\ \frac{y^{\mathsf{C}}_i}{z^{\mathsf{C}}_i} \end{bmatrix} \tag{5.5}$$

You might have noticed, that in Equation (5.5) we deal with normalised image coordinates. This is indicated by the omitted focal length f from the Equations (3.2) and (3.3) and the lack of internal parameters, like pixel dimension or position of the principal point, described in Section 3.2.1. The problem with this is, that the camera sensor just provides measurements in pixel coordinates. Thus, we need to convert these measurements into the appropriate representation to be able to apply Equation (5.5).

This is the place where the *measurement noise* is introduced. The mapping of pixel on image coordinates is not unique. One pixel merges several image

points. The amount of image points referring to one pixel depends on the resolution of the camera sensor or in other words, to the dimension of one pixel given in image points. Therefore, the measurement model is augmented by random noise values v_x and v_y in $\mathsf{x}-$ and $\mathsf{y}-$direction. They represent the discrepancy between real and estimated image coordinates of the projections of world points onto the image plane. In Equation (5.5) no noise is introduced and therefore $\hat{\mathbf{y}}_i^{\mathsf{I}}$ is labelled with a hat. We will not know the individual random noise values after each time step. Thus, the measurement noise is in the following assumed to be normally distributed white noise with zero mean and certain variance values σ_x^2 and σ_y^2 which refer to the pixel dimension. We combine the mentioned random values in the measurement noise vector $\mathbf{v} = (v_x, v_y)^\top$ and the variances in the covariance matrix $\mathbf{R}$.

$$p(\mathbf{v}) \sim N(0, \mathbf{R})$$

The covariance matrix is of the following form

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

By setting the covariance values, situated outside the main diagonal, to zero we assume that the noise in $\mathsf{x}-$ and $\mathsf{y}-$direction is uncorrelated.

If we augment Equation (5.5) with the noise $\mathbf{v}$ we have

$$h_i(\mathbf{x}, \mathbf{v}) = \mathbf{y}_i^{\mathsf{I}} = \begin{bmatrix} x_i^{\mathsf{I}} \\ y_i^{\mathsf{I}} \end{bmatrix} = \begin{bmatrix} \frac{x_i^{\mathsf{C}}}{z_i^{\mathsf{C}}} + v_x \\ \frac{y_i^{\mathsf{C}}}{z_i^{\mathsf{C}}} + v_y \end{bmatrix} \tag{5.6}$$

We derive the complete measurement model $h(\mathbf{x}, \mathbf{v})$ by substituting Equation (5.4) for $\mathbf{y}_i^{\mathsf{C}} = (x_i^{\mathsf{C}}, y_i^{\mathsf{C}}, z_i^{\mathsf{C}})^\top$ in Equation 5.6.

To summarise, we have

$$\mathbf{z} = h(\mathbf{x}, \mathbf{v}) = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_i \\ \vdots \\ \mathbf{z}_n \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}, \mathbf{v}) \\ \vdots \\ h_i(\mathbf{x}, \mathbf{v}) \\ \vdots \\ h_n(\mathbf{x}, \mathbf{v}) \end{bmatrix}$$

As already said before, we are just able to approximate the individual noise values for v_x and v_y at each point in time. Therefore, we assume them to be zero, which is the mean of the according distribution. The projections for each world point are therefore predicted by Equation (5.5). Then, $h(\mathbf{x}, 0)$ is represented by

$$\hat{\mathbf{z}} = h(\mathbf{x}, 0) = \begin{bmatrix} \hat{\mathbf{z}}_1 \\ \vdots \\ \hat{\mathbf{z}}_i \\ \vdots \\ \hat{\mathbf{z}}_n \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}, 0) \\ \vdots \\ h_i(\mathbf{x}, 0) \\ \vdots \\ h_n(\mathbf{x}, 0) \end{bmatrix}$$

where $h_i(\mathbf{x}, 0)$ is calculated by Equation (5.5). $\mathbf{y}_i^{\mathsf{C}}$ in (5.5) is derived by Equation 5.4.

Due to the nonlinearity of rotations and the projection, the measurement model is also nonlinear. To apply the Kalman Filter approach to this problem we need to linearise the equations about the current estimate. Firstly, this means to calculate the Jacobian matrix $\mathbf{H}$ which contains the partial derivatives of the measurement model h with respect to the state $\mathbf{x}$. The Jacobian is given in detail in Appendix A. The input for the measurement model will not be the real state $\mathbf{x}$ of the system like it is assumed so far, because we simply do not know it. Instead we have an estimate $\hat{\mathbf{x}}$ derived from the process model. This estimate is accompanied by an uncertainty about it represented by the covariance matrix $\mathbf{P}$. The Jacobian matrix $\mathbf{H}$ is needed to project this uncertainty into the measurement space. Thus, depending on the uncertainty about the state, the uncertainty about the predicted measurement can be calculated.

But $\mathbf{P}$ is not the sole component that contributes to the uncertainty about the measurement prediction. The measurement noise also affects the amount of uncertainty. Thus, we secondly need to calculate the Jacobian matrix $\mathbf{V}$ which contains the partial derivatives of h with respect to the noise vector $\mathbf{v}$ to project the measurement noise into the measurement space. $\mathbf{V}$ is also given in Detail in Appendix A.

Thus, if we have a state estimate $\hat{\mathbf{x}}$, we can predict the measurement $\hat{\mathbf{z}}$ by calculating $h(\hat{\mathbf{x}}, 0)$. The uncertainty about the predicted measurement is represented by the covariance matrix $\mathbf{S}$ and is calculated by

$$\mathbf{S} = \mathbf{HPH}^\top + \mathbf{VRV}^\top$$

5.3 Using a Stereo Camera

The difficulty with using a monocular camera to solve the SLAM problem is that the depth of a 3D feature point is not determinable from a single 2D projection. The process of projection is not invertible. Remember Figure 3.1 where the pinhole model is depicted. All points situated on one ray from the camera projection centre through the image plane result in the same 2D projection: the intersection point between ray and image plane.

Obviously, we have a problem with initialising the position of a 3D feature point via a single 2D projection. For simplification we assumed in the previous section that some 3D point features are already initialised with a specific uncertainty. Nevertheless, Davison [9] presented a solution for the initialisation problem via a monocular camera.

If we already have some predefined feature points with predefined uncertainty, the uncertainty about the distance between the camera and each world point will not decrease after the first Kalman Filter cycle. Opposed to the depth, the estimates about the feature position in horizontal and vertical direction are much more precise because the 2D projection provides sufficient information to calculate them. Two or more different corresponding projections of one world point enables to obtain more confidence about the depth of the feature. Although, we acquire these additional information during the motion of the camera, the uncertainty about depth will just decrease slowly. This is due to the fact, that the 3D position of the feature considered is computed based

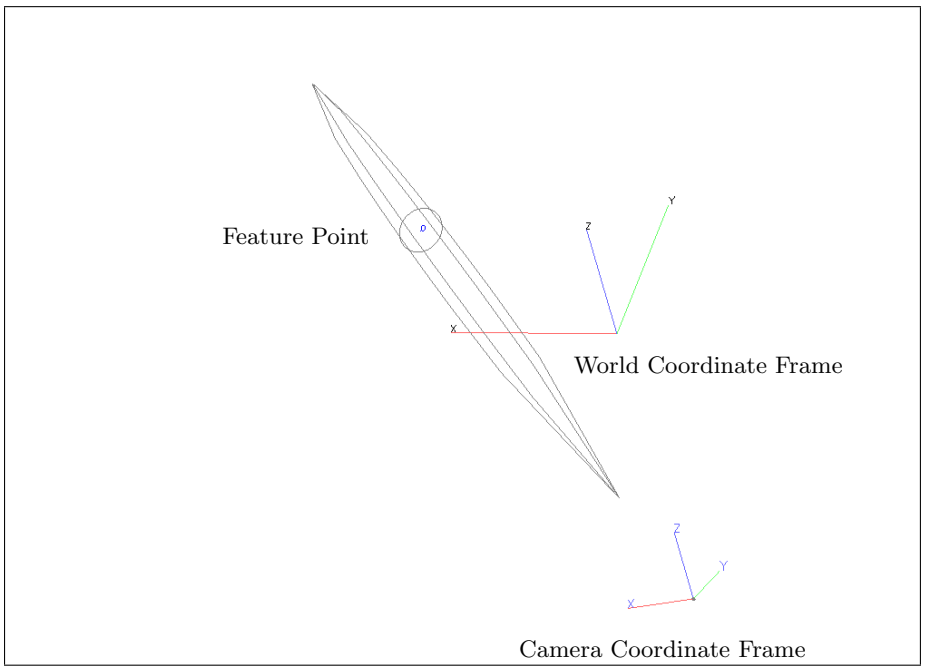

Figure 5.2: *Uncertainty, Represented by the Grid Ellipsoid, about the 3D position of a Point Feature after one Kalman Filter Cycle. The measurement is obtained by a single camera.*

on the uncertainly known state of the camera and on noisy measurements. The calculated values are therefore also accompanied by uncertainty. In Figure 5.2 the situation after one Kalman Filter cycle is depicted.

This drawback in the estimation of the feature point position can be solved by using a stereo camera. If the two corresponding projections of one world point are determined and the stereo camera system is in standard stereo geometry, the 3D coordinates can be calculated by Equations (3.8) to (3.10). Thus, we can easily initialise the 3D position of landmarks and the uncertainty about this position will decrease similarly in all directions. The situation is depicted in Figure 5.3. The same initial state and error covariance is used as for the situation where the image is obtained by a single camera. When comparing with Figure 5.2, it can be noted that the uncertainty about depth is smaller than with using a monocular vision sensor.

Nevertheless, we will stick to a certain number of pre-initialised features with pre-initialised uncertainty.

In the following we present a vision based SLAM approach where we use a stereo camera instead of a monocular camera. First, we will set up the state as the description of the system. This is followed by the process model which relates the state at one point in time to the next. The state estimate provided by the process model is augmented by a measurement of the system's output. Therefore, we will introduce the measurement vector and the measurement model which relates the state to the measurements.

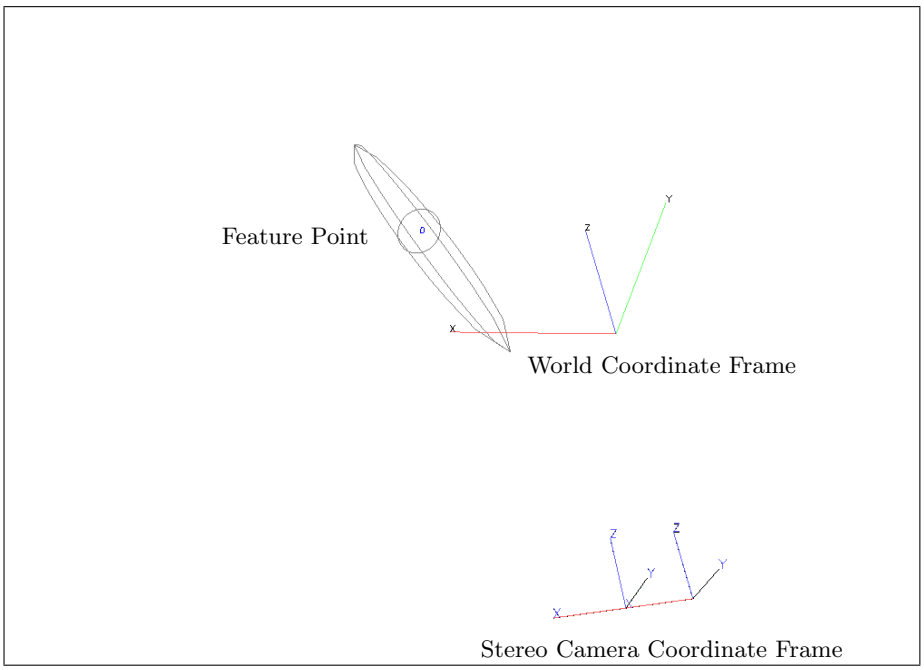

Figure 5.3: *Uncertainty, Represented by the Grid Ellipsoid, about the 3D position of a Point Feature after one Kalman Filter Cycle. The measurement is obtained by a stereo camera.*

5.3.1 System and Process Model

The system we consider here resembles the one in Section 5.2. It consists of several 3D point features situated in an environment and a camera moving in this environment with a certain velocity. The sole difference is the vision sensor. Instead of a monocular camera, we will use a stereo camera to increase the precision in the estimation of the 3D point position.

In the following, we will shortly summarise Section 5.2.1 and 5.2.2 because state and process model for SLAM with a stereo camera are the same as for SLAM with a single camera. For a more detailed explanation have a look at these sections.

The first component of the state $\mathbf{x}$ is the position $\mathbf{t}^{\mathsf{W}} = (t_x, t_y, t_z)^\top$ of the camera within the world. In Section 5.2.1 this information refers to the camera projection centre. If we use a stereo camera, we have two camera systems and therefore two projection centres. We define, that $\mathbf{t}^{\mathsf{W}}$ indicates the position of the left one.

All the other state components of the system with the single camera sensor, like orientation, linear and angular velocity and the position of the features, remain the same as for the monocular camera.

Therefore, as state we have

$$\mathbf{x} = \begin{bmatrix} \mathbf{t}^{\mathsf{W}} \\ \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} \\ \mathbf{y}_1^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_i^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_n^{\mathsf{W}} \end{bmatrix} = \begin{bmatrix} \text{Position of the camera} \\ \text{Orientation of the camera} \\ \text{Linear velocity} \\ \text{Angular velocity} \\ 1^{st} \text{ feature} \\ \vdots \\ i^{th} \text{ feature} \\ \vdots \\ n^{th} \text{ feature} \end{bmatrix}.$$

If the dimension of the state is m, $\mathbf{P}$ is the according $m \times m$ error covariance matrix.

As well as the state, also the process, namely the motion of the camera, stays the same. In the following, subscript *new* refers to estimates for the new point in time. Values of the previous point in time have no subscript. As a process model we have

$$\mathbf{x}_{new} = f(\mathbf{x}, \mathbf{w}) = \begin{bmatrix} \mathbf{t}_{new}^{\mathsf{W}} \\ \mathbf{q}_{new}^{\mathsf{CW}} \\ \mathbf{v}_{new}^{\mathsf{W}} \\ \omega_{new}^{\mathsf{CW}} \\ \mathbf{y}_{1,new}^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_{i,new}^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_{n,new}^{\mathsf{W}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{\mathsf{W}} + (\mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}}) \triangle k \\ \mathbf{q}((\omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}}) \triangle k) \times \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}} \\ \mathbf{y}_1^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_i^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_n^{\mathsf{W}} \end{bmatrix}$$

$\mathbf{V}^{\mathsf{W}}$ and $\mathbf{\Omega}^{\mathsf{CW}}$ represent the process noise as some unknown random acceleration. They are summarised in the noise vector $\mathbf{w} = (\mathbf{V}^{\mathsf{W}}, \mathbf{\Omega}^{\mathsf{CW}})^\top$. The values for the noise vector are approximated by zero which is the mean of the corresponding distribution. Then, we have

$$\hat{\mathbf{x}}_{new} = f(\hat{\mathbf{x}}, 0) = \begin{bmatrix} \hat{\mathbf{t}}_{new}^{\mathsf{W}} \\ \hat{\mathbf{q}}_{new}^{\mathsf{CW}} \\ \hat{\mathbf{v}}_{new}^{\mathsf{W}} \\ \hat{\omega}_{new}^{\mathsf{CW}} \\ \hat{\mathbf{y}}_{1,new}^{\mathsf{W}} \\ \vdots \\ \hat{\mathbf{y}}_{i,new}^{\mathsf{W}} \\ \vdots \\ \hat{\mathbf{y}}_{n,new}^{\mathsf{W}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{r}}^{\mathsf{W}} + \hat{\mathbf{v}}^{\mathsf{W}} \triangle k \\ \hat{\mathbf{q}}(\hat{\omega}^{\mathsf{CW}} \triangle k) \times \hat{\mathbf{q}}^{\mathsf{CW}} \\ \hat{\mathbf{v}}^{\mathsf{W}} \\ \hat{\omega}^{\mathsf{CW}} \\ \hat{\mathbf{y}}_1^{\mathsf{W}} \\ \vdots \\ \hat{\mathbf{y}}_i^{\mathsf{W}} \\ \vdots \\ \hat{\mathbf{y}}_n^{\mathsf{W}} \end{bmatrix}$$

As this is the same nonlinear process model as in Section 5.2.2, we also need to derive the Jacobian matrices $\mathbf{A}$ and $\mathbf{W}$ to apply the Kalman Filter approach to this nonlinear system. Both matrices are given in detail in Appendix A. The matrix $\mathbf{A}$ contains the partial derivatives of the process model $f(\mathbf{x}, 0)$ with

respect to the state $\mathbf{x}$. It is used to project the uncertainty represented by the error covariance matrix $\mathbf{P}$ about one estimate to the next point in time. The Jacobian matrix $\mathbf{W}$ contains the partial derivatives of $f(\mathbf{x}, 0)$ with respect to the noise vector $\mathbf{v}$.

Thus, after we have predicted the state estimate $\hat{\mathbf{x}}_{new}$ for the next point in time by calculating $f(\mathbf{x}, 0)$, we can derive the uncertainty about this estimate represented by $\mathbf{P}_{new}$ with

$$\mathbf{P}_{new} = \mathbf{A}\mathbf{P}\mathbf{A}^\top + \mathbf{W}\mathbf{Q}\mathbf{W}^\top.$$

5.3.2 Output of the System

Opposed to a monocular camera, a stereo camera provides not just one 2D projection of a 3D feature per time step. We have two camera systems and therefore a left and right projection of one landmark is available at considered points in time. With this information we can directly compute the according camera coordinates by applying Equations (3.8) to (3.10) if we assume that the camera is in standard stereo geometry.

Depending on the position and orientation of the camera just some features might be visible for both camera systems. These features are referred to as *fully visible* in contrast to features where just one or no projection is visible. Thus, the output of the system depends on its state.

If l features are fully visible, the output of the system consists of l pairs of image coordinates $\mathbf{x}_l^{\mathsf{l}} = (x_l^{\mathsf{l}}, y_l^{\mathsf{l}})^\top$ and $\mathbf{x}_r^{\mathsf{l}} = (x_r^{\mathsf{l}}, y_r^{\mathsf{l}})^\top$ which are the left and right projection of the visible landmarks. In our case we just measure the image coordinates of the left projection and the disparity. Remember that the disparity is the difference between the left and right x-coordinate! Because we assumed that the stereo image is rectified, the y-coordinate of both projections is the same. So, one projection and the disparity contain all necessary information to derive the 3D position of a feature point.

One might ask, if it would not be easier to measure the camera coordinates of the feature directly. Especially, the differentiation of the measurement model would be less complicated due to the omitted division as you will see in the following section. Choosing the disparity instead has the advantage that its value indirectly influences the uncertainty about the corresponding camera coordinates. Small disparity values refer to features that are far away; large values indicate a close feature position. Landmarks that are far away from the camera can be less precisely measured than nearer ones. If a measurement value of a feature with a large depth is accompanied by a certain measurement error, the error in the calculated 3D position would be larger than for a nearer feature and the same measurement error. This situation is depicted by Figure 5.4.

This circumstance is not exploited if we measure the camera coordinates of the landmark directly. The uncertainty about the 3D position would be the same, whether it is far away or near to the camera.

To summarise, if l features are fully visible the measurement vector $\mathbf{z}$ consists of l three-dimensional vectors $\mathbf{z}_i$.

$$\mathbf{z}_i = \begin{bmatrix} x_{l,i}^{\mathsf{l}} \\ y_{l,i}^{\mathsf{l}} \\ d_i^{\mathsf{l}} \end{bmatrix} = \begin{bmatrix} \text{x-coordinate of the left projection} \\ \text{y-coordinate of the left/right projection} \\ \text{Disparity} \end{bmatrix}$$

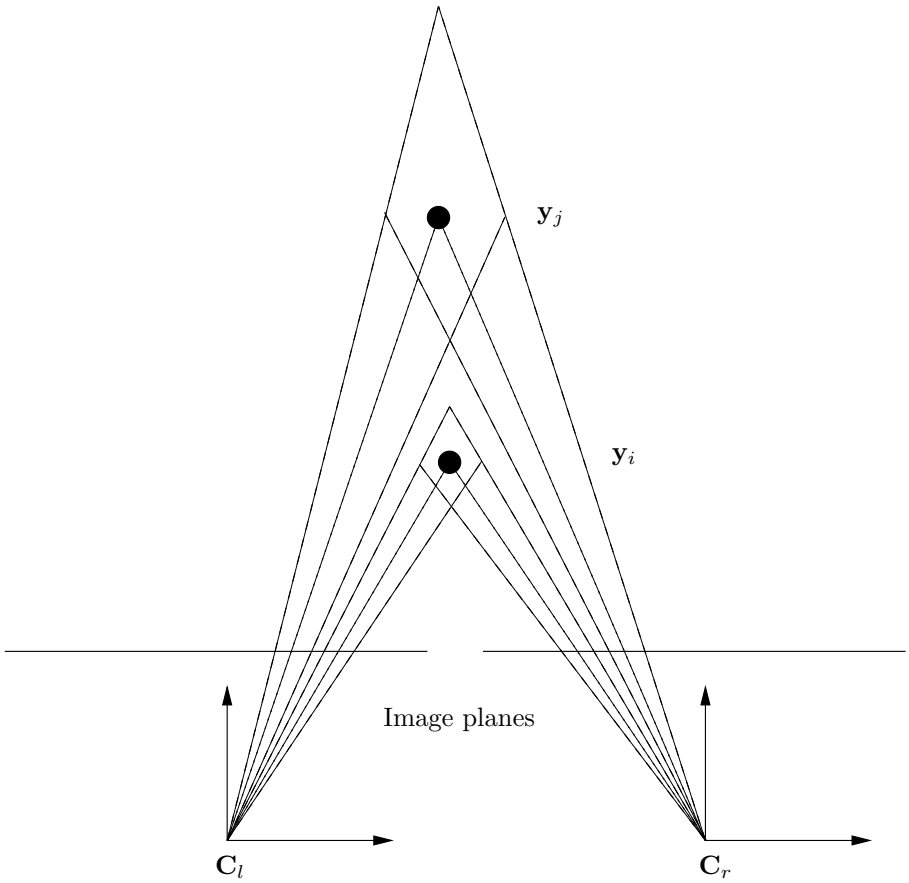

Figure 5.4: *Regions of uncertainty for two point features* $\mathbf{y}_i$ *and* $\mathbf{y}_j$ *in 2D. Trapezoids around the 2D points bound the region of uncertainty referring to the position of the considered features. Dotted lines indicate the line of sight from the camera projection centres* $\mathbf{C}_l$ *or* $\mathbf{C}_r$ *to each feature point. Dashed lines can be taken as an error bound and are defined by the deviation of the measurement error. Farer feature points with a small disparity are accompanied by a larger uncertainty about their position. Projections of nearer feature points with a larger disparity provide more information about their 2D position. This circumstance can be adopted to 3D without restrictions.*

Each row and column of the according error covariance matrix $\mathbf{S}$ refers to one component of the measurement vector. $\mathbf{S}$ is therefore of dimension $3l \times 3l$. It represents the uncertainty about a predicted measurement.

5.3.3 Measurement Model

The measurement model relates the state of the system to the measurement of its output. The state contains information about the position and orientation of the camera in the world, its average velocity during one time step and the position of the landmarks in the world. The measurement vector contains the image coordinates of the projections of the fully visible feature points onto the left image plane of the stereo camera and the appropriate disparity.

The relation between the left projections and the state of the camera is clearly the same as the relation between the single projections and the state of the monocular camera in Section 5.3.3. Firstly, the world coordinates of the landmarks $\mathbf{y}^{\mathsf{W}}$ need to be transformed into camera coordinates $\mathbf{y}_l^{\mathsf{C}}$ for the left camera system. This transformation consists of a rotation and a translation based on the appropriate values derived from the state: $\mathbf{t}^{\mathsf{W}}$ as the position and $\mathbf{q}^{\mathsf{CW}}$ as the orientation of the camera. Regarding to one feature point $\mathbf{y}_i^{\mathsf{W}}$, we have

$$\mathbf{y}_{i,l}^{\mathsf{C}} = \begin{bmatrix} x_{i,l}^{\mathsf{C}} \\ y_{i,l}^{\mathsf{C}} \\ z_{i,l}^{\mathsf{C}} \end{bmatrix} = \mathbf{R}^{\mathsf{CW}}(\mathbf{y}_i^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}}). \tag{5.7}$$

The matrix $\mathbf{R}^{\mathsf{CW}}$ denotes the rotation matrix derived from the quaternion $\mathbf{q}^{\mathsf{CW}}$ and is introduced here to simplify the notation. Equation (5.7) is the same as Equation (5.4).

Secondly, we need to project the camera point $\mathbf{y}_{i,l}^{\mathsf{C}}$ onto the left image plane. This is easily derived by a division.

$$\hat{\mathbf{y}}_{i,l}^{\mathsf{I}} = \begin{bmatrix} x_{i,l}^{\mathsf{I}} \\ y_{i,l}^{\mathsf{I}} \end{bmatrix} = \begin{bmatrix} \frac{x_{i,l}^{\mathsf{C}}}{z_{i,l}^{\mathsf{C}}} \\ \frac{y_{i,l}^{\mathsf{C}}}{z_{i,l}^{\mathsf{C}}} \end{bmatrix} \tag{5.8}$$

Like already explained for Equation (5.5), $\hat{\mathbf{y}}_{i,l}^{\mathsf{I}}$ is just an estimate of the real projection of the landmark $\mathbf{y}_i^{\mathsf{W}}$. What the camera provides are rather pixel than image coordinates. The extraction of image coordinates out of a pixel location is not unique. Depending on the solution of the camera, a certain number of image points are merged to one pixel. Thus, we need to introduce a random noise variable to model the unknown displacement of the image coordinates in the horizontal and vertical direction. We define $\mathbf{v}_l = (v_{l,x}, v_{l,y})^\top$ as the random measurement noise vector. It is assumed to be normally distributed with zero mean and a certain covariance matrix $\mathbf{R_l}$.

$$p(\mathbf{v}_l) \sim N(0, \mathbf{R_l})$$

The covariance matrix is of the following form

$$\mathbf{R}_l = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

By setting the covariance values, situated outside the main diagonal, to zero we assume that the noise is uncorrelated in x− and y−direction.

If we augment Equation (5.8) with this noise vector $\mathbf{v}_l$ we derive the real measurement $\mathbf{y}^{\mathsf{I}}_{i,l}$ for the landmark $\mathbf{y}^{\mathsf{W}}_i$.

$$\mathbf{y}^{\mathsf{I}}_{i,l} = \begin{bmatrix} x^{\mathsf{I}}_{i,l} \\ y^{\mathsf{I}}_{i,l} \end{bmatrix} = \begin{bmatrix} \frac{x^{\mathsf{C}}_{i,l}}{z^{\mathsf{C}}_{i,l}} + v_{l,x} \\ \frac{y^{\mathsf{C}}_{i,l}}{z^{\mathsf{C}}_{i,l}} + v_{l,y} \end{bmatrix} \tag{5.9}$$

$v_{l,x}$ and $v_{l,y}$ are assumed to be zero. Thus, to predict the left projections of the fully visible features, we use Equation (5.8).

Besides the left projection $\hat{\mathbf{y}}^{\mathsf{I}}_{i,l}$ of one landmark, the third component of the specific measurement $\mathbf{z}_i$ is the disparity d^{I}_i. It is defined as the difference between the x−coordinates of the left and right projection of the considered landmark. Thus, to be able to calculate the disparity we also need to define the relation between the state of the system and the right projections of the fully visible landmarks. In principle, this relation resembles the one for the left projection. Firstly, we need to transform the world coordinates of the appropriate landmarks into camera coordinates for the right camera system. Secondly, we project the computed camera point onto the image plane.

The difference between the left and right camera influences the first transformation. Both camera systems are orientated the same way. Thus, the quaternion $\mathbf{q}^{\mathsf{CW}}$ is also used to rotate world into right camera coordinates. The position $\mathbf{t}^{\mathsf{W}}$ is referred to the left camera projection centre which is unequal to the right one. But, to transform world into right camera coordinates we need to obtain the world coordinates $\mathbf{t}^{\mathsf{W}}_r$ of the right camera projection centre. We know the distance, namely the baseline $\mathbf{b}$, between both projection centres. This baseline is rotated around the left camera projection centre about $\mathbf{q}^{\mathsf{WC}}$ which denotes the inverse rotation to $\mathbf{q}^{\mathsf{CW}}$. Therefore, the world coordinates of the right projection centre are computed by

$$\mathbf{t}^{\mathsf{W}}_r = \mathbf{t}^{\mathsf{W}} + \mathbf{R}^{\mathsf{WC}}\mathbf{b}$$

To transform a specific landmark $\mathbf{y}^{\mathsf{W}}_i$ into right camera coordinates $\mathbf{y}^{\mathsf{C}}_{i,r}$, we substitute $\mathbf{t}^{\mathsf{W}}_r$ for $\mathbf{t}^{\mathsf{W}}$ in Equation (5.7) and derive

$$\begin{aligned} \mathbf{y}^{\mathsf{C}}_{i,r} = \begin{bmatrix} x^{\mathsf{C}}_{i,r} \\ y^{\mathsf{C}}_{i,r} \\ z^{\mathsf{C}}_{i,r} \end{bmatrix} &= \mathbf{R}^{\mathsf{CW}}(\mathbf{y}^{\mathsf{W}}_i - \mathbf{t}^{\mathsf{W}}_r) \\ &= \mathbf{R}^{\mathsf{CW}}(\mathbf{y}^{\mathsf{W}}_i - (\mathbf{t}^{\mathsf{W}} + \mathbf{R}^{\mathsf{WC}}\mathbf{b})). \end{aligned}$$

The second transformation, the projection of a right camera point onto the image plane, stays the same as for a left camera point in Equation (5.8).

$$\hat{\mathbf{y}}^{\mathsf{I}}_{i,r} = \begin{bmatrix} x^{\mathsf{I}}_{i,r} \\ y^{\mathsf{I}}_{i,r} \end{bmatrix} = \begin{bmatrix} \frac{x^{\mathsf{C}}_{i,r}}{z^{\mathsf{C}}_{i,r}} \\ \frac{y^{\mathsf{C}}_{i,r}}{z^{\mathsf{C}}_{i,r}} \end{bmatrix} \tag{5.10}$$

Here, $\hat{\mathbf{y}}^{\mathsf{I}}_{i,r}$ is also an estimate of the real measurement $\mathbf{y}^{\mathsf{I}}_{i,r}$. The discrepancy is due to an unknown noise term $\mathbf{v}_r = (v_{r,x}, v_{r,y})^\top$ analogous to $\mathbf{v}_l$ for the

projection on the left image plane. $\mathbf{v}_r$ is likely to differ from $\mathbf{v}_l$ at each point in time, but the covariance matrices are the same.

$$p(\mathbf{v}_r) \sim N(0, \mathbf{R}_r) \sim N(0, \mathbf{R}_l)$$

As usual, we assume the noise to be equal to the mean of its distribution. Therefore, to predict the right projections of the fully visible landmarks we compute Equation (5.10).

Besides the left projection, the specific measurement vectors $\mathbf{z}_i$ for a fully visible landmark $\mathbf{y}_i^{\mathsf{W}}$ contains the disparity d_i^{I} rather than the image coordinates for the right projection. d_i^{I} is calculated by

$$d_i^{\mathsf{I}} = x_{i,l}^{\mathsf{I}} - x_{i,r}^{\mathsf{I}} \tag{5.11}$$

where $x_{i,l}^{\mathsf{I}}$ and $x_{i,r}^{\mathsf{I}}$ is calculated by Equation (5.8) or (5.10) respectively. Two estimated values are used to compute the disparity. Therefore, the disparity itself is an estimate. The difference between the real disparity and its estimate is covered by a random noise value v_d which is the sum of the noise values for each x-coordinate. We do not know the individual values for $v_{l,x}$ and $v_{r,x}$ at each point in time and therefore also not for v_d. Because we assumed $v_{l,x}$ and $v_{r,x}$ to be zero, v_d results to zero. Thus, the noise in the disparity value is also normally distributed with zero mean. Its standard deviation is the sum of the standard deviations for $v_{l,x}$ and $v_{r,x}$: $2\sigma_x$.

$$p(v_d) \sim N(0, 2\sigma_x)$$

We have a three-dimensional measurement vector $\mathbf{z}_i$ for each fully visible landmark $\mathbf{y}_i^{\mathsf{W}}$ containing the appropriate left projection and disparity. Each component is accompanied by some noise. In all cases we do not know the individual noise values and model them as normally distributed white noise with zero mean and a specific standard deviation. We now summarise these noise values into the measurement noise vector $\mathbf{v} = (v_{l,x}, v_{l,x}, v_d)^\top$. It can be altogether modelled as normally distributed white noise with zero mean and covariance matrix $\mathbf{R}$.

$$p(\mathbf{v}) \sim N(0, \mathbf{R}).$$

The covariance matrix has the following form:

$$\mathbf{R} = \begin{bmatrix} \sigma_x & 0 & 0 \\ 0 & \sigma_y & 0 \\ 0 & 0 & 2\sigma_x \end{bmatrix}$$

To summarise, we calculate $\mathbf{z}_i$ regarding to one world point $\mathbf{y}_i^{\mathsf{W}}$ by:

$$\mathbf{z}_i = h_i(\mathbf{x}, \mathbf{v}) = \begin{bmatrix} x_{l,i}^{\mathsf{I}} \\ y_{l,i}^{\mathsf{I}} \\ d_i^{\mathsf{I}} \end{bmatrix} = \begin{bmatrix} \frac{x_{i,l}^{\mathsf{C}}}{z_{i,l}^{\mathsf{C}}} + v_{1,x} \\ \frac{y_{i,l}^{\mathsf{C}}}{z_{i,l}^{\mathsf{C}}} + v_{1,y} \\ x_{i,l}^{\mathsf{I}} - x_{i,r}^{\mathsf{I}} + v_d \end{bmatrix}$$

where $\mathbf{y}_{i,l}^{\mathsf{C}}$ is calculated by Equation (5.7) and $x_{i,r}^{\mathsf{I}}$ by Equation (5.10).

The individual values for $\mathbf{v}$ are unknown and $\mathbf{z}_i$ is therefore estimated without it as $\hat{\mathbf{z}}_i$ by calculating:

$$\hat{\mathbf{z}}_i = h_i(\mathbf{x}, \mathbf{v} = 0) = \begin{bmatrix} x^{\mathsf{I}}_{l,i} \\ y^{\mathsf{I}}_{l,i} \\ d^{\mathsf{I}}_i \end{bmatrix} = \begin{bmatrix} \frac{x^{\mathsf{C}}_{i,l}}{z^{\mathsf{C}}_{i,l}} \\ \frac{y^{\mathsf{C}}_{i,l}}{z^{\mathsf{C}}_{i,l}} \\ x^{\mathsf{I}}_{i,l} - x^{\mathsf{I}}_{i,r} \end{bmatrix}$$

where $\mathbf{y}^{\mathsf{C}}_{i,l}$ is again calculated by Equation (5.9) and $x^{\mathsf{I}}_{i,r}$ by Equation (5.10).

As already stated in the previous section, if l features are fully visible, our measurement vector $\mathbf{z}$ consists of l three-dimensional vectors $\mathbf{z}_i$ for each fully visible landmark $\mathbf{y}^{\mathsf{W}}_i$. Therefore, the whole measurement model can be summarised as

$$\mathbf{z} = h(\mathbf{x}, \mathbf{v}) = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_i \\ \vdots \\ \mathbf{z}_n \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}, \mathbf{v}) \\ \vdots \\ h_i(\mathbf{x}, \mathbf{v}) \\ \vdots \\ h_n(\mathbf{x}, \mathbf{v}) \end{bmatrix}.$$

Setting the noise vector to zero, we derive an estimate $\hat{\mathbf{z}}$ for the measurement vector.

$$\hat{\mathbf{z}} = h(\mathbf{x}, 0) = \begin{bmatrix} \hat{\mathbf{z}}_1 \\ \vdots \\ \hat{\mathbf{z}}_i \\ \vdots \\ \hat{\mathbf{z}}_n \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}, 0) \\ \vdots \\ h_i(\mathbf{x}, 0) \\ \vdots \\ h_n(\mathbf{x}, 0) \end{bmatrix}.$$

This estimate is accompanied by an uncertainty about it represented by the covariance matrix $\mathbf{S}$. Until now, we always assumed that the real state vector $\mathbf{x}$ is used to calculate or predict the measurement. We do not have access to this vector and therefore, take an estimate $\hat{\mathbf{x}}$ instead. The uncertainty about this estimate is represented by the covariance matrix $\mathbf{P}$. If we take the state estimate to predict the measurement, the uncertainty about it will contribute to the uncertainty about the estimated measurement. Due to the rotation and projection, the measurement model is nonlinear. To project $\mathbf{P}$ into the measurement space, we need the Jacobian matrix $\mathbf{H}$ which contains the partial derivatives of $h(\mathbf{x}, 0)$ with respect to the state $\mathbf{x}$. But not just the uncertainty about the state estimate will contribute to the uncertainty about the predicted measurement. Additionally we have the measurement noise. We project the noise covariance matrix $\mathbf{R}$ into the measurement space by using the Jacobian matrix $\mathbf{V}$ which contains the partial derivatives of the measurement model $h(\mathbf{x}, 0)$ with respect to the noise vector $\mathbf{v}$.

After we have obtained the predicted the measurement vector $\hat{\mathbf{z}}$ for the specific state estimate $\hat{\mathbf{x}}$ by calculating $h(\hat{\mathbf{x}}, 0)$ we can calculate the uncertainty about it by

$$\mathbf{S} = \mathbf{HPH}^\top + \mathbf{VRV}^\top$$

5.4 Predict and Correct Steps

In the previous sections we have presented the appropriate models to apply the EKF algorithm to the SLAM problem by either using a single camera or a stereo camera. The EKF algorithm itself stays the same regardless of which vision sensor is used.

As already said above, we assume that the system is already initialised with values for the position and orientation of the robot as well as with positions of n landmarks in the world. There is no uncertainty about the robot's but about the landmark's position. This means, that the components of the main diagonal of the error covariance matrix $\mathbf{P}$ referring to the feature points are set to some initial values. The values outside the main diagonal are equal to zero.

The EKF is cycling through two steps: the predict and the correct step. As the name implies, in the predict step a prediction is made for the state of the system in the next point in time. In the correct step, this estimate is corrected by including measurements of the output of the system.

5.4.1 Predict Step

Assume, that we have given a state estimate $\hat{\mathbf{x}}_{k-\triangle k}$ and the according error covariance matrix $\mathbf{P}_{k-\triangle k}$ at the point $(k - \triangle k)$ in time. Then, we can predict the state $\hat{\mathbf{x}}_k^-$ for k by calculating

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-\triangle k}, 0).$$

The uncertainty given in $\mathbf{P}_k^-$ about the *a priori* estimate $\hat{\mathbf{x}}_k^-$ also has to be calculated. Therefore, the Jacobians $\mathbf{A}$ and $\mathbf{W}$ containing the partial derivatives of f with respect to the state $\mathbf{x}$ or the process noise vector $\mathbf{w}$ respectively, need to be calculated for the current time. Then, we can compute

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-\triangle k}\mathbf{A}^\top + \mathbf{W}\mathbf{Q}\mathbf{W}^\top$$

The uncertainty is projected to the next point in time indicated by the term $\mathbf{A}\hat{\mathbf{P}}_{k-\triangle k}\mathbf{A}^\top$, and also increased by the process noise $\mathbf{W}\mathbf{Q}\mathbf{W}^\top$.

5.4.2 Correct Step

In contrast to the state of the system, we have access to the real measurements of the system's output. In the correct step, the difference between the predicted and the real measurement, the residual, is weighted with the Kalman Gain $\mathbf{K}$ and added to the *a priori* estimate $\hat{\mathbf{x}}_k^-$. Thus, we first need to compute the Kalman Gain. For this task the Jacobians $\mathbf{H}$ and $\mathbf{V}$ of the measurement model h with respect to the state $\mathbf{x}$ or the measurement noise vector $\mathbf{v}$ respectively are needed. If they had been computed, the Kalman Gain can be derived by

$$\mathbf{K}_k = \frac{\mathbf{P}_k^-\mathbf{H}^\top}{(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{V}\mathbf{R}\mathbf{V}^\top)}.$$

Then, we predict the measurement $\hat{\mathbf{z}}$ which depends on the *a priori* state estimate

$$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_k, 0).$$

After the real measurement $\mathbf{z}_k$ had been obtained, the residual can be calculated and weighted. The last step is to add this to the *a priori* state estimate.

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-, 0)).$$

Besides the state, the uncertainty about the predicted state is corrected as well by computing

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-.$$

Intuitively said, if a concrete measurement is involved in the correct step, the uncertainty about the state estimate $\hat{\mathbf{x}}_\mathbf{k}$ will shrink. But as you might remember from the simple example in Section 4.2.4, there are cases, where the uncertainty increases. It occurs when estimates are made based on measurements that are corrupted by a high measurement noise.

Chapter 6

An Observation Strategy

In the previous chapter we applied the Extended Kalman Filter approach to the SLAM problem. A problem of using the EKF is that it does not scale very well. The complexity is cubic in the number of features in the map. In this chapter, we will examine strategies to reduce the complexity to at least $O(n^2)$ where n is the number of features.

One of these strategies includes that just a single feature instead of all visible ones is measured. In [35] it is shown, that this is sufficient for tracking. If we do so, we need to select the best feature based on a heuristic. In the following, we will refer to this heuristic as an observation strategy. It is adapted from Davison in [9] and [8].

In this chapter, we will firstly concentrate on the ways to reduce the time complexity of one EKF cycle. This examination is chiefly based on [23]. Secondly, an appropriate heuristic is introduced to realise the selection of the best landmark. The two SLAM scenarios, the first with a single camera, the second with a stereo camera, are handled separately.

6.1 Complexity of the Kalman Filter

We will first examine the general time complexity of the Extended Kalman Filter algorithm in detail. Considering each step during one EKF cycle, we will introduce methods to reduce the cubic time complexity to at least $O(n^2)$. Just to remember, the appropriate equations are listed in Figure 6.1.

If we have a look at these equations, we can state that there are two major time consuming operations: matrix multiplication and matrix inversion. If the matrix multiplication is carried out in a straightforward manner, its time complexity is $O(n^3)$ if we multiply $n \times n$ matrices. Matrix inversion also grows cubic with the number of visible and measured features.

In the case of the EKF, the maximal size of a matrix, here $\mathbf{P}$, is $(13 + 3n) \times (13 + 3n)$ where n is the number of features. The matrix which will be inverted is the innovation covariance. It is of dimension $(2l \times 2l)$ or $(3l \times 3l)$.[1] l denotes the number of visible and measurable features. Because the number of

[1] The dimension of the measurements using a monocular camera is 2. If a stereo camera is used as a vision sensor, the measurement is three-dimensional

1. Predict Step
   (a) Predict the state ahead.
   $$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, 0)$$
   (b) Predict the error covariance matrix ahead.
   $$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{W}\mathbf{Q}\mathbf{W}^\top$$
2. Correct Step
   (a) Calculate the Kalman Gain.
   $$\mathbf{K}_k = \frac{\mathbf{P}_k^-\mathbf{H}^\top}{(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{V}\mathbf{R}\mathbf{V}^\top)}$$
   (b) Correct the *a priori* state estimate
   $$\hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-, 0))$$
   (c) Correct the *a posteriori* error covariance matrix estimate
   $$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$$

Figure 6.1: *Equations of one Extended Kalman Filter Cycle.*

measurable features cannot be larger than the number of known features, the overall complexity of the EKF is $O(13 + 3n) = O(n^3)$.

We can reduce this complexity to $O(n^2)$ by considering aspects related to the SLAM problem. First of all, the process model just affects the state of the camera and the velocities, summarised in $\mathbf{x}_v$. The known features are not involved and thus not the whole state of the system.

Secondly, usually just a small subset of feature points can be measured at each point in time, due to the constraints of the viewing direction. In the following, we will explain this in detail, first for the predict step and after that for the correct step.

6.1.1 Complexity of the Predict Step

In the predict step of the Kalman Filter, we predict the state $\mathbf{x}$ of the system as $\hat{\mathbf{x}}^-$ and the related error covariance $\mathbf{P}$ as $\mathbf{P}^-$. The process model f relates the state at one point in time to the next. But, as already mentioned above, just the state of the camera and its velocities are affected. Thus,the Jacobian matrix $\mathbf{A}$, containing the partial derivatives of the process model with respect to the state, is of the following form,

$$\mathbf{A} = \begin{bmatrix} \frac{\partial f_v}{\partial \mathbf{x}_v} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

where f_v is the first part of the measurement model.

$$\mathbf{x}_{v,new} = f(\mathbf{x}_v, \mathbf{w} = 0) = \begin{bmatrix} \mathbf{t}^{\mathsf{W}}_{new} \\ \mathbf{q}^{\mathsf{CW}}_{new} \\ \mathbf{v}^{\mathsf{W}}_{new} \\ \omega^{\mathsf{CW}}_{new} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{\mathsf{W}} + \mathbf{v}^{\mathsf{W}} \triangle k \\ \mathbf{q}(\omega^{\mathsf{CW}} \triangle k) \times \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} \end{bmatrix}$$

The detailed Jacobian matrix $\mathbf{A}$ can be found in Appendix A.

The overall dimension of the state is $m = 13 + 3n$ where n is the number of the 3D landmarks and 13 is the dimension of $\mathbf{x}_v$. Thus, $\mathbf{A}$ is a $m \times m$ Jacobian matrix as well as the error covariance matrix $\mathbf{P}$. The block $\frac{\partial f_v}{\partial \mathbf{x}_v}$ is of dimension 13×13.

Let's consider the first summand $\mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top$ of the prediction of the error covariance matrix as $\mathbf{P}_k^-$ and let the old $\mathbf{P}_{k-1}$ be denoted by

$$\mathbf{P}_{k-1} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{bmatrix}.$$

$\mathbf{P}_{11}$ is a covariance matrix also of dimension 13×13 related to $\mathbf{x}_v$. $\mathbf{P}_{12}$ and $\mathbf{P}_{21}$ are of dimension $13 \times 3n$ and $3n \times 13$, respectively. [2] $\mathbf{P_{22}}$ is then a $3n \times 3n$ covariance matrix.

If we perform the matrix operation for $\mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top$ explicitly, we obtain:

$$\begin{aligned} \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top &= \begin{bmatrix} \frac{\partial f_v}{\partial \mathbf{x}_v} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{P_{11}} & \mathbf{P_{12}} \\ \mathbf{P_{21}} & \mathbf{P_{22}} \end{bmatrix} \begin{bmatrix} (\frac{\partial f_v}{\partial \mathbf{x}_v})^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial f_v}{\partial \mathbf{x}_v}\mathbf{P}_{11}(\frac{\partial f_v}{\partial \mathbf{x}_v})^\top & \frac{\partial f_v}{\partial \mathbf{x}_v}\mathbf{P}_{12} \\ (\frac{\partial f_v}{\partial \mathbf{x}_v}\mathbf{P}_{12})^\top & \mathbf{P}_{22} \end{bmatrix} \end{aligned}$$

[2] Note that $\mathbf{P}_{12}$ is the transpose of $\mathbf{P}_{21}$ because of the symmetry of covariances.

Regarding to the dimensions of the matrices, the term $\frac{\partial f_v}{\partial \mathbf{x}_v}\mathbf{P}_{11}(\frac{\partial f_v}{\partial \mathbf{x}_v})^\top$ can be evaluated by $2(13*13*13)$ multiplications. To solve $\frac{\partial f_v}{\partial \mathbf{x}_v}\mathbf{P}_{12}$ we need $13*13*3n$ multiplications. $(\frac{\partial f_v}{\partial \mathbf{x}_v}\mathbf{P}_{12})^\top$ is just the transpose of the previous term and do not need to be evaluated again. Altogether, the whole amount of multiplications to evaluate $\mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top$ lies at $2(13*13*13)+(13*13*3n)$.

The second summand $\mathbf{W}\mathbf{Q}\mathbf{W}^\top$ of the prediction function can be considered equivalently. The Jacobian matrix $\mathbf{W}$ contains the partial derivatives of the process model with respect to the process noise. It is of the following form:

$$\mathbf{W} = \begin{bmatrix} \frac{\partial f_v}{\partial \mathbf{V}^{\mathsf{W}}} & \frac{\partial f_v}{\partial \mathbf{\Omega}^{\mathsf{CW}}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

For the detailed matrix, have a look at Appendix A.

Since the process noise vector $\mathbf{w}$ is of dimension 6, $\mathbf{W}$ is a $m \times 6$ matrix. The blocks $\frac{\partial f_v}{\partial \mathbf{V}^{\mathsf{W}}}$ as well as $\frac{\partial f_v}{\partial \mathbf{\Omega}^{\mathsf{CW}}}$ carry 13×3 elements. The process noise does not affect the coordinates of the known features. Thus, the according elements of $\mathbf{W}$ are equal to zero.

The process noise covariance $\mathbf{Q}$ can be denoted by:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{22} \end{bmatrix}$$

It is a 6×6 matrix and the blocks $\mathbf{Q}_{11}$ and $\mathbf{Q}_{22}$ are each of dimension 3×3.

If we perform the matrix multiplication $\mathbf{W}\mathbf{Q}\mathbf{W}^\top$ explicitly, we derive:

$$\begin{aligned} \mathbf{W}\mathbf{Q}\mathbf{W}^\top &= \begin{bmatrix} \frac{\partial f_v}{\partial \mathbf{V}^{\mathsf{W}}} & \frac{\partial f_v}{\partial \mathbf{\Omega}^{\mathsf{CW}}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{22} \end{bmatrix} \begin{bmatrix} (\frac{\partial f_v}{\partial \mathbf{V}^{\mathsf{W}}})^\top & \mathbf{0} \\ (\frac{\partial f_v}{\partial \mathbf{\Omega}^{\mathsf{CW}}})^\top & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial f_v}{\partial \mathbf{V}^{\mathsf{W}}}\mathbf{Q}_{11}(\frac{\partial f_v}{\partial \mathbf{V}^{\mathsf{W}}})^\top + \frac{\partial f_v}{\partial \mathbf{\Omega}^{\mathsf{W}}}\mathbf{Q}_{22}(\frac{\partial f_v}{\partial \mathbf{\Omega}^{\mathsf{W}}})^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \end{aligned}$$

Because no block of a size related to the n known features is involved in the one block unequal to the zero matrix, the number of multiplications is independent of n. We exactly need $4(13*3*3)$ multiplications.

Thus, the cost of the predict step in all is linear in m.

6.1.2 Complexity of the Correct Step

Since just a few features of all known are visible for the camera sensor at each point in time, the Jacobian matrix $\mathbf{H}$ containing all partial derivatives of the measurement model h with respect to the state, carries a large number of zeros. Let's assume that we just measure one feature $\mathbf{y}_i^{\mathsf{W}}$ after each time step. Then $\mathbf{H}$ is of the following form:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial h}{\partial \mathbf{x}_v} & \mathbf{0} & \frac{\partial h}{\partial \mathbf{y}_i^{\mathsf{W}}} & \mathbf{0} \end{bmatrix}$$

The detailed Jacobian matrix can be found in Appendix A.

We know, that the dimension of the state vector $\mathbf{x}$ is $m = 13 + 3n$. The dimension p of the measurement vector is either 2 or 3, depending on whether we use a single or stereo camera. Thus, the whole matrix $\mathbf{H}$ is of dimension $p \times m$. The block $\frac{\partial h}{\partial \mathbf{x}_v}$ carries $p \times 13$ elements whereas $\frac{\partial h}{\partial \mathbf{y}_i^{\mathsf{W}}}$ is of dimension $p \times 3$.

To evaluate the Kalman Gain $\mathbf{K}$, we need to perform the multiplication $\mathbf{P}_k^- \mathbf{H}^\top$. For this case, $\mathbf{P}_k^-$ is represented by:

$$\mathbf{P}_k^- = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_{01} & \mathbf{P}_2 & \mathbf{P}_{02} \end{bmatrix} \tag{6.1}$$

The block $\mathbf{P}_1$ contains $m \times 13$ and the block $\mathbf{P}_2$ $m \times 3$ elements. If we perform this multiplication explicitly, we obtain

$$\mathbf{P}_k^- \mathbf{H}^\top = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_{01} & \mathbf{P}_2 & \mathbf{P}_{02} \end{bmatrix} \begin{bmatrix} \frac{\partial h}{\partial \mathbf{x}_v}^\top \\ \mathbf{0} \\ \frac{\partial h}{\partial \mathbf{y}_i^{\mathsf{W}}}^\top \\ \mathbf{0} \end{bmatrix} = \mathbf{P}_1 \frac{\partial h}{\partial \mathbf{x}_v}^\top + \mathbf{P_2} \frac{\partial \mathbf{h}}{\partial \mathbf{y}_\mathbf{i}^{\mathsf{W}}}^\top .$$

The number of multiplications adds up to $16pm$.

After evaluating $\mathbf{P}_k^- \mathbf{H}^\top$ we need to derive the innovation covariance $\mathbf{S}$. It can be obtained by equation $\mathbf{H}\mathbf{P}_k^- \mathbf{H}^\top + \mathbf{V}\mathbf{R}\mathbf{V}^\top$. We will firstly consider the first summand.

The result for $\mathbf{P}_k^- \mathbf{H}^\top$ is a $m \times p$ matrix and is represented by

$$\mathbf{P}_k^- \mathbf{H}^\top = \begin{bmatrix} \mathbf{P}'_1 \\ \mathbf{P}'_{01} \\ \mathbf{P}'_2 \\ \mathbf{P}'_{02} \end{bmatrix}$$

where the block $\mathbf{P}'_1$ is a $13 \times p$ and $\mathbf{P}'_2$ a $3 \times p$ matrix.

As result for the product $\mathbf{H}\mathbf{P}_k^- \mathbf{H}^\top$ we obtain

$$\mathbf{H}\mathbf{P}_k^- \mathbf{H}^\top = \begin{bmatrix} \frac{\partial h}{\partial \mathbf{x}_v} & \mathbf{0} & \frac{\partial h}{\partial \mathbf{y}_i^{\mathsf{W}}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P}'_1 \\ \mathbf{P}'_{01} \\ \mathbf{P}'_2 \\ \mathbf{P}'_{02} \end{bmatrix} = \frac{\partial h}{\partial \mathbf{x}_v} \mathbf{P}'_1 + \frac{\partial h}{\partial \mathbf{y}_i^{\mathsf{W}}} \mathbf{P}'_2$$

The amount of multiplications lies at $16p^2$, where p is either 2 or 3.

The second summand $\mathbf{V}\mathbf{R}\mathbf{V}^\top$ in the equation to derive the innovation covariance can be simplified equivalently. $\mathbf{R}$ is the measurement error covariance of dimension $p \times p$. The Jacobian matrix $\mathbf{V}$ contains the partial derivatives of the measurement model with respect to the measurement noise. Because the measurement noise vector is an additive constant in both SLAM scenarios whether with a single or stereo camera, $\mathbf{V}$ is an identity matrix regardless of the value of p. We have

$$\mathbf{V}\mathbf{R}\mathbf{V}^\top = \mathbf{R}.$$

The overall amount of multiplications to calculate the innovation covariance is $16p^2$.

To evaluate the Kalman Gain K we need to invert S. As already mentioned above, the complexity of matrix inversion grows cubic with the number of rows or columns, respectively, of the considered quadratic matrix. Here, we have a $p \times p$ matrix to invert. Thus, we need p^3 multiplications.

The whole amount of multiplications to calculate the Kalman Gain is therefore $16pm + 16p^2 + p^3$ which is linear in m.

Until now, the complexity of all equations whether in the predict or correct step were linear in m. The second equation of the correct step updating the

Predict Step	
$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{W}\mathbf{Q}\mathbf{W}^\top$	$O(m) = O(13+3n) = O(n)$
Correct Step	
$\mathbf{K}_k = \frac{\mathbf{P}_k^-\mathbf{H}^\top}{(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{V}\mathbf{R}\mathbf{V}^\top)}$	$O(m) = O(13+3n) = O(n)$
$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$	$O(m^2) = O((13+3n)^2) = O(n^2)$

Table 6.1: *Complexities for the Equations of one Extended Kalman Filter Cycle.*

error covariance $\mathbf{P}$ is responsible for the quadratic complexity. We have to evaluate the summand $\mathbf{K}_k\mathbf{H}\mathbf{P}_k^-$. We will first consider the product $\mathbf{H}\mathbf{P}_k^-$. $\mathbf{H}$ is as already stated above represented by

$$\mathbf{H} = \begin{bmatrix} \frac{\partial h}{\partial \mathbf{x}_v} & \mathbf{0} & \frac{\partial h}{\partial \mathbf{y}_i^{\mathrm{W}}} & \mathbf{0} \end{bmatrix}$$

The predicted error covariance matrix $\mathbf{P}_k^-$ is denoted by

$$\mathbf{P}_k^- = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_{01} \\ \mathbf{P}_2 \\ \mathbf{P}_{02} \end{bmatrix}.$$

Note that these blocks are not the same as in Equation (6.1) although they split up the same matrix $\mathbf{P}_k^-$. Here, $\mathbf{P}_1$ is of dimension $13 \times m$. $\mathbf{P}_2$ carries $3 \times m$ elements. If we evaluate the product, we obtain

$$\mathbf{H}\mathbf{P}_k^- = \begin{bmatrix} \frac{\partial h}{\partial \mathbf{x}_v} & \mathbf{0} & \frac{\partial h}{\partial \mathbf{y}_i^{\mathrm{W}}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_{01} \\ \mathbf{P}_2 \\ \mathbf{P}_{02} \end{bmatrix} = \frac{\partial h}{\partial \mathbf{x}_v}\mathbf{P}_1 + \frac{\partial h}{\partial \mathbf{y}_i^{\mathrm{W}}}\mathbf{P}_2$$

where the result is a $p \times m$ matrix. $16pm$ multiplications are needed. The last step is to multiply the Kalman Gain $\mathbf{K}$ with this result. Either $\mathbf{K}$ which is a $m \times p$ matrix, nor $\mathbf{H}\mathbf{P}_k^-$ carries a zero or identity matrix. Therefore, we derive an $m \times m$ matrix by performing pm^2 multiplications.

Thus, the time complexity of the correct step is $O(m^2)$ or if we just consider the number of known features $O((13+3n)^2) = O(n^2)$. At the same time, this is the time complexity of one EKF cycle. The results presented in Section 6.1 are summarised in Table 6.1.2.

6.2 A Heuristic to Decide which Feature to Track

In the last section we presented methods to reduce the complexity of one EKF cycle by taking the particular structure of the SLAM problem into account.

For one of these methods it is assumed, that we just measure one of the visible feature points per point in time. But if we do so, two questions may arise:

- Is it sufficient for the estimation of the state to measure just one feature?
- Which feature of the several visible ones is best to be measured?

Considering the first question, Welch and Bishop [35] presented the *SCAAT* method where it is shown that measuring a single landmark after each time step is sufficient to observe 3D structure and motion of a scene over time.[3] In the case of 3D-SLAM, a single measurement of a 2D projection of a 3D landmark just provides partial or incomplete information about the whole state of the system, e.g., nothing about the (linear or angular) velocity of the camera and nothing about the depth of the 3D feature position. Systems operating just by obtaining incomplete measurements are referred to as *unobservable* because the whole system's state cannot not be inferred from them. Such systems must incorporate a sufficient set of these measurements to obtain observability. This can be achieved over space or over time. The latter is adopted by the SCAAT technique. It is based on the Extended Kalman Filter where individual measurements providing incomplete information about the system's state are blended into a complete state estimate. The mean for this blending provided by the filter describes the state estimate. Based on several experiments, SCAAT was shown to be accurate, stable, fast and flexible.

To find an answer on the second question, we first need to find a criteria to rate the feature. An intuitive idea is stated by Davison in [9]: The more uncertain we are about the 3D position of a feature, the more profitable it is to measure this one. Or in other words, measurements of features, that are difficult to predict, provide more information about the position of this feature and of the camera than measurements of features which can be reliably predicted.

The innovation covariance $\mathbf{S}$ describes the uncertainty about each predicted measurement. Thus, it contains the basic information to decide which visible feature should be measured at each point in time. It is calculated as follows

$$\mathbf{S} = \mathbf{H}\mathbf{P}\mathbf{H}^\top + \mathbf{V}\mathbf{R}\mathbf{V}^\top \tag{6.2}$$

where $\mathbf{H}$ and $\mathbf{V}$ are the Jacobian matrices of the measurement model $h(\mathbf{x}, 0)$ with respect to the state $\mathbf{x}$ and the measurement noise $\mathbf{v}$, respectively. $\mathbf{P}$ is the error covariance matrix linked to the state and $\mathbf{R}$ is the measurement noise covariance.

$\mathbf{S}$ is a multivariate Gaussian. Therefore, covariance matrices $\mathbf{S}_i$ for each predicted measurement $\hat{\mathbf{z}}_i$ corresponding to a visible feature point $\mathbf{y}_i^{\mathsf{W}}$ can be extracted from it. These smaller covariances refer to a Gaussian with the measurement $\hat{\mathbf{z}}_i$ as its mean. According to Whaite and Ferrie [36], depending on the measurement space, each $\mathbf{S}_i$ can be represented either by an ellipse or ellipsoid centred around the mean of the distribution. They are also referred to as *ellipses* or *ellipsoids of confidence* and represent the amount of uncertainty about the predicted measurement. Or in other words, we can be confident, that

[3] **S**ingle **C**onstraint **A**t **A** **T**ime

the real measurement is situated within the ellipse or ellipsoid. By calculating the surface area or volume of these objects, we can decide which predicted measurement is most uncertain.

Besides its role as a measure of the information content expected of a measurement, $\mathbf{S}_i$ also defines a search region where the according measurement $\hat{\mathbf{z}}_i$ should be located in with high probability. Thus, if we have decided to measure a specific feature, we can send the parameters of the search region to the feature tracker. The advantages of this method are obvious. The feature tracker just needs to search a small region of interest instead of the whole picture. Furthermore, the chances of a mismatch are reduced.

In the previous chapter, we considered two SLAM cases: SLAM with a single camera and SLAM with a stereo camera. In the following sections, the heuristic is discussed in detail with respect to the different vision sensors.

6.2.1 Deriving the Innovation Covariance Matrix for SLAM with a Single Camera

In the case we use a single camera, we predict two dimensional measurements $\hat{\mathbf{y}}_i^{\mathsf{I}}$ for each visible three-dimensional feature $\mathbf{y}_i^{\mathsf{W}}$ referring to its 2D projection onto the image plane. Thus, if l features are visible, $\mathbf{S}$ is a $2l \times 2l$ matrix and l 2×2 covariance matrices $\mathbf{S_i}$ regarding to the visible features can be extracted from it. These covariance matrices represent a two-dimensional standard distribution over image coordinates. Its mean is the predicted measurement $\hat{\mathbf{y}}_i^{\mathsf{I}}$. The distribution can be visualised by an ellipse of confidence on the picture. Its focal point refers to the mean, the direction of its axes are given by the eigenvectors of the covariance matrix and the square root of the according eigenvalues specifies the deviation of the distribution along the axis.

According to [36], the surface area of the ellipse can be used as a measure of uncertainty. If a and b denote the length of the principal axes of the ellipse, the surface area A is calculated by

$$A = \pi ab.$$

The standard deviation of a distribution describes the average deviation of the related Gaussian. The values of the whole distribution diversify much more. Possible realisations of the predicted measurement situated beyond the average deviation are just less probable but should also be involved in the calculation of the amount of uncertainty and in the size of the search region.

Thus, we introduce the factor n_σ and multiply the length of the principle axes of the ellipse with it. Consider the estimated measurement $\hat{\mathbf{y}}_i^{\mathsf{I}}$ with eigenvalues $e_{1,i}$ and $e_{2,i}$ of the according covariance matrix $\mathbf{S}_i$. To derive the surface area of the demanded ellipse, we have to compute

$$A_i = \pi n_\sigma \sqrt{e_{1,i} e_{2,i}}. \tag{6.3}$$

The value for n_σ should extend the standard deviation such that the probability for a measurement to be found within the considered region is approximately 100%. In [9], Davison chose $n_\sigma = 3$. The probability that the possible realisations of a standard deviated random variable lie within the 3σ-region around the mean of the distribution is approximately 99% ([16], p. 1119).

After calculating the amount of uncertainty about the predicted measurement of each visible 3D feature, we can rank them and send the parameters (predicted measurement and corresponding covariance matrix) of the landmark whose measurement is most difficult to predict to the feature tracker. The corresponding covariance matrix specifies the search region for the demanded feature measurement within the image and centred around the estimated measurement.

6.2.2 Deriving the Innovation Covariance Matrix for SLAM with a Stereo Camera

In the second case of SLAM scenarios, we use a stereo camera to measure the visible features. For each, we derive three-dimensional measurement vectors. Thus, if l features are visible, l 3×3 smaller covariances $\mathbf{S}_i$, each referring to one of the predicted measurements for the visible features, can be extracted from the innovation covariance matrix $\mathbf{S}$. As already mentioned for the two-dimensional case, these covariances are related to a standard distribution. Their means are the predicted measurements.

Considering one visible feature point $\mathbf{y}_i^{\mathsf{W}}$, the measurement vector for the SLAM scenario with a stereo camera consists of the image coordinates of the projection of this feature on the left image plane $\mathbf{y}_i^{\mathsf{l}} = (x_l^{\mathsf{l}}, y_l^{\mathsf{l}})^\top$ and the disparity d^{l}. The according innovation covariance matrix $\mathbf{S}_i$ is therefore not defined over one of the image coordinate frames as it was the case wen using a monocular vision sensor. It can be represented as an ellipsoid in the space spanned by x_l^{l}, y_l^{l} and d^{l}. Analogous to the surface area of the ellipses, the volume of the ellipsoids can be seen as a measure for uncertainty. The equation to calculate the volume of an ellipsoid is

$$V = \frac{4}{3}\pi abc.$$

where a, b and c are the lengths of its principal axes. If we substitute the square root of the eigenvalues for a, b and c and introduce the factor n_σ again, we derive the equation to calculate the volume of each $\mathbf{S}_i$:

$$V_i = \frac{4}{3}\pi n_\sigma \sqrt{e_{1,i} e_{2,i} e_{3,i}}$$

After calculating this volume for each ellipsoid, we are able to rank the visible 3D feature points. The corresponding predicted measurement and innovation covariance of the landmark whose measurement is most difficult to predict is sent to the feature tracker. Centred around this prediction, the covariance matrix defines the search region where the real measurement is likely to be found. Note that in this case, the search region is not defined on image coordinates as for the usage of a monocular camera. Thus, the feature tracker needs to project the ellipsoid onto both image planes to define the search region on the pictures.

Chapter 7

Experiments

We will perform a number of experiments to evaluate the presented methods based on simulated data. Besides real-time capability, one of the main approaches of this work is to obtain the best possible accuracy.

We will examine the accuracy of our approach by measuring the deviation of the estimate from the real state of the system. Additionally we will also regard to the overall amount of uncertainty about the state. It is distinguished between the usage of a monocular and a stereo camera. The outcomes are compared to explore whether the usage of a stereo camera is beneficial in terms of accuracy.

We fill firstly describe the evaluation method and the experimental setup. After that, the results are presented.

7.1 Setup

The following vision sensors are compared:

"Mono" This refers to the application of a monocular camera to the 3D SLAM problem as described in Section 5. Single 2D projections of 3D landmarks are obtained after each time step and related to the predicted state of the system.

"Stereo" This refers to the application of a stereo camera to the 3D SLAM problem as described in Section 5.3. The obtained measurements after each time step consist of the 2D projection on the left image and the corresponding disparity.

The usage of both vision sensors yield to an estimate of the system's state at the current point in time and the according uncertainty about that estimate represented by the error covariance matrix $\mathbf{P}$.

Two different scenarios are evaluated. They are depicted in Figure 7.1 and 7.2. In both cases, the scene consist of 20 landmarks randomly distributed inside a cuboid. These 3D feature points are projected onto the image plane(s) with the following internal parameters which are equal for the both camera systems:

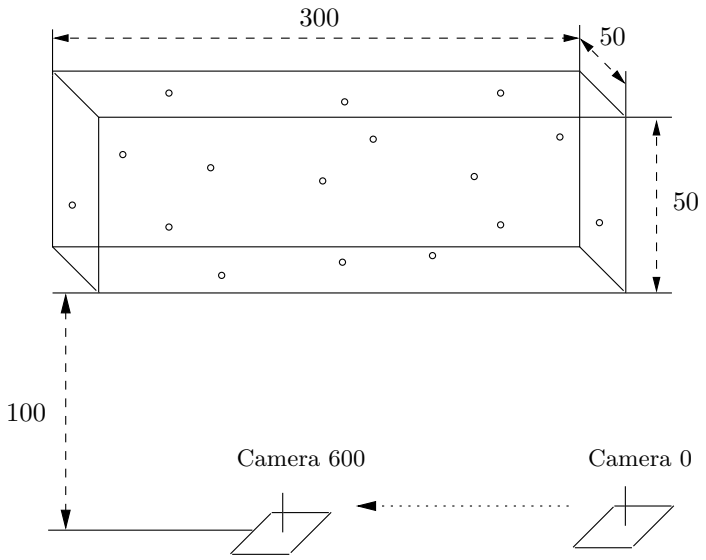

Figure 7.1: *First Scenario. The camera moves along the* x*−axis, perpendicular to the viewing direction.*

image width	640 pixels
image height	480 pixels
focal length	800 pixels
principal point	$x = 320$, $y = 240$ pixels

The baseline of the stereo camera has a length of 10 units.

In both scenarios, the camera moves with a linear velocity of 0.5 units per time step either along the x− or z−axis. 600 camera positions are used. To simulate noisy measurements, we perturb the 2D projections by adding noise vectors $n = (n_x, n_y)^\top$ as a realisation of normally distributed white noise with zero mean and a certain standard deviation. We performed experiments with a deviation of either 2.0 or 1.0 in the x− and y−direction.

The Kalman Filter is initialised with the real starting position of the camera within the world coordinate frame. The entries in the main diagonal of the initial error covariance matrix referring to the camera position are set to zero. This means, that the filter is perfectly sure about the location of the camera.

In contrast to that, the positions of the features are just known approximately. This is represented by an initial error covariance matrix where the entries of the main diagonal referring to the features are set to 10. The real coordinates of the landmarks are perturbed according to this initial uncertainty and inserted into the initial state. All other entries of the error covariance matrix outside the main diagonal are equal to zero. This means that neither the features are correlated with each other nor with the camera position.

The standard deviation of the process noise referring to the linear velocity of the camera in all directions is set to 0.01. The deviation of the measurement noise are set to the real values: either 2.0 or 1.0.

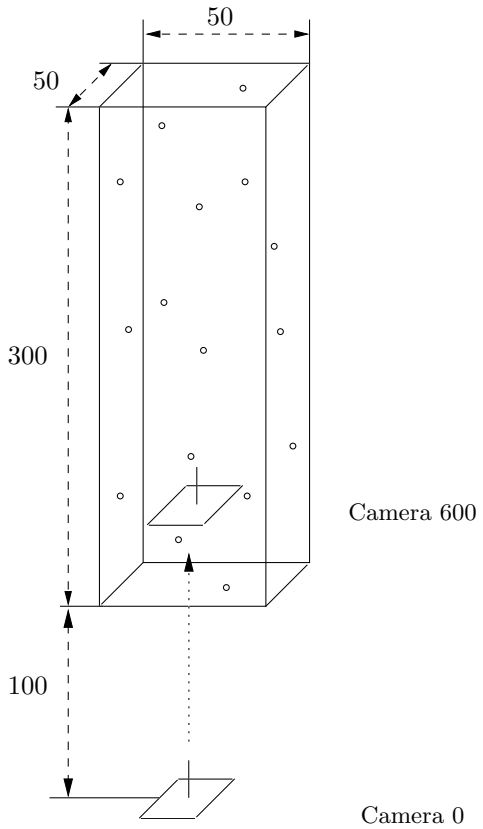

Figure 7.2: *Second Scenario. The camera moves along the* z*−axis, parallel to the viewing direction.*

7.2 Results

The criteria that were used to evaluate the usage of both vision sensors are:

Position error The Euclidean distance between the estimated position of the camera and the real position within the world coordinate frame at a certain point in time.

Position uncertainty The volume of the ellipsoid of confidence representing the uncertainty about the position of the camera within the world coordinate frame.

Mapping error The magnitude of the error vector between the estimated positions of all known landmarks and the real positions within the camera coordinate frame at a certain point in time.

Mapping uncertainty The sum of the volumes of all ellipsoids of confidence representing the uncertainty about the positions of the landmarks within the camera coordinate frame.

In contrast to the first two criterias, the last ones are relative to the camera coordinate frame instead of to the world coordinate frame. Because of this, the considered values for mapping error or mapping uncertainty are independent of the corresponding values for the estimation of the camera position relative to the world. They just depend on the current position of the 3D landmarks relative to the camera.

We will begin with evaluating exemplarily one instance of the first scenario with respect to the mentioned criterias. The results for the usage of two different measurement error variances are compared.

After that, we go on analyzing one instance of the second scenario. We will perform experiments with the same different measurement error variances as for the first scenario.

The stereo camera has to deal with measurements that are perturbed twice: one time in the left and one time in the right camera system. By using different noise variances in the measurements, we like to find out if the advantage of the stereo camera over the single camera in terms of accuracy is negativly affected by increasing the measurement error.

For both scenarios, we expect that the stereo camera performs better than the single camera. Especially, the second scenario should just provide poor information on the depth of the features for the monocular camera. Therefore, error and uncertainty for localization and mapping should be higher than in the case we use the stereo camera as the vision sensor.

7.2.1 Evaluation of the First Scenario

In the first scenario, the camera moves perpendicular to the viewing direction. In Figure 7.3 and Figure 7.4 the evolution of the position error is depicted with a measurement error variance of either $(1.0)^2$ or $(2.0)^2$. First of all, we can state that considered over most of the filter cycles, the error in the position estimates for the monocular camera is constantly increasing no matter what measurement noise is introduced. Just between filter cycle 200 and 400 we have a quite long period of decreasing position error. This can be traced back to special conditions

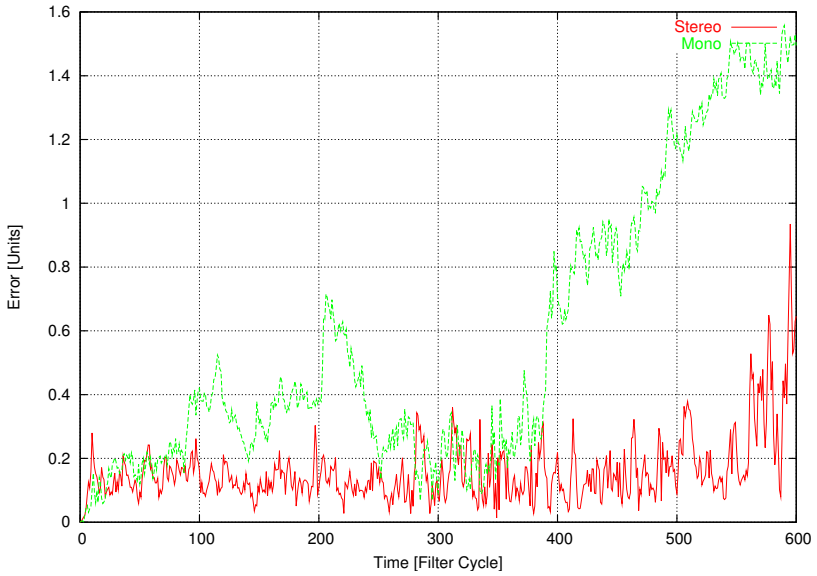

Figure 7.3: *Position error for motion perpendicular to the viewing direction (first scenario) for each filter cycle with a measurement error variance of* $(1.0)^2$.

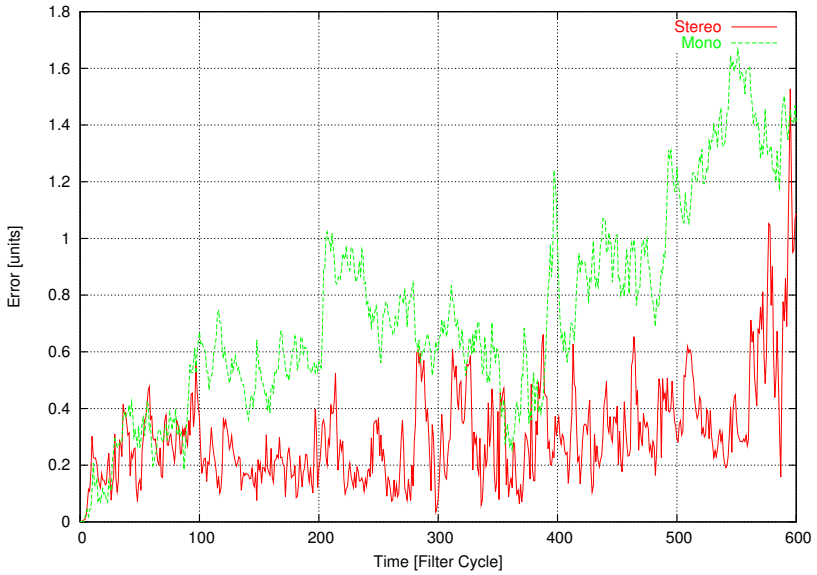

Figure 7.4: *Position error for motion perpendicular to the viewing direction (first scenario) for each filter cycle with a measurement error standard deviation of* 2.0.

in the considered instance of the first scenario. From filter cycle 200 to 300 a bunch of landmarks comes into view and stays there until approximatly filter cycle 390. Then we have a gap between this former bunch and the next small group of features. The estimated positions for the landmarks coming into view are therefore less correlated to the already measured ones. Additionally, just few of the twenty landmarks in the scene provide information to orientate at. The position error increases again.

Compared to the monocular camera, the position error derived by SLAM with a stereo camera seems to oscillate around a rather constant and low value. Just at the end of the scenario it strongly increases. This is also entailed by the small amount of visible landmarks.

The drift in the position estimation for the monocular camera is caused by the accumulation of error due to the process and measurement noise. As already explained in Section 5.1, the error would decrease if the features measured at the beginning of the experiment come into view again. Of course, this also holds for the stereo camera. Because of more accurate measurements, the position estimation for the stereo camera suffers less from accumulating error.

If we compare the resulting curves for a measurement error standard deviation of 1.0 with the curves derived by experiments with a measurement error standard deviation of 2.0, we can note that the overall amount of error rises for the usage of both vision sensors. The deviation of the error values also increases for the single and stereo camera.
In the following table the average error values are given.

	1.0	2.0
Mono	0.578149	0.738551
Stereo	0.161140	0.305079

We can state, that the average position error is smaller if we use a stereo camera. It increases stronger than the error for the single camera if we introduce the double measurement error variance.

In Figure 7.5 and Figure 7.6 the position uncertainty is diagrammed also for the two mentioned values of the measurement error variance.

For both cases of possible measurement noise, we can state that the position uncertainty increases if we use a single as well as a stereo camera. The slope of the curve for the monocular camera is much steeper. The general rising of the position uncertainty is as already explained due to the noisy process and measurements. Because the stereo camera obtains informations about the landmark positions that are much more accurate, it is also much more certain about its own position.

If we compare the curves for a measurement error variance of $(1.0)^2$ and $(2.0)^2$, we can again note a general rise of uncertainty for both vision sensors. The relative growth of uncertainty is slightly better for the monocular camera. Nevertheless, in a direct comparison, the stereo camera is clearly benificial in terms of position uncertainty.

In Figure 7.7 and Figure 7.8 the mapping error is depicted.

In general, we can determine a constantly decreasing mapping error for both vision sensors and measurement error variances. The echelon form is due to

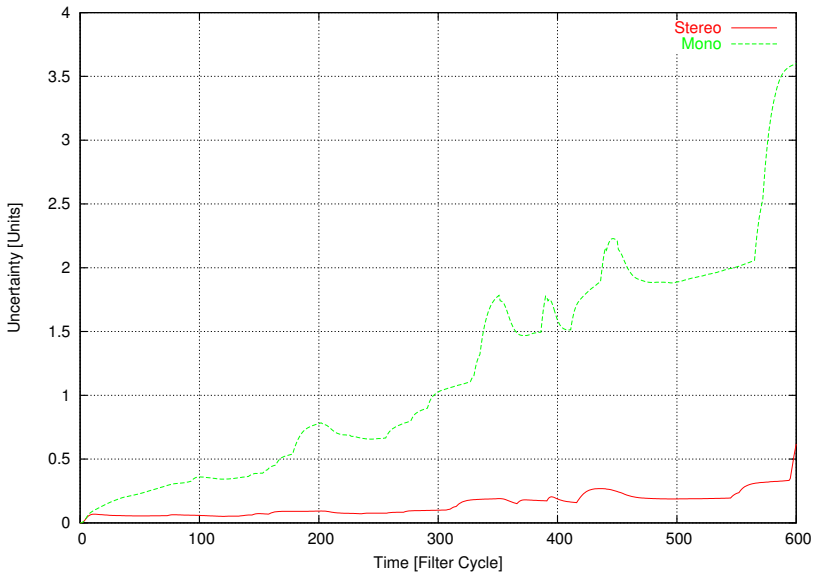

Figure 7.5: *Position uncertainty for motion perpendicular to the viewing direction (first scenario) for each filter cycle with a measurement error standard deviation of* 1.0.

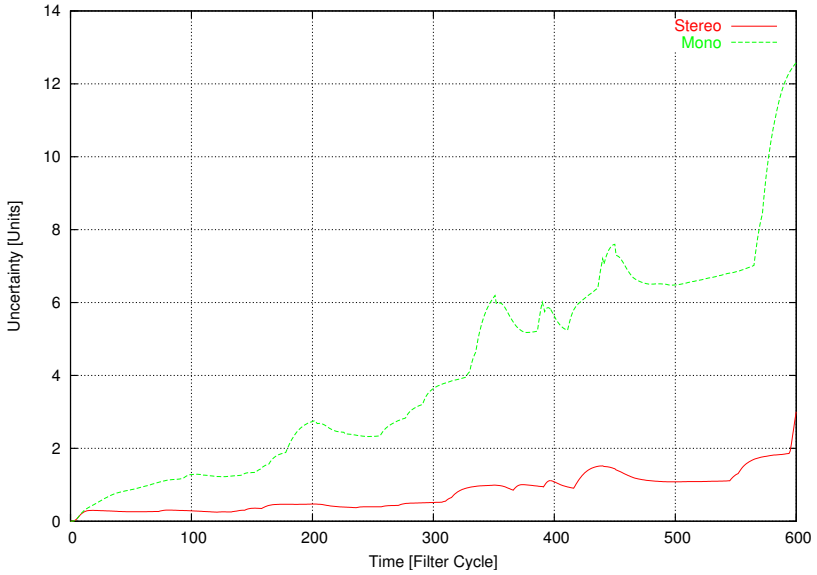

Figure 7.6: *Position uncertainty for motion perpendicular to the viewing direction (first scenario) for each filter cycle with a measurement error standard deviation of* 2.0.

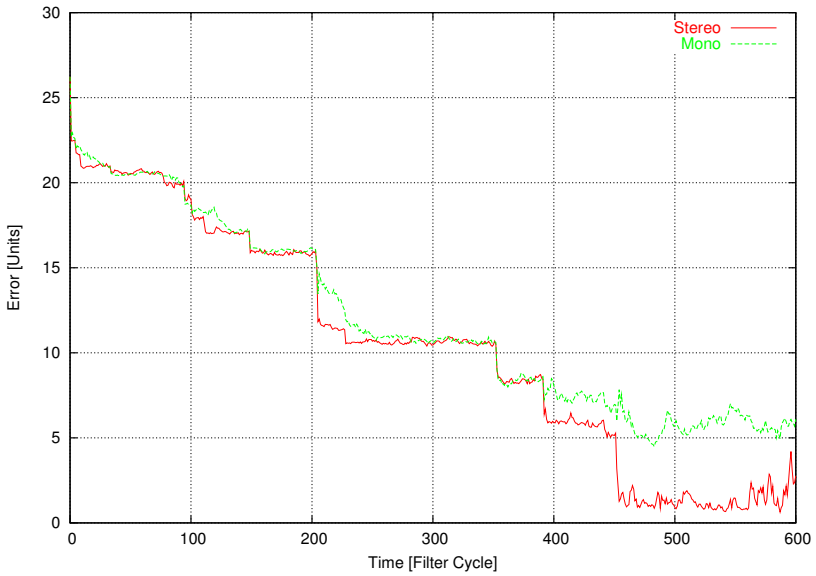

Figure 7.7: *Mapping error for motion perpendicular to the viewing direction (first scenario) for each filter cycle with a measurement error standard deviation of* 1.0.

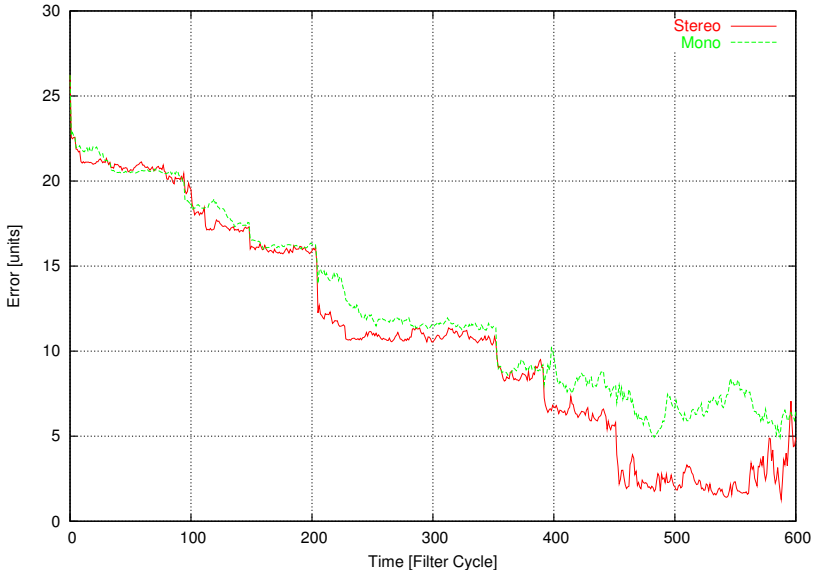

Figure 7.8: *Mapping error for motion perpendicular to the viewing direction (first scenario) for each filter cycle with a measurement error standard deviation of* 2.0.

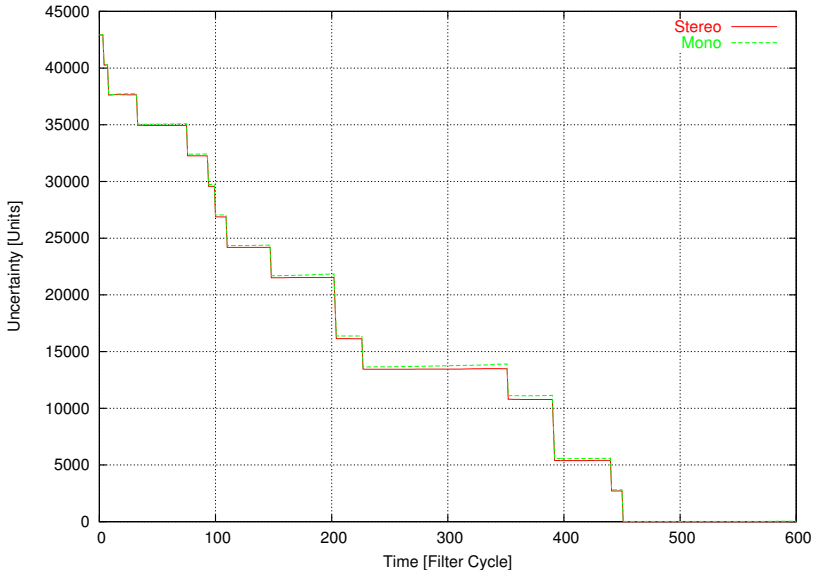

Figure 7.9: *Mapping uncertainty for motion perpendicular to the viewing direction (first scenario) for each filter cycle with a measurement error standard deviation of* 1.0.

the sudden increment of information when a landmarks comes into view for the first time. For most of the filter cycles, the accuracy when using a single or a stereo camera just differs slightly. From approximately filter cycle 450 to 550 we have a long period where the stereo camera provides better results. Here, the higher amount of information on the depth of the already measured features take effect.

The quite good performance of the monocular camera can be explained by the kind of scenario. The sideways motion of the camera provides beneficial measurements such that the single camera obtains many informations on the depth of the considered landmarks over time.

In the following table the average mapping error derived by the usage of each vision sensor and for different possible measurement error are given.

	1.0	2.0
Mono	11.900710	12.502029
Stereo	10.449180	10.927803

The slightly faster relative growth of position error when using a stereo camera and introducing an increasing measurement error variance does not occur when evaluating the mapping error. Here, the single camera performs worse even when considering the relative growth of the mapping error.

In Figure 7.9 and Figure 7.10 the mapping uncertainty is illustrated also for the different measurement error variances. Because the range of the values is that large, the developing between filter cycle 500 and 600 is depicted in a

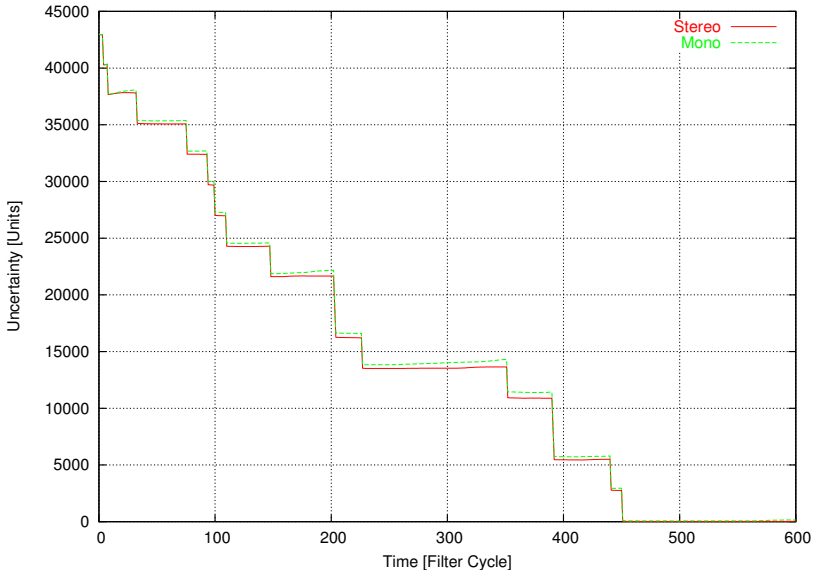

Figure 7.10: *Mapping uncertainty for motion perpendicular to the viewing direction (first scenario) for each filter cycle with a measurement error standard deviation of* 2.0.

larger scale in Figure 7.11 and Figure 7.12. As already noted for the mapping error, also the mapping uncertainty is constantly decreasing. The echelon form is caused by the sudden drop of uncertainty when measuring a feature for the first time. If we consider the enlargements of Figure 7.9 and 7.10, we can note a slight increase of uncertainty after the sudden drop. The gain in information about the currently visible features is canceled out by the increase in uncertainty about the position of the landmarks currently out of view.

This effect is due to the used evaluation criteria, where the correlation between the features is not taken into consideration. It enforced at the end of the experiment, where just three features are visible and the knowledge about the position of the others is getting more and more uncertain.

In terms of mapping uncertainty, the stereo camera performs better. The difference is not that significantly as for the position uncertainty.

If we compare the relative growth of mapping uncertainty when introducing a measurement error covariance of $(2.0)^2$, just a slight disadvantage of the stereo camera can be noted.

7.2.2 Evaluation of the Second Scenario

In the second scenario, the camera moves parallel to the viewing direction into a cuboid where twenty 3D landmarks are distributed randomly. In Figure 7.13 and Figure 7.14 the position error is depicted with a measurement error standard deviation of either 1.0 or 2.0.

Let us first consider Figure 7.3. As already noted for the position error in the first scenario, it is constantly increasing if a single camera is used. For the case of a stereo camera, the curve is oscillating about a rather constant value. Just

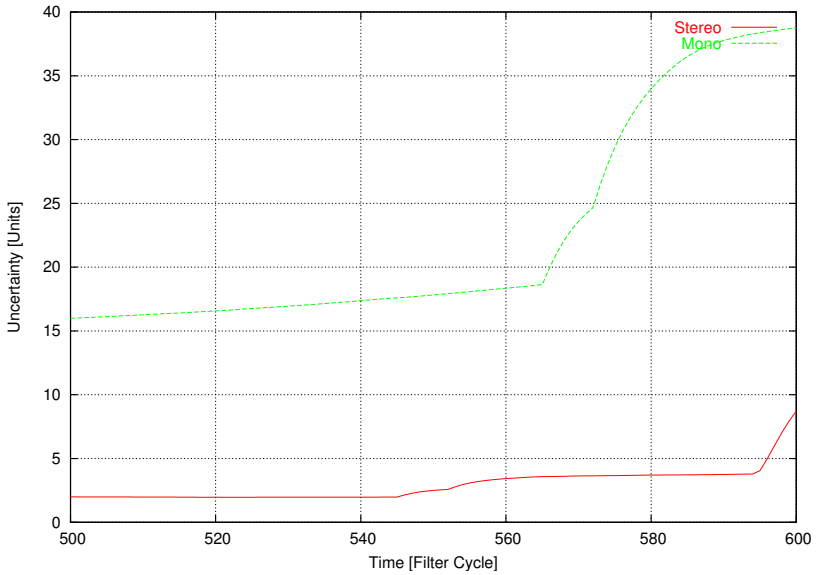

Figure 7.11: *Mapping uncertainty in the first scenario between filter cycle 500 and 600. Clipping of Figure 7.9.*

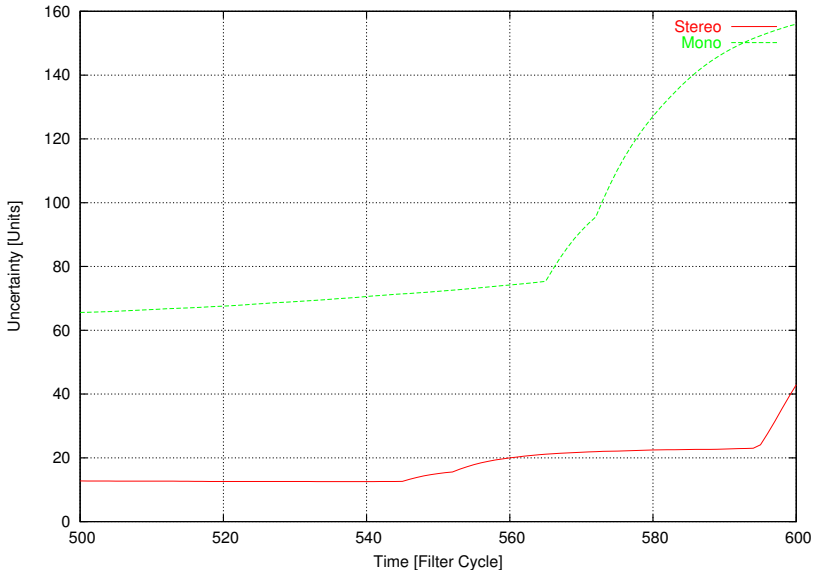

Figure 7.12: *Mapping uncertainty in the first scenario between filter cycle 500 and 600. Clipping of Figure 7.10.*

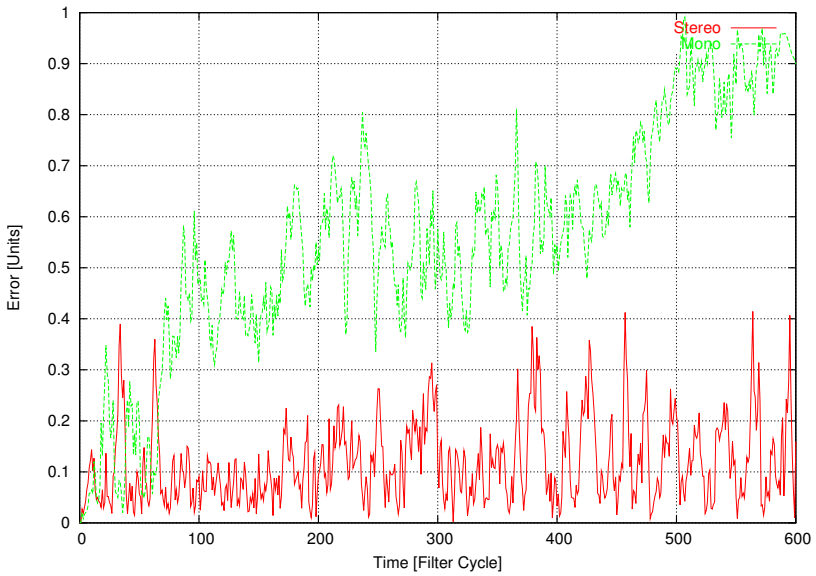

Figure 7.13: *Position error for motion parallel to the viewing direction (second scenario) for each filter cycle with a measurement error standard deviation of* 1.0.

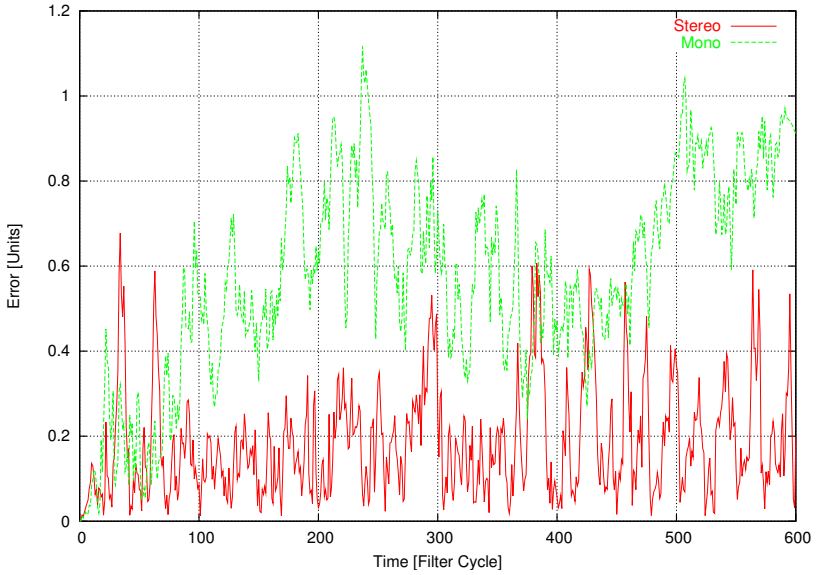

Figure 7.14: *Position error for motion parallel to the viewing direction (second scenario) for each filter cycle with a measurement error standard deviation of* 2.0.

a slightly increasing deviation of the error values appears. Considering position error, we can state that the advantages of the stereo camera are emphasised in the second scenario. In the following, we will show that this effect will continue for the other criterias.

This is clearly due to the ability of the stereo camera to obtain information about the depth of a considered landmark from just one measurement. The single camera needs several measurements to infer rather uncertain depth values. In this scenario the motion of the feature points relative to the camera provide less information about their depth than in the first scenario. Thus, the monocular vision sensor suffers from a stronger accumulation of the error.

If we have a closer look at Figure 7.4 where a higher measurement error variance is used, we can note a general increase of the position error for both vision sensors. The curve regarding to the stereo camera is again oscillating about a rather constant value. This constant value is higher than the one for a measurement error deviation of 1.0. Also the the deviation of values increases.

Between filter cycle 200 and 400, the curve regarding to the monocular camera is decreasing. This is caused by special conditions in the considered instance of the second scenario. In this period of the experiment, a quite large bunch of landmarks to orientate at is constantly visible.

In the following table, the average position error for the usage of the different vision sensors and possible measurement errors are given.

	1.0	2.0
Mono	0.559175	0.574538
Stereo	0.116864	0.183878

The relative growth of the average position error when introducing more measurement noise is higher for the usage of a stereo camera than for a monocular camera. Nevertheless, the stereo camera is clearly beneficial when comparing the curves directly.

If we re-consider the average position errors for the first scenario, we can see that the corresponding values for the second scenario are lower. This is caused by the circumstance that in the latter all known features are visible from the initial position of the camera. The “early” position estimates for all feature points are therefore more certain than for the case the camera moves perpendicular to the viewing direction. They pose better flags during the camera motion as we explained in Section 5.1.

In Figure 7.15 and Figure 7.16 the position uncertainty is illustrated for the different mentioned measurement error variances.

Independent of the introduced possible measurement error, we can note that the uncertainty about the position of the single camera increases constantly whereas the uncertainty about the position of the stereo camera stays rather constant. The hooks in the curves refer to situations where a feature gets invisible and certainty about the camera position needs to be recovered. The rise of the uncertainty at the end of the experiment is caused by the fact that just two features are still visible.

The increase in the uncertainty about the monocular camera position is caused by a shrinking amount of landmarks to orientate at. For the stereo

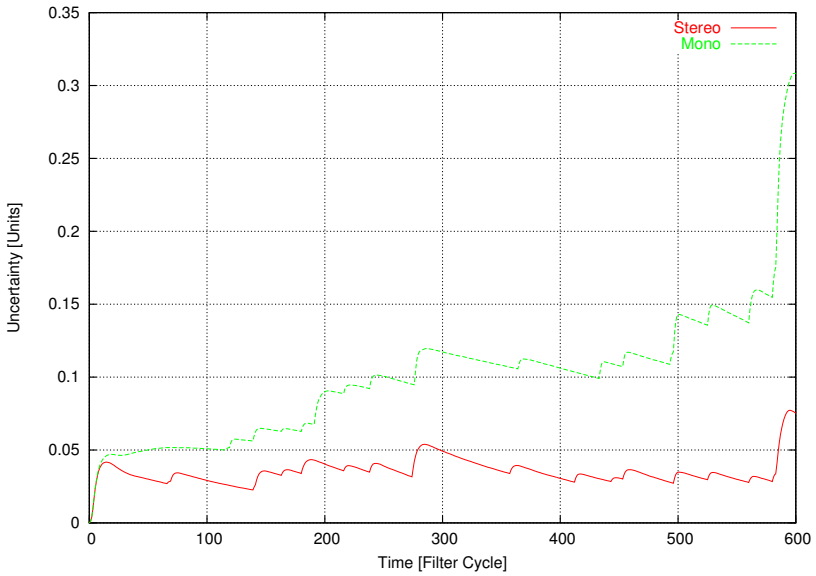

Figure 7.15: *Position uncertainty for motion parallel to the viewing direction (second scenario) for each filter cycle with a measurement error standard deviation of* 1.0.

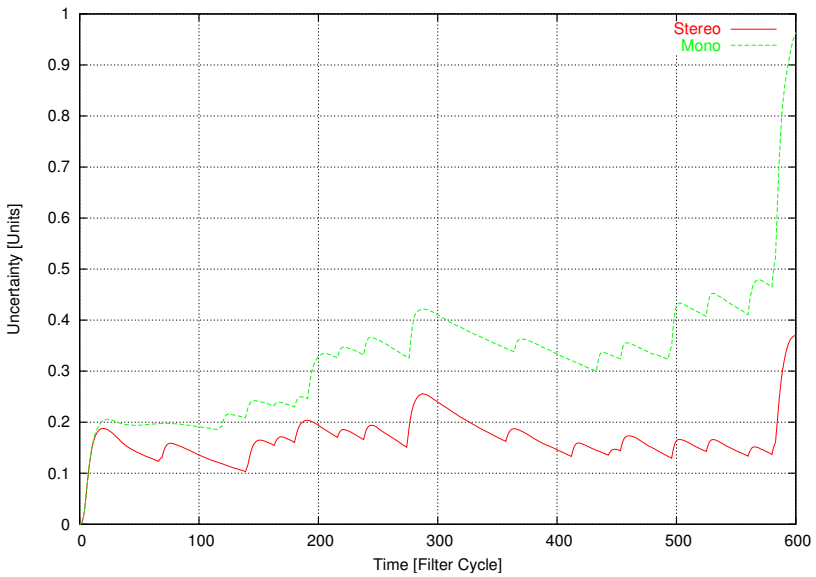

Figure 7.16: *Position uncertainty for motion parallel to the viewing direction (second scenario) for each filter cycle with a measurement error standard deviation of* 2.0.

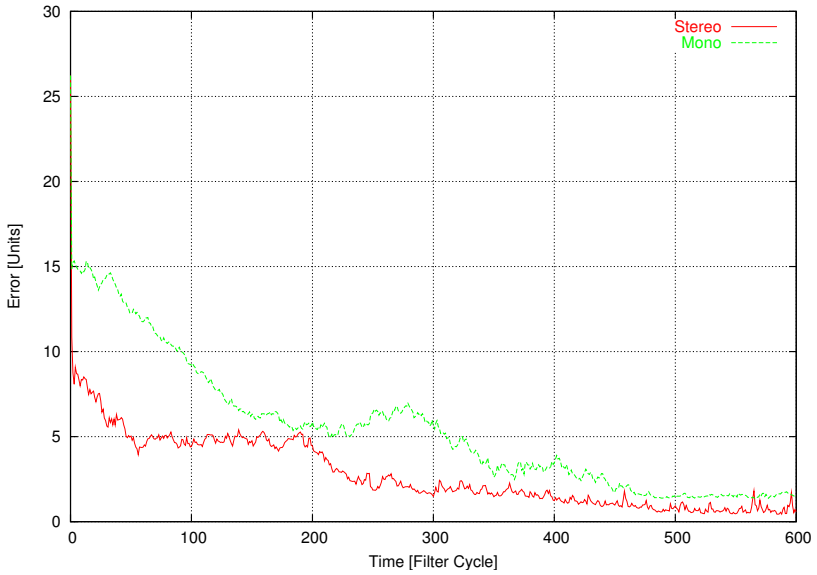

Figure 7.17: *Mapping error for motion parallel to the viewing direction (second scenario) for each filter cycle with a measurement error standard deviation of* 1.0.

camera this poses a smaller problem because it can rely on more certain measurements.

If we introduce more measurement noise, the overall amount of position uncertainty increases independent of which vision sensor is used. Considering the relative growth of uncertainty, the stereo camera performs again a bit worse.

In Figure 7.17 and Figure 7.18 the mapping error is presented.

In general, we can note a constant decrease of the mapping error for both vision sensors and independent of the measurement error as it was already the case for the first scenario. The curves have no echelon form because the feature do not come into view bit by bit, but are all visible right from the beginning. This initially large amount of information about all known landmarks is also the reason for the faster shrinking of the error values to a low level compared to the first scenario.

Mostly, there is a significant distance between the lower mapping error provided by the stereo camera and the values of the single camera. But there are exceptions that are again caused by specific conditions given in the correspoding instance of the second scenario. Consider the curves for a measurement error standard deviation of 1.0. Between approximately filter cycle 50 and 200 the mapping error for the stereo camera is rather constant whereas the error provided by the single camera decreases. During this period of the experiment, a number of features gets out of view. Regarding to the image plane, the corresponding projections change their position rapidly. This large amount of motion is advantageous for the single camera and compensates the disadvantage that some feature are getting invisible. The amount of information about the

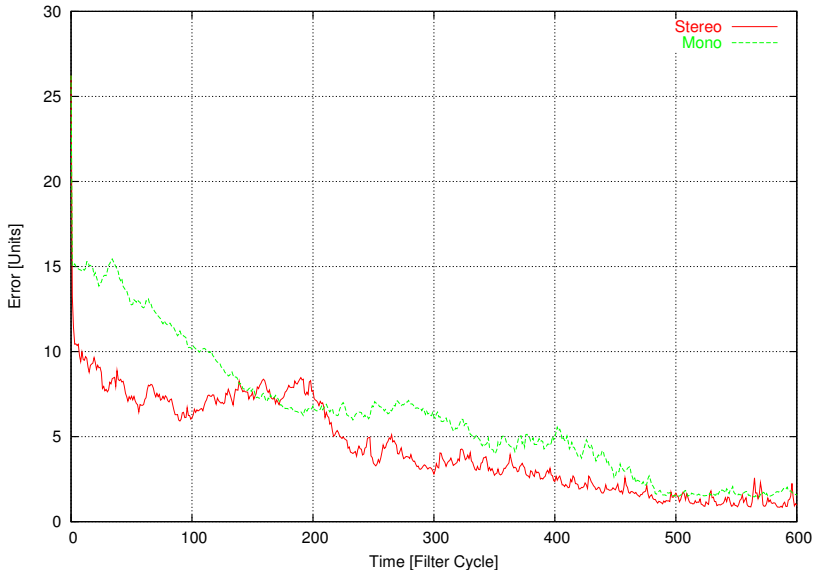

Figure 7.18: *Mapping error for motion parallel to the viewing direction (second scenario) for each filter cycle with a measurement error standard deviation of* 2.0.

landmark position obtained by the stereo camera does not depend on the kind of motion. Therefore, it just suffers from the disappearance of these features. Between the mentioned filter cycles, the mapping error provided by the stereo camera remains rather static.

After that period, it follows a phase where a bunch of landmarks is visible and moves rather straight to the camera. This does not provide that much information for the monocular camera.

In the following table, the average mapping error values are given for both vision sensors and the two different measurement error standard deviations.

	1.0	2.0
Mono	5.478627	6.308648
Stereo	2.816210	4.397468

The relative growth of the average mapping error is significantly higher for the stereo camera when introducing a higher measurement error variance. Nevertheless, the stereo camera performs better than the single camera.

If we compare these values with the results of the first scenario, we can state that the average mapping error is much smaller for the second scenario.

In Figure 7.17 and Figure 7.18 the mapping uncertainty is depicted. Independent of the introduced measurement error and utilized vision sensor, right from the beginning of the experiment the mapping uncertainty is decreasing massivly. This is due to the fact, that initially all known features are

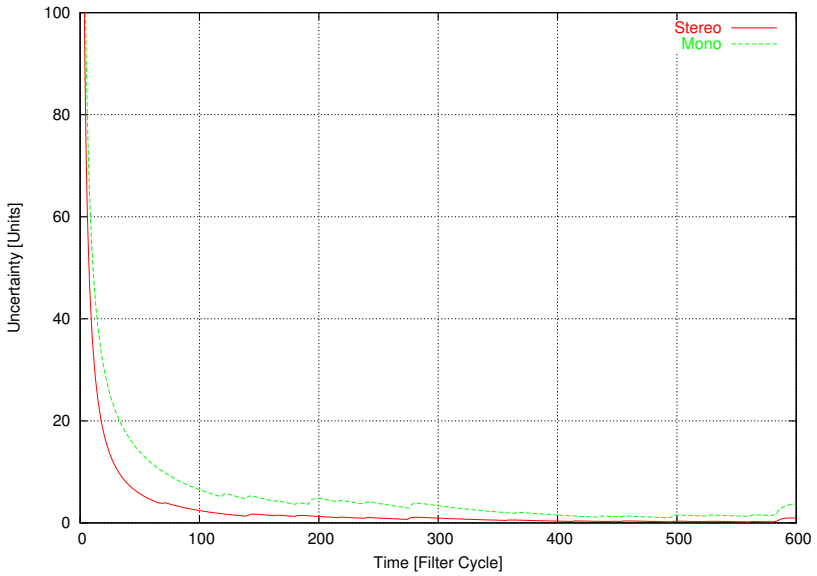

Figure 7.19: *Mapping uncertainty for motion parallel to the viewing direction (second scenario) for each filter cycle with a measurement error standard deviation of* 1.0.

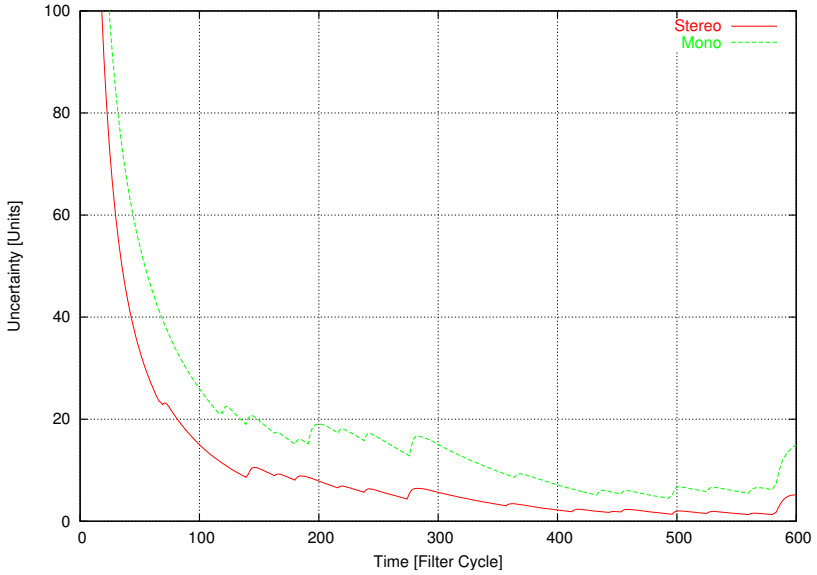

Figure 7.20: *Mapping uncertainty for motion parallel to the viewing direction (second scenario) for each filter cycle with a measurement error standard deviation of* 2.0.

visible. The hooks in the curves are again caused by situations where features become invisible for the according vision sensor.

For both cases of measurement error, the mapping uncertainty obtained by the monocular camera is higher than by the stereo camera. As expected, the overall amount of uncertainty increases with the introduction of a higher measurement error.

7.3 Discussion

If we consider the results of all previous experiments, we can clearly say that the usage of a stereo camera as the measurement device to solve the SLAM problem is advantageous over the usage of a monocular camera.

In general, it provides less error in the position estimates of the camera itself and of the landmarks. The uncertainty about these estimates is also much smaller.

The advantages of the stereo camera are emphasised when the camera moves straight toward landmarks that should be mapped. In the case of such a motion, a monocular camera is just able to obtain very uncertain depth information on the visible landmarks.

If we compare the developing of the results when introducing more measurement noise, we can state that the considered values grow independent of the utilized vision sensor. For the stereo camera, the relative growth of rather error or uncertainty is in most of the experiments worse than for the single camera. But independent of the amount of measurement error, the values provided by the stereo camera will never overtake the values of the single camera. This is due to the fact that both vision sensors obtain the same measurements perturbed by the same noise from the left camera system in case we use a stereo device or rather from the single camera system in case we use a monocular device. As long as the additional information from the right camera system is provided for the state estimation, the stereo camera will perform better. The amount of uncertainty about this addition affects the degree of improvement compared to the state estimate obtained by the single camera.

Chapter 8

Conclusions

The main requirements on vision-based mobile robot navigation are real-time compliance and accuracy. With our approach of 3D Simultaneous Localisation and Mapping, we aim to meet these demands. Based on the Extended Kalman Filter, we want to recover the trajectory of the mobile robot during the period of navigation and build a map of the environment this robot is moving in at the same time. In our navigation method, we track landmarks represented by three-dimensional point features through several images.

In contrast to the classic EKF-SLAM approaches which is mainly based on range-only sensors and provide 2D abstractions of the three-dimensional process of navigation only, our approach is purely based on vision. Visual sensors such as monocular or stereo cameras are able to supply spatial information. In robotics, they can be used to recover the 3D structure of a scene considered. Their weak point is their dependency on the lighting conditions and the scene in general.

In addition to the EKF-SLAM approaches, one may also consider techniques based on multiple-view geometry. In fact, Tobias Pietzsch [24] presented a purely vision-based approach to mobile robot navigation.

The constraints in multiple view geometry where utilised to recover structure and motion of a scene in real-time. The results of the work showed that the error in the trajectory of the robot and in the mapping accumulates over time. In this work, we like to avoid an accumulation of the error and a degradation of accuracy over time.

As the main contribution of our work, we presented the process and sensor models for the application of a monocular and stereo camera to the problem of SLAM. Thereby, we expected the stereo camera to perform better in terms of accuracy than the single camera.

Furthermore, we evaluated the real-time capability of the approach. We presented a method to reduce the time complexity of the EKF from $O(n^3)$ to $O(n^2)$ where n is the number of known landmarks. This method includes that just one of the currently visible features should be measured. A basis for a decision which of the several visible landmarks should be measured is presented.

We were able to demonstrate that the expectation on the higher accuracy achieved by the stereo camera was right. The need to avoid the accumulation of the error in the localisation and mapping was also met by both vision sensors, although the stereo camera performs significantly better. The ability of the filter to re-recognise landmarks after periods of neglect is beneficial to that.

A problem of the Kalman Filter based approach can be found in its computational complexity. If we like to maintain real-time performance, we are just able to carry the knowledge on a very limited number n of landmarks due to a computational time complexity of $O(n^2)$. Davison [9] stated that an implementation of a SLAM shortcut method is required, if the number of features goes past 100. A recognition of for example walls or objects such as tables will not be possible with such a small number of known 3D landmarks.

A subject of our future work is therefore to improve the real-time compliance of the Kalman Filter to the point where scatter plots are obtained in real-time that are feasible to recognise objects. In [38] and [37] a promising hierarchical Kalman Filter approach is presented. The correlation of the feature points is exploited to split up a large filter into several sub-filters and one main-filter.

However, the next step will be to implement an automatic initialization of the filter and a sufficient map management to be able to add and delete feature. We will test this for a sequence of real images.

Appendix A

Jacobian Matrices of the Model Equations

In this work we present a Kalman Filter based approach to solve the SLAM problem with different vision sensors. The process and measurement models are nonlinear whether we use a monocular or a stereo camera. Thus, we need to apply the Extended Kalman Filter. This includes the set up of several Jacobian matrices.

A Jacobian matrix contains all first order partial derivatives of a function f. If this function f is defined as

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

then the Jacobian matrix is a $n \times m$ matrix. To derive them is quite complex but solvable. Especially for the Jacobian matrices of the measurement model we used a method, called *Automatic* or *Algorithmic Differentiation*.

After a short introduction to Automatic Differentiation, we present the Jacobian matrices for the process model which is the same for SLAM with a single camera as well as with a stereo camera. This is followed by the results for the measurement models for each SLAM scenario.

For more detailed information on Automatic Differentiation, have a look at [27] or [11].

A.1 Basic Idea of Automatic Differentiation

The method of Automatic Differentiation is based on the application of the chain rule. If we like to evaluate a function $f(x)$ for a certain x, it is decomposed into elementary functions, like $+, -, *, /$ or sin. For the differentiation of $f(x)$ with respect to x we can easily differentiate these elementary functions and combine them via the chain rule.

To remember, if we have a composite function $f(x) = g(h(x))$ of one variable, the chain rule is

$$f'(x) = g'(h(x)) * h'(x). \tag{A.1}$$

To simplify the notation we introduce $\circ$ to denote the composition of functions. Then, the function f can be written as $f = g \circ h$. Its value at the point x is

denoted as $f(x) = (g \circ h)(x)$. Equation (A.1) then becomes

$$f'(x) = (g' \circ h(x)) * h'(x).$$

To evaluate $f'(x)$ at the point x we firstly need the derivatives g' and h' of g and h. Then, the first factor of the product is derived by evaluate h' at the point x. The second factor can be either derived by the evaluation of the composed function $g' \circ h(x)$ at the point x or the evaluation of g' at the point $h(x)$.

If the function f is composed of a finite number n of functions $g_1, g_2, \ldots, g_{n-1}, g_n$, it can be denoted as

$$f = g_1 \circ g_2 \circ \ldots \circ g_{n-1} \circ g_n$$

By a repeated application of the chain rule, the derivation $f'(x)$ at a point x is given by

$$\begin{aligned} f'(x) &= (g_1' \circ g_2 \circ \ldots \circ g_{n-1} \circ g_n)(x) * (g_2' \circ \ldots \circ g_{n-1} \circ g_n)(x) * \\ &\quad \ldots * (\circ g_{n-1} \circ g_n)(x) * g_n'(x) \end{aligned} \tag{A.2}$$

Now, to evaluate $f'(x)$ at the point x, we need all derivatives $g_1', g_2', \ldots, g_{n-1}', g_n'$ of its components and the values of all factors on the right side of Equation (A.2).

The composite function f can be also represented by a sequence. Thus, if we set

$$\begin{aligned} f_0 &= x \\ f_1 &= g_n(f_0) \\ f_2 &= g_{n-1}(f_1) \\ &\vdots \\ f_{n-1} &= g_2(f_{n-2}) \\ f_n &= g_1(f_{n-1}) \end{aligned} \tag{A.3}$$

then the calculation of the sequence of values $f_1, f_2, \ldots, f_{n-1}, f_n$ yields to $f(x) = f_n$. If we have given this sequence and the derivatives g_i' we can compute the value for the derivative of f at the point x. By applying the chain rule to each row of the Sequence (A.3) except from the first one, we have

$$\begin{aligned} f_0' &= 1 \\ f_1' &= g_n'(f_0) * f_0' \\ f_2' &= g_{n-1}'(f_1) * f_1' \\ &\vdots \\ f_{n-1}' &= g_2'(f_{n-2}) * f_{n-2}' \\ f_n' &= g_1'(f_{n-1}) * f_{n-1}'. \end{aligned} \tag{A.4}$$

Thus, the value for the derivative f' at the point x is $f'(x) = f_n'$. If we substitute $f_0', f_1', \ldots, f_{n-1}'$ back into the appropriate equations of the Sequence (A.4), we derive

$$\begin{aligned} f'(x) &= \prod_{i=1}^{n} g_i'(f_{n-1}) * f_0' \\ &= g_1'(f_{n-1}) * g_2'(f_{n-2}) * \ldots * g_{n-1}'(f_1) * g_n'(f_0) * f_0', \end{aligned}$$

which is required by Equation (A.2).

In the following we will set up the sequences for each considered model as in (A.3) and (A.4) to establish the appropriate Jacobians.

A.2 Process Model for SLAM with a Single and Stereo Camera

Our process model $f(\mathbf{x}, \mathbf{w})$ depends on two independent variable vectors: the state of the system $\mathbf{x}$ and the process noise vector $\mathbf{w}$. The state vector is defined as

$$\mathbf{x} = \begin{bmatrix} \mathbf{t}^{\mathsf{W}} \\ \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} \\ \mathbf{y}_1^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_n^{\mathsf{W}} \end{bmatrix} = \begin{bmatrix} \text{Position of the camera} \\ \text{Orientation of the camera} \\ \text{Linear velocity} \\ \text{Angular velocity} \\ 1^{st} \text{ feature} \\ \vdots \\ n^{th} \text{ feature} \end{bmatrix}.$$

The process noise vector contains

$$\mathbf{w} = \begin{bmatrix} \mathbf{V}^{\mathsf{W}} \\ \mathbf{\Omega}^{\mathsf{CW}} \end{bmatrix} = \begin{bmatrix} \text{Acceleration in linear direction} \\ \text{Acceleration in angular direction} \end{bmatrix}.$$

The process model updates the state $\mathbf{x}$ and provides the state estimate $\mathbf{x_{new}}$ for the next point in time. In detail we have

$$\mathbf{x}_{new} = f(\mathbf{x}, \mathbf{w}) = \begin{bmatrix} \mathbf{t}_{new}^{\mathsf{W}} \\ \mathbf{q}_{new}^{\mathsf{CW}} \\ \mathbf{v}_{new}^{\mathsf{W}} \\ \omega_{new}^{\mathsf{CW}} \\ \mathbf{y}_{1,new}^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_{n,new}^{\mathsf{W}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{\mathsf{W}} + (\mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}}) \triangle k \\ \mathbf{q}((\omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}}) \triangle k) \times \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}} \\ \mathbf{y}_1^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_n^{\mathsf{W}} \end{bmatrix}$$

Because we do not know the individual values of the noise vector $\mathbf{w}$ at each point in time, we will set it to zero. Then we have

$$\hat{\mathbf{x}}_{new} = f(\mathbf{x}, 0) = \begin{bmatrix} \mathbf{t}_{new}^{\mathsf{W}} \\ \mathbf{q}_{new}^{\mathsf{CW}} \\ \mathbf{v}_{new}^{\mathsf{W}} \\ \omega_{new}^{\mathsf{CW}} \\ \mathbf{y}_{1,new}^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_{n,new}^{\mathsf{W}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{\mathsf{W}} + \mathbf{v}^{\mathsf{W}} \triangle k \\ \mathbf{q}(\omega^{\mathsf{CW}} \triangle k) \times \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} \\ \mathbf{y}_1^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_n^{\mathsf{W}} \end{bmatrix}$$

A.2.1 Jacobian Matrix with Respect to the State

We first want to derive the Jacobian matrix $\mathbf{A} = \frac{\partial f}{\partial \mathbf{x}}$ which contains the partial derivatives of $f(\mathbf{x}, \mathbf{w})$ with respect to the state $\mathbf{x}$. If the state is of dimension

m, $\mathbf{A}$ is a $m \times m$ matrix. Represented by several blocks of derivatives, we have

$$\mathbf{A} = \begin{bmatrix} \frac{\partial \mathbf{t}^{\mathsf{W}}_{new}}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial \mathbf{t}^{\mathsf{W}}_{new}}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial \mathbf{t}^{\mathsf{W}}_{new}}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial \mathbf{t}^{\mathsf{W}}_{new}}{\partial \omega^{\mathsf{CW}}} & \frac{\partial \mathbf{t}^{\mathsf{W}}_{new}}{\partial \mathbf{y}^{\mathsf{W}}_{1}} & \cdots & \frac{\partial \mathbf{r}^{\mathsf{W}}_{new}}{\partial \mathbf{y}^{\mathsf{W}}_{n}} \\ \frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \omega^{\mathsf{CW}}} & \frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \mathbf{y}^{\mathsf{W}}_{1}} & \cdots & \frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \mathbf{y}^{\mathsf{W}}_{n}} \\ \frac{\partial \mathbf{v}^{\mathsf{W}}_{new}}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial \mathbf{v}^{\mathsf{W}}_{new}}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial \mathbf{v}^{\mathsf{W}}_{new}}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial \mathbf{v}^{\mathsf{W}}_{new}}{\partial \omega^{\mathsf{CW}}} & \frac{\partial \mathbf{v}^{\mathsf{W}}_{new}}{\partial \mathbf{y}^{\mathsf{W}}_{1}} & \cdots & \frac{\partial \mathbf{v}^{\mathsf{W}}_{new}}{\partial \mathbf{y}^{\mathsf{W}}_{n}} \\ \frac{\partial \omega^{\mathsf{CW}}_{new}}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial \omega^{\mathsf{CW}}_{new}}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial \omega^{\mathsf{CW}}_{new}}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial \omega^{\mathsf{CW}}_{new}}{\partial \omega^{\mathsf{CW}}} & \frac{\partial \omega^{\mathsf{CW}}_{new}}{\partial \mathbf{y}^{\mathsf{W}}_{1}} & \cdots & \frac{\partial \omega^{\mathsf{CW}}_{new}}{\partial \mathbf{y}^{\mathsf{W}}_{n}} \\ \frac{\partial \mathbf{y}^{\mathsf{W}}_{1,new}}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial \mathbf{y}^{\mathsf{W}}_{1,new}}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial \mathbf{y}^{\mathsf{W}}_{1,new}}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial \mathbf{y}^{\mathsf{W}}_{1,new}}{\partial \omega^{\mathsf{CW}}} & \frac{\partial \mathbf{y}^{\mathsf{W}}_{1,new}}{\partial \mathbf{y}^{\mathsf{W}}_{1}} & \cdots & \frac{\partial \mathbf{y}^{\mathsf{W}}_{1,new}}{\partial \mathbf{y}^{\mathsf{W}}_{n}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}^{\mathsf{W}}_{n,new}}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial \mathbf{y}^{\mathsf{W}}_{n,new}}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial \mathbf{y}^{\mathsf{W}}_{n,new}}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial \mathbf{y}^{\mathsf{W}}_{n,new}}{\partial \omega^{\mathsf{CW}}} & \frac{\partial \mathbf{y}^{\mathsf{W}}_{n,new}}{\partial \mathbf{y}^{\mathsf{W}}_{1}} & \cdots & \frac{\partial \mathbf{y}^{\mathsf{W}}_{n,new}}{\partial \mathbf{y}^{\mathsf{W}}_{n}} \end{bmatrix}$$

Most of these blocks are trivial. If $\mathbf{I}$ denotes the unit matrix and $\mathbf{0}$ the zero matrix, $\mathbf{A}$ can be simplified to

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{I}\triangle k & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \mathbf{q}^{\mathsf{CW}}} & \mathbf{0} & \frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \omega^{\mathsf{CW}}} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} \end{bmatrix}$$

The blocks $\frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \mathbf{q}^{\mathsf{CW}}}$ and $\frac{\partial \mathbf{q}^{\mathsf{CW}}_{new}}{\partial \omega^{\mathsf{CW}}}$ are a bit more complicated.

We will first write down the detailed system of equations for $\mathbf{q}^{\mathsf{CW}}_{new} = \mathbf{q}(\omega^{\mathsf{CW}}\triangle k) \times \mathbf{q}^{\mathsf{CW}}$ to enhance the understanding of the differentiation. Then we will differentiate it with respect to the state $\mathbf{x}$ and noise vector $\mathbf{w}$.

The term $\mathbf{q}(\omega^{\mathsf{CW}}\triangle k)$ denotes the quaternion $\mathbf{q}$ which is derived by an angle θ and axis $\mathbf{x}_\theta$, summarised in the three-dimensional vector $\omega^{\mathsf{CW}} = (\omega_x, \omega_y, \omega_z)^\top$. ω^{CW} represents the angular velocity between two points in time. It is pointing in the direction of the rotation and its magnitude $|\omega^{\mathsf{CW}}| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$ is the amount of rotation per second in radians.
According to Definition (B.1), the unit quaternion $\mathbf{q}$ representing a rotation by θ radians about the unit vector $\mathbf{x}_\theta$ is defined as

$$\mathbf{q} = \begin{bmatrix} \cos(\frac{1}{2}\theta) \\ \sin(\frac{1}{2}\theta)\mathbf{x}_\theta \end{bmatrix}.$$

Thus, besides calculating the magnitude of ω^{CW}, we need to normalise it. A vector is normalised by the division of each component by the vector's magnitude. Thus, we have

$$\bar{\omega}^{\mathsf{CW}} = \begin{bmatrix} \bar{\omega}_x \\ \bar{\omega}_y \\ \bar{\omega}_z \end{bmatrix} = \begin{bmatrix} \frac{\omega_x}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k} \\ \frac{\omega_y}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k} \\ \frac{\omega_z}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k} \end{bmatrix}$$

If we now substitute $|\omega^{\mathsf{CW}}|$ for θ and $\bar{\omega}^{\mathsf{CW}}$ for $\mathbf{x}_\theta$ in Equation (A.2.1), we derive

$$\mathbf{q}(\omega^{\mathsf{CW}}\triangle k) = \begin{bmatrix} q_0^\omega \\ q_1^\omega \\ q_2^\omega \\ q_3^\omega \end{bmatrix} = \begin{bmatrix} \cos(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k) \\ \sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k)\frac{\omega_x}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k} \\ \sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k)\frac{\omega_y}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k} \\ \sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k)\frac{\omega_z}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\triangle k} \end{bmatrix} \tag{A.5}$$

The quaternion $\mathbf{q}(\omega^{\mathsf{CW}}\triangle k)$ is just the first factor of the quaternion product $\mathbf{q}(\omega^{\mathsf{CW}}\triangle k) \times \mathbf{q}^{\mathsf{CW}}$. After applying the quaternion multiplication, the result expressed in a matrix multiplication yields

$$\mathbf{q}_{new}^{\mathsf{CW}} = \begin{bmatrix} q_0^{new} \\ q_1^{new} \\ q_2^{new} \\ q_3^{new} \end{bmatrix} = \begin{bmatrix} q_0^\omega q_0^{\mathsf{CW}} - q_1^\omega q_1^{\mathsf{CW}} - q_2^\omega q_2^{\mathsf{CW}} - q_3^\omega q_3^{\mathsf{CW}} \\ q_0^\omega q_1^{\mathsf{CW}} + q_1^\omega q_0^{\mathsf{CW}} + q_2^\omega q_3^{\mathsf{CW}} - q_3^\omega q_2^{\mathsf{CW}} \\ q_0^\omega q_2^{\mathsf{CW}} + q_2^\omega q_0^{\mathsf{CW}} + q_3^\omega q_1^{\mathsf{CW}} - q_1^\omega q_3^{\mathsf{CW}} \\ q_0^\omega q_3^{\mathsf{CW}} + q_3^\omega q_0^{\mathsf{CW}} + q_1^\omega q_2^{\mathsf{CW}} - q_2^\omega q_1^{\mathsf{CW}} \end{bmatrix} \tag{A.6}$$

$$= \begin{bmatrix} q_0^{\mathsf{CW}} & -q_1^{\mathsf{CW}} & -q_2^{\mathsf{CW}} & -q_3^{\mathsf{CW}} \\ q_1^{\mathsf{CW}} & q_0^{\mathsf{CW}} & q_3^{\mathsf{CW}} & -q_2^{\mathsf{CW}} \\ q_2^{\mathsf{CW}} & q_0^{\mathsf{CW}} & q_1^{\mathsf{CW}} & -q_3^{\mathsf{CW}} \\ q_3^{\mathsf{CW}} & q_0^{\mathsf{CW}} & q_2^{\mathsf{CW}} & -q_1^{\mathsf{CW}} \end{bmatrix} \begin{bmatrix} q_0^\omega \\ q_1^\omega \\ q_2^\omega \\ q_3^\omega \end{bmatrix} \tag{A.7}$$

Now, we can differentiate the equations to derive the new quaternion $\mathbf{q}_{new}^{\mathsf{CW}}$ with respect to the old quaternion $\mathbf{q}^{\mathsf{CW}}$. The according block within the Jacobian matrix $\mathbf{A}$ is of the following form

$$\frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \mathbf{q}^{\mathsf{CW}}} = \begin{bmatrix} \frac{\partial q_0^{new}}{\partial q_0^{\mathsf{CW}}} & \frac{\partial q_0^{new}}{\partial q_1^{\mathsf{CW}}} & \frac{\partial q_0^{new}}{\partial q_2^{\mathsf{CW}}} & \frac{\partial q_0^{new}}{\partial q_3^{\mathsf{CW}}} \\ \frac{\partial q_1^{new}}{\partial q_0^{\mathsf{CW}}} & \frac{\partial q_1^{new}}{\partial q_1^{\mathsf{CW}}} & \frac{\partial q_1^{new}}{\partial q_2^{\mathsf{CW}}} & \frac{\partial q_1^{new}}{\partial q_3^{\mathsf{CW}}} \\ \frac{\partial q_2^{new}}{\partial q_0^{\mathsf{CW}}} & \frac{\partial q_2^{new}}{\partial q_1^{\mathsf{CW}}} & \frac{\partial q_2^{new}}{\partial q_2^{\mathsf{CW}}} & \frac{\partial q_2^{new}}{\partial q_3^{\mathsf{CW}}} \\ \frac{\partial q_3^{new}}{\partial q_0^{\mathsf{CW}}} & \frac{\partial q_3^{new}}{\partial q_1^{\mathsf{CW}}} & \frac{\partial q_3^{new}}{\partial q_2^{\mathsf{CW}}} & \frac{\partial q_3^{new}}{\partial q_3^{\mathsf{CW}}} \end{bmatrix}$$

Regarding to Equation (A.6), we can easily derive the partial derivatives.

$$\frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \mathbf{q}^{\mathsf{CW}}} = \begin{bmatrix} q_0^\omega & -q_1^\omega & -q_2^\omega & -q_3^\omega \\ q_1^\omega & q_0^\omega & -q_3^\omega & q_2^\omega \\ q_2^\omega & q_3^\omega & q_0^\omega & -q_1^\omega \\ q_3^\omega & -q_2^\omega & q_1^\omega & q_0^\omega \end{bmatrix}$$

The last block to differentiate within the Jacobian matrix $\mathbf{A}$ is $\mathbf{q}_{new}^{\mathsf{CW}}$ with respect to the average angular velocity ω^{CW}. It is of the following form

$$\frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \omega^{\mathsf{CW}}} = \begin{bmatrix} \frac{\partial q_0^{new}}{\partial \omega_x^{\mathsf{CW}}} & \frac{\partial q_0^{new}}{\partial \omega_y^{\mathsf{CW}}} & \frac{\partial q_0^{new}}{\partial \omega_z^{\mathsf{CW}}} \\ \frac{\partial q_1^{new}}{\partial \omega_x^{\mathsf{CW}}} & \frac{\partial q_1^{new}}{\partial \omega_y^{\mathsf{CW}}} & \frac{\partial q_1^{new}}{\partial \omega_z^{\mathsf{CW}}} \\ \frac{\partial q_2^{new}}{\partial \omega_x^{\mathsf{CW}}} & \frac{\partial q_2^{new}}{\partial \omega_y^{\mathsf{CW}}} & \frac{\partial q_2^{new}}{\partial \omega_z^{\mathsf{CW}}} \\ \frac{\partial q_3^{new}}{\partial \omega_x^{\mathsf{CW}}} & \frac{\partial q_3^{new}}{\partial \omega_y^{\mathsf{CW}}} & \frac{\partial q_3^{new}}{\partial \omega_z^{\mathsf{CW}}} \end{bmatrix}$$

Components of the vector ω^{CW} are just contained in the equations for the quaternion $\mathbf{q}(\omega^{\mathsf{CW}}\triangle k)$. Therefore, to derivate $\mathbf{q}_{new}^{\mathsf{CW}}$ by ω^{CW} we need the partial derivatives $\frac{\partial \mathbf{q}(\omega^{\mathsf{CW}}\triangle k)}{\partial \omega^{\mathsf{CW}}}$. If we have a look at the Matrix Expression (A.7), we can see,

that the components of the left matrix are just multiplicative constants. Thus, $\frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \omega^{\mathsf{CW}}}$ can be conveniently expressed as

$$\frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \omega^{\mathsf{CW}}} = \begin{bmatrix} q_0^{\mathsf{CW}} & -q_1^{\mathsf{CW}} & -q_2^{\mathsf{CW}} & -q_3^{\mathsf{CW}} \\ q_1^{\mathsf{CW}} & q_0^{\mathsf{CW}} & q_3^{\mathsf{CW}} & -q_2^{\mathsf{CW}} \\ q_2^{\mathsf{CW}} & -q_3^{\mathsf{CW}} & q_0^{\mathsf{CW}} & q_1^{\mathsf{CW}} \\ q_3^{\mathsf{CW}} & q_2^{\mathsf{CW}} & -q_1^{\mathsf{CW}} & q_0^{\mathsf{CW}} \end{bmatrix} \begin{bmatrix} \frac{\partial q_0^{\omega}}{\partial \omega_x} & \frac{\partial q_0^{\omega}}{\partial \omega_y} & \frac{\partial q_0^{\omega}}{\partial \omega_z} \\ \frac{\partial q_1^{\omega}}{\partial \omega_x} & \frac{\partial q_1^{\omega}}{\partial \omega_y} & \frac{\partial q_1^{\omega}}{\partial \omega_z} \\ \frac{\partial q_2^{\omega}}{\partial \omega_x} & \frac{\partial q_2^{\omega}}{\partial \omega_y} & \frac{\partial q_2^{\omega}}{\partial \omega_z} \\ \frac{\partial q_3^{\omega}}{\partial \omega_x} & \frac{\partial q_3^{\omega}}{\partial \omega_y} & \frac{\partial q_3^{\omega}}{\partial \omega_z} \end{bmatrix}$$

In the following, just the results of the differentiation of $\mathbf{q}(\omega^{\mathsf{CW}} \triangle k)$ by ω^{CW} are given. We start with the first row of the Jacobian matrix $\frac{\partial \mathbf{q}(\omega^{\mathsf{CW}} \triangle k)}{\partial \omega^{\mathsf{CW}}}$.

$$\frac{\partial q_0^{\omega}}{\omega_x} = -\sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\Delta t)\frac{\omega_x \Delta t}{2\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}} = -\frac{\Delta t^2}{2} q_1^{\omega}$$

In an analogous manner, we have

$$\frac{\partial q_0^{\omega}}{\omega_y} = -\frac{\Delta t^2}{2} q_2^{\omega}$$
$$\frac{\partial q_0^{\omega}}{\omega_z} = -\frac{\Delta t^2}{2} q_3^{\omega}$$

Next, we examine the partial derivatives, where the considered variable is the numerator of the fraction in Equation (A.5).

$$\begin{aligned} \frac{\partial q_1^{\omega}}{\omega_x} =& \cos(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\Delta t)\frac{\Delta t \omega_x^2}{2(\omega_x^2+\omega_y^2+\omega_z^2)} \\ &+ \sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\Delta t)\frac{1}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}} \\ &- \sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\Delta t)\frac{\omega_x^2}{(\omega_x^2+\omega_y^2+\omega_z^2)^{\frac{3}{2}}} \\ =& \frac{\Delta t \omega_x}{\omega_x^2+\omega_y^2+\omega_z^2}\left(q_0^{\omega}\frac{\omega_x}{2} - q_1^{\omega}\right) + \sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\Delta t)\frac{1}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}} \end{aligned}$$

Again, in an analogous manner we derive $\frac{\partial q_2^{\omega}}{\omega_y}$ and $\frac{\partial q_3^{\omega}}{\omega_z}$.

$$\begin{aligned} \frac{\partial q_2^{\omega}}{\omega_y} =& \frac{\Delta t \omega_y}{\omega_x^2+\omega_y^2+\omega_z^2}\left(q_0^{\omega}\frac{\omega_y}{2} - q_2^{\omega}\right) + \sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\Delta t)\frac{1}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}} \\ \frac{\partial q_3^{\omega}}{\omega_z} =& \frac{\Delta t \omega_z}{\omega_x^2+\omega_y^2+\omega_z^2}\left(q_0^{\omega}\frac{\omega_z}{2} - q_3^{\omega}\right) + \sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\Delta t)\frac{1}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}} \end{aligned}$$

Finally, we have the partial derivatives, where the considered variable is just situated within the square roots of Equation (A.5).

$$\begin{aligned}\frac{\partial q_1^\omega}{\omega_y} =& \cos(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\Delta t)\frac{\Delta t\omega_x\omega_y}{2(\omega_x^2+\omega_y^2+\omega_z^2)} \\ & - \sin(\frac{1}{2}\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}\Delta t)\frac{\omega_x\omega_y}{(\omega_x^2+\omega_y^2+\omega_z^2)^{\frac{3}{2}}} \\ =& \frac{\Delta t\omega_y}{\omega_x^2+\omega_y^2+\omega_z^2}\left(q_0^\omega\frac{\omega_x}{2}-q_1^\omega\right)\end{aligned}$$

The same way, we derive the remaining partial derivatives.

$$\begin{aligned}\frac{\partial q_1^\omega}{\omega_z} =& \frac{\Delta t\omega_z}{\omega_x^2+\omega_y^2+\omega_z^2}\left(q_0^\omega\frac{\omega_x}{2}-q_1^\omega\right) \\ \frac{\partial q_2^\omega}{\omega_x} =& \frac{\Delta t\omega_x}{\omega_x^2+\omega_y^2+\omega_z^2}\left(q_0^\omega\frac{\omega_x}{2}-q_2^\omega\right) \\ \frac{\partial q_2^\omega}{\omega_z} =& \frac{\Delta t\omega_z}{\omega_x^2+\omega_y^2+\omega_z^2}\left(q_0^\omega\frac{\omega_x}{2}-q_2^\omega\right) \\ \frac{\partial q_3^\omega}{\omega_x} =& \frac{\Delta t\omega_x}{\omega_x^2+\omega_y^2+\omega_z^2}\left(q_0^\omega\frac{\omega_x}{2}-q_3^\omega\right) \\ \frac{\partial q_3^\omega}{\omega_y} =& \frac{\Delta t\omega_y}{\omega_x^2+\omega_y^2+\omega_z^2}\left(q_0^\omega\frac{\omega_x}{2}-q_3^\omega\right)\end{aligned}$$

A.2.2 Jacobian Matrix with Respect to the Process Noise

In this section we want to derive the Jacobian matrix $\mathbf{W}$ which contains the partial derivatives of the process model $f(\mathbf{x},\mathbf{w})$ with respect to the noise vector $\mathbf{w} = (\mathbf{V}^{\mathsf{W}}, \mathbf{\Omega}^{\mathsf{CW}})^\top$. Each component of $\mathbf{w}$ is a three-dimensional vector. If the state $\mathbf{x}$ is of dimension m, $\mathbf{W}$ is a $m \times 6$ matrix. Represented by several blocks of derivatives, we have

$$\mathbf{W} = \frac{\partial f}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial \mathbf{r}_{new}^{\mathsf{W}}}{\partial \mathbf{V}^{\mathsf{W}}} & \frac{\partial \mathbf{r}_{new}^{\mathsf{W}}}{\partial \mathbf{\Omega}^{\mathsf{CW}}} \\ \frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \mathbf{V}^{\mathsf{W}}} & \frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \mathbf{\Omega}^{\mathsf{CW}}} \\ \frac{\partial \mathbf{v}_{new}^{\mathsf{W}}}{\partial \mathbf{V}^{\mathsf{W}}} & \frac{\partial \mathbf{v}_{new}^{\mathsf{W}}}{\partial \mathbf{\Omega}^{\mathsf{CW}}} \\ \frac{\partial \omega_{new}^{\mathsf{CW}}}{\partial \mathbf{V}^{\mathsf{W}}} & \frac{\partial \omega_{new}^{\mathsf{CW}}}{\partial \mathbf{\Omega}^{\mathsf{CW}}} \end{bmatrix}$$

Just to remember, the process model is given by

$$\mathbf{x}_{new} = f(\mathbf{x},\mathbf{w}) = \begin{bmatrix} \mathbf{t}_{new}^{\mathsf{W}} \\ \mathbf{q}_{new}^{\mathsf{CW}} \\ \mathbf{v}_{new}^{\mathsf{W}} \\ \omega_{new}^{\mathsf{CW}} \\ \mathbf{y}_{1,new}^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_{n,new}^{\mathsf{W}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{\mathsf{W}} + (\mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}})\triangle k \\ \mathbf{q}((\omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}})\triangle k) \times \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} + \mathbf{V}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}} \\ \mathbf{y}_1^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_n^{\mathsf{W}} \end{bmatrix}$$

If we have a look at it, we can see that most of the partial derivatives are trivial again.

$$\mathbf{W} = \frac{\partial f}{\partial \mathbf{w}} = \begin{bmatrix} \mathbf{I}\triangle k & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \mathbf{\Omega}^{\mathsf{CW}}} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

If we now have a closer look at $\mathbf{q}_{new}^{\mathsf{CW}} = \mathbf{q}((\omega^{\mathsf{CW}} + \mathbf{\Omega}^{\mathsf{CW}})\triangle k) \times \mathbf{q}^{\mathsf{CW}}$ we can see, that differential changes to ω^{CW} and $\mathbf{\Omega}^{\mathsf{CW}}$ have the same effect. Thus, if we set $\mathbf{\Omega}^{\mathsf{CW}} = (0,0,0)^\top$ we have

$$\frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \mathbf{\Omega}^{\mathsf{CW}}} = \frac{\partial \mathbf{q}_{new}^{\mathsf{CW}}}{\partial \omega^{\mathsf{CW}}}.$$

A.3 Measurement Model for SLAM with a Single Camera

The measurement model $h(\mathbf{x}, \mathbf{v})$ we use to predict measurements derived by a single camera, depends on two independent variables: the state $\mathbf{x}$ and the measurement noise vector $\mathbf{v}$. As already mentioned in several sections in this work, the state $\mathbf{x}$ is defined as

$$\mathbf{x} = \begin{bmatrix} \mathbf{t}^{\mathsf{W}} \\ \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} \\ \mathbf{y}_1^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_n^{\mathsf{W}} \end{bmatrix} = \begin{bmatrix} \text{Position of the camera} \\ \text{Orientation of the camera} \\ \text{Linear velocity} \\ \text{Angular velocity} \\ 1^{st} \text{ feature} \\ \vdots \\ n^{th} \text{ feature} \end{bmatrix}.$$

The measurement noise vector contains

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \text{Error in } \texttt{x-direction} \\ \text{Error in } \texttt{y-direction} \end{bmatrix}$$

The measurement model relates the state of the system to the measurement of its output. In the case, we use a single camera as the vision sensor, the measurement is represented by a vector $\mathbf{z}$ containing image coordinates $\mathbf{y}_i^{\mathsf{I}}$ of the projections of all visible landmarks $\mathbf{y}_i^{\mathsf{W}}$ in the environment at the considered point in time. If there are l visible landmarks, $\mathbf{z}$ is of the following form

$$\mathbf{z} = h(\mathbf{x}, \mathbf{v}) = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_j \\ \vdots \\ \mathbf{z}_l \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1^{\mathsf{I}} \\ \vdots \\ \mathbf{y}_j^{\mathsf{I}} \\ \vdots \\ \mathbf{y}_l^{\mathsf{I}} \end{bmatrix}$$

The image coordinates $\mathbf{y}_i^{\mathsf{I}}$ are calculated by

$$\mathbf{y}_j^{\mathsf{I}} = \begin{bmatrix} x_j^{\mathsf{I}} \\ y_j^{\mathsf{I}} \end{bmatrix} = \begin{bmatrix} \frac{x_j^{\mathsf{C}}}{z_j^{\mathsf{C}}} + v_x \\ \frac{y_j^{\mathsf{C}}}{z_j^{\mathsf{C}}} + v_y \end{bmatrix} \tag{A.8}$$

where the camera coordinates $\mathbf{y}_j^{\mathsf{C}}$ are derived by

$$\mathbf{y}_j^{\mathsf{C}} = \begin{bmatrix} x_j^{\mathsf{C}} \\ y_j^{\mathsf{C}} \\ z_j^{\mathsf{C}} \end{bmatrix} = \mathbf{R}^{\mathsf{CW}}(\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}}). \tag{A.9}$$

We do not know the individual values of the measurement noise $\mathbf{v}$ at each point in time. Therefore, we set it to zero and derive $h(\mathbf{x}, 0)$. Equation (A.8) then changes to

$$\mathbf{y}_j^{\mathsf{I}} = \begin{bmatrix} x_j^{\mathsf{I}} \\ y_j^{\mathsf{I}} \end{bmatrix} = \begin{bmatrix} \frac{x_j^{\mathsf{C}}}{z_j^{\mathsf{C}}} \\ \frac{y_j^{\mathsf{C}}}{z_j^{\mathsf{C}}} \end{bmatrix} \tag{A.10}$$

In the following we will convert the measurement model into a more detailed form to make the set up of the Jacobian matrices traceable.

We will mainly disassemble Equation (A.9). The matrix $\mathbf{R}^{\mathsf{CW}}$ in this equation denotes the rotation matrix derived by the quaternion $\mathbf{q}^{\mathsf{CW}}$. Further details can be obtained from Appendix B. To be more exactly, this matrix results from applying the quaternion multiplication to the vector $(\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}})$.

$$\mathbf{y}_j^{\mathsf{C}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_j^{\mathsf{W}} - t_x \\ y_j^{\mathsf{W}} - t_y \\ z_j^{\mathsf{W}} - t_z \end{bmatrix} \tag{A.11}$$

where the r's are the components of the rotation matrix $\mathbf{R}^{\mathsf{CW}}$. Expressed in a system of linear equations, the camera coordinates $\mathbf{y}_j^{\mathsf{C}}$ can be calculated by

$$\begin{aligned} x_j^{\mathsf{C}} &= r_{11}(x_j^{\mathsf{W}} - t_x) + r_{12}(y_j^{\mathsf{W}} - t_y) + r_{13}(z_j^{\mathsf{W}} - t_z) \\ y_j^{\mathsf{C}} &= r_{21}(x_j^{\mathsf{W}} - t_x) + r_{22}(y_j^{\mathsf{W}} - t_y) + r_{23}(z_j^{\mathsf{W}} - t_z) \\ z_j^{\mathsf{C}} &= r_{31}(x_j^{\mathsf{W}} - t_x) + r_{32}(y_j^{\mathsf{W}} - t_y) + r_{33}(z_j^{\mathsf{W}} - t_z). \end{aligned}$$

The components of the rotation matrix in detail are

$$\begin{aligned} r_{11} &= (q_0^{\mathsf{CW}})^2 + (q_1^{\mathsf{CW}})^2 - (q_2^{\mathsf{CW}})^2 - (q_3^{\mathsf{CW}})^2 \\ r_{21} &= 2(q_2^{\mathsf{CW}} q_1^{\mathsf{CW}} + q_0^{\mathsf{CW}} q_3^{\mathsf{CW}}) \\ r_{31} &= 2(q_3^{\mathsf{CW}} q_1^{\mathsf{CW}} - q_0^{\mathsf{CW}} q_2^{\mathsf{CW}}) \\ \\ r_{12} &= 2(q_1^{\mathsf{CW}} q_2^{\mathsf{CW}} - q_0^{\mathsf{CW}} q_3^{\mathsf{CW}}) \\ r_{22} &= (q_0^{\mathsf{CW}})^2 - (q_1^{\mathsf{CW}})^2 + (q_2^{\mathsf{CW}})^2 - (q_3^{\mathsf{CW}})^2 \\ r_{32} &= 2(q_0^{\mathsf{CW}} q_1^{\mathsf{CW}} + q_3^{\mathsf{CW}} q_2^{\mathsf{CW}}) \\ \\ r_{13} &= 2(q_1^{\mathsf{CW}} q_3^{\mathsf{CW}} + q_0^{\mathsf{CW}} q_2^{\mathsf{CW}}) \\ r_{23} &= 2(q_2^{\mathsf{CW}} q_3^{\mathsf{CW}} - q_0^{\mathsf{CW}} q_1^{\mathsf{CW}}) \\ r_{33} &= (q_0^{\mathsf{CW}})^2 - (q_1^{\mathsf{CW}})^2 - (q_2^{\mathsf{CW}})^2 + (q_3^{\mathsf{CW}})^2 \end{aligned} \tag{A.12}$$

A.3.1 Jacobian Matrix with Respect to the State

In this section we want to derive the Jacobian matrix $\mathbf{H} = \frac{\partial h}{\partial \mathbf{x}}$ containing the partial derivatives of the measurement model $h(\mathbf{x}, \mathbf{v})$ with respect to the state

$\mathbf{x}$. The Jacobian matrix $\mathbf{H}$, represented by several blocks of derivatives is given as

$$\mathbf{H} = \begin{bmatrix} \frac{\partial h_1}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial h_1}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial h_1}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial h_1}{\partial \omega^{\mathsf{CW}}} & \frac{\partial h_1}{\partial \mathbf{y}_1^{\mathsf{W}}} & \cdots & \frac{\partial h_1}{\partial \mathbf{y}_j^{\mathsf{W}}} & \cdots & \frac{\partial h_1}{\partial \mathbf{y}_l^{\mathsf{W}}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ \frac{\partial h_j}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial h_j}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial h_j}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial h_j}{\partial \omega^{\mathsf{CW}}} & \frac{\partial h_j}{\partial \mathbf{y}_1^{\mathsf{W}}} & \cdots & \frac{\partial h_j}{\partial \mathbf{y}_j^{\mathsf{W}}} & \cdots & \frac{\partial h_j}{\partial \mathbf{y}_l^{\mathsf{W}}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ \frac{\partial h_l}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial h_l}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial h_l}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial h_l}{\partial \omega^{\mathsf{CW}}} & \frac{\partial h_l}{\partial \mathbf{y}_1^{\mathsf{W}}} & \cdots & \frac{\partial h_l}{\partial \mathbf{y}_j^{\mathsf{W}}} & \cdots & \frac{\partial h_l}{\partial \mathbf{y}_l^{\mathsf{W}}} \end{bmatrix}$$

If we have a closer look at the j^{th} row of $\mathbf{H}$, we see that some blocks are again equal to the zero matrix, represented by $\mathbf{0}$.

$$\frac{\partial h_j}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h_j}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial h_j}{\partial \mathbf{q}^{\mathsf{CW}}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \frac{\partial h_j}{\partial \mathbf{y}_j^{\mathsf{W}}} & \dots & \mathbf{0} \end{bmatrix}$$

All the other blocks are non-trivial. But they can be simplified by applying the quotient rule. Any partial derivative of $h(\mathbf{x}, 0)$ will always have the following form:

$$h'_j = \begin{bmatrix} \frac{(x_j^{\mathsf{C}})' * z_j^{\mathsf{C}} - x_j^{\mathsf{C}} * (z_j^{\mathsf{C}})'}{z_j^{\mathsf{C}}()^2} \\ \\ \frac{(y_j^{\mathsf{C}})' * z_j^{\mathsf{C}} - y_j^{\mathsf{C}} * (z_j^{\mathsf{C}})'}{(z_j^{\mathsf{C}})^2} \end{bmatrix}.$$

This can be expressed by a matrix multiplication.

$$h'_j = \mathbf{Q} * (\mathbf{y}_j^{\mathsf{C}})' = \begin{bmatrix} \frac{1}{z_j^{\mathsf{C}}} & 0 & -\frac{x_j^{\mathsf{C}}}{(z_j^{\mathsf{C}})^2} \\ 0 & \frac{1}{z_j^{\mathsf{C}}} & -\frac{y_j^{\mathsf{C}}}{(z_j^{\mathsf{C}})^2} \end{bmatrix} * \begin{bmatrix} (x_j^{\mathsf{C}})' \\ (y_j^{\mathsf{C}})' \\ (z_j^{\mathsf{C}})' \end{bmatrix} \tag{A.13}$$

Thus, to obtain the partial derivatives $\frac{\partial h_j(\mathbf{x},0)}{\partial \mathbf{x}}$ we need to differentiate $\mathbf{y}_j^{\mathsf{C}}$ with respect to $\mathbf{x}$.

According to Equation (A.13), the j^{th} row of $\mathbf{H}$ is then

$$\begin{aligned} \frac{\partial h_j}{\partial \mathbf{x}} &= \mathbf{Q} * \frac{\partial \mathbf{y}_\mathbf{j}^{\mathsf{C}}}{\partial \mathbf{x}} \\ &= \begin{bmatrix} \frac{1}{z_j^{\mathsf{C}}} & 0 & -\frac{x_j^{\mathsf{C}}}{(z_j^{\mathsf{C}})^2} \\ 0 & \frac{1}{z_j^{\mathsf{C}}} & -\frac{y_j^{\mathsf{C}}}{(z_j^{\mathsf{C}})^2} \end{bmatrix} * \begin{bmatrix} \frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{q}^{\mathsf{CW}}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{y}_i^{\mathsf{W}}} & \dots & \mathbf{0} \end{bmatrix} \end{aligned}$$

In the following, just the result for the remaining blocks of partial derivatives are given. We start with $\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{t}^{\mathsf{W}}}$.

$$\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{t}^{\mathsf{W}}} = \begin{bmatrix} \frac{\partial x_j^{\mathsf{C}}}{\partial t_x^{\mathsf{W}}} & \frac{\partial x_j^{\mathsf{C}}}{\partial t_y^{\mathsf{W}}} & \frac{\partial x_j^{\mathsf{C}}}{\partial t_z^{\mathsf{W}}} \\ \frac{\partial y_j^{\mathsf{C}}}{\partial t_x^{\mathsf{W}}} & \frac{\partial y_j^{\mathsf{C}}}{\partial t_y^{\mathsf{W}}} & \frac{\partial y_j^{\mathsf{C}}}{\partial t_z^{\mathsf{W}}} \\ \frac{\partial z_j^{\mathsf{C}}}{\partial t_x^{\mathsf{W}}} & \frac{\partial z_j^{\mathsf{C}}}{\partial t_y^{\mathsf{W}}} & \frac{\partial z_j^{\mathsf{C}}}{\partial t_z^{\mathsf{W}}} \end{bmatrix}$$

If we remember Equation (A.11), we can easily see, that the partial derivatives with respect to each component of the translation vector $\mathbf{t}^{\mathsf{W}} = (t_x, t_y, t_z)^\top$ are

in each case one negated column of the rotation matrix. Thus the whole block $\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{t}^{\mathsf{W}}}$ equates $-\mathbf{R}^{\mathsf{CW}}$. In detail we have

$$\begin{aligned}
\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial t_x^{\mathsf{W}}} &= \begin{bmatrix} -r_{11} \\ -r_{21} \\ -r_{31} \end{bmatrix} = \begin{bmatrix} -(q_0^{\mathsf{CW}})^2 - (q_1^{\mathsf{CW}})^2 + (q_2^{\mathsf{CW}})^2 + (q_3^{\mathsf{CW}})^2 \\ -2(q_2^{\mathsf{CW}} q_1^{\mathsf{CW}} + q_0^{\mathsf{CW}} q_3^{\mathsf{CW}}) \\ -2(q_3^{\mathsf{CW}} q_1^{\mathsf{CW}} - q_0^{\mathsf{CW}} q_2^{\mathsf{CW}}) \end{bmatrix} \\
\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial t_y^{\mathsf{W}}} &= \begin{bmatrix} -r_{12} \\ -r_{22} \\ -r_{32} \end{bmatrix} = \begin{bmatrix} -2(+q_1^{\mathsf{CW}} q_2^{\mathsf{CW}} - q_0^{\mathsf{CW}} q_3^{\mathsf{CW}}) \\ -(q_0^{\mathsf{CW}})^2 + (q_1^{\mathsf{CW}})^2 - (q_2^{\mathsf{CW}})^2 + (q_3^{\mathsf{CW}})^2 \\ -2(q_3^{\mathsf{CW}} q_2^{\mathsf{CW}} + q_0^{\mathsf{CW}} q_1^{\mathsf{CW}}) \end{bmatrix} \\
\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial t_z^{\mathsf{W}}} &= \begin{bmatrix} -r_{13} \\ -r_{23} \\ -r_{33} \end{bmatrix} = \begin{bmatrix} -2(q_1^{\mathsf{CW}} q_3^{\mathsf{CW}} + q_0^{\mathsf{CW}} q_2^{\mathsf{CW}}) \\ -2(q_2^{\mathsf{CW}} q_3^{\mathsf{CW}} - q_0^{\mathsf{CW}} q_1^{\mathsf{CW}}) \\ -(q_0^{\mathsf{CW}})^2 + (q_1^{\mathsf{CW}})^2 + (q_2^{\mathsf{CW}})^2 - (q_3^{\mathsf{CW}})^2 \end{bmatrix}
\end{aligned}$$

In the same manner, we derive the block $\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{y}_j^{\mathsf{W}}}$ of partial derivatives.

$$\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{y}_j^{\mathsf{W}}} = \begin{bmatrix} \frac{\partial x_j^{\mathsf{C}}}{\partial x_j^{\mathsf{W}}} & \frac{\partial x_j^{\mathsf{C}}}{\partial y_j^{\mathsf{W}}} & \frac{\partial x_j^{\mathsf{C}}}{\partial z_j^{\mathsf{W}}} \\ \frac{\partial y_j^{\mathsf{C}}}{\partial x_j^{\mathsf{W}}} & \frac{\partial y_j^{\mathsf{C}}}{\partial y_j^{\mathsf{W}}} & \frac{\partial y_j^{\mathsf{C}}}{\partial z_j^{\mathsf{W}}} \\ \frac{\partial z_j^{\mathsf{C}}}{\partial x_j^{\mathsf{W}}} & \frac{\partial z_j^{\mathsf{C}}}{\partial y_j^{\mathsf{W}}} & \frac{\partial z_j^{\mathsf{C}}}{\partial z_j^{\mathsf{W}}} \end{bmatrix}$$

Here, the whole block resembles the rotation matrix $\mathbf{R}$. The components are given in detail in Equation (A.12).

The block $\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{q}^{\mathsf{CW}}}$ is a bit more complicated.

$$\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial \mathbf{q}^{\mathsf{CW}}} = \begin{bmatrix} \frac{\partial x_j^{\mathsf{C}}}{\partial q_0^{\mathsf{CW}}} & \frac{\partial x_j^{\mathsf{C}}}{\partial q_1^{\mathsf{CW}}} & \frac{\partial x_j^{\mathsf{C}}}{\partial q_2^{\mathsf{CW}}} & \frac{\partial x_j^{\mathsf{C}}}{\partial q_3^{\mathsf{CW}}} \\ \frac{\partial y_j^{\mathsf{C}}}{\partial q_0^{\mathsf{CW}}} & \frac{\partial y_j^{\mathsf{C}}}{\partial q_1^{\mathsf{CW}}} & \frac{\partial y_j^{\mathsf{C}}}{\partial q_2^{\mathsf{CW}}} & \frac{\partial y_j^{\mathsf{C}}}{\partial q_3^{\mathsf{CW}}} \\ \frac{\partial z_j^{\mathsf{C}}}{\partial q_0^{\mathsf{CW}}} & \frac{\partial z_j^{\mathsf{C}}}{\partial q_1^{\mathsf{CW}}} & \frac{\partial z_j^{\mathsf{C}}}{\partial q_2^{\mathsf{CW}}} & \frac{\partial z_j^{\mathsf{C}}}{\partial q_3^{\mathsf{CW}}} \end{bmatrix}$$

In Equation (A.11) we can see, that in this case the vector $(\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}})$ is just a multiplicative constant regarding to the considered variables in each partial derivative. Thus, for each partial derivative we just need to differentiate the appropriate row of the rotation matrix with respect to the quaternion $\mathbf{q}^{\mathsf{CW}}$.

$$\begin{aligned}
\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial q_0^{\mathsf{CW}}} &= \begin{bmatrix} 2q_0^{\mathsf{CW}} & -2q_3^{\mathsf{CW}} & 2q_2^{\mathsf{CW}} \\ 2q_3^{\mathsf{CW}} & 2q_0^{\mathsf{CW}} & -2q_1^{\mathsf{CW}} \\ -2q_2^{\mathsf{CW}} & 2q_1^{\mathsf{CW}} & 2q_0^{\mathsf{CW}} \end{bmatrix} * (\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}}) \\
\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial q_1^{\mathsf{CW}}} &= \begin{bmatrix} 2q_1^{\mathsf{CW}} & 2q_2^{\mathsf{CW}} & 2q_3^{\mathsf{CW}} \\ 2q_2^{\mathsf{CW}} & -2q_1^{\mathsf{CW}} & -2q_0^{\mathsf{CW}} \\ 2q_3^{\mathsf{CW}} & 2q_0^{\mathsf{CW}} & -2q_1^{\mathsf{CW}} \end{bmatrix} * (\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}}) \\
\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial q_2^{\mathsf{CW}}} &= \begin{bmatrix} -2q_2^{\mathsf{CW}} & 2q_1^{\mathsf{CW}} & 2q_0^{\mathsf{CW}} \\ 2q_1^{\mathsf{CW}} & 2q_2^{\mathsf{CW}} & 2q_3^{\mathsf{CW}} \\ -2q_0^{\mathsf{CW}} & 2q_3^{\mathsf{CW}} & -2q_2^{\mathsf{CW}} \end{bmatrix} * (\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}}) \\
\frac{\partial \mathbf{y}_j^{\mathsf{C}}}{\partial q_3^{\mathsf{CW}}} &= \begin{bmatrix} -2q_3^{\mathsf{CW}} & -2q_0^{\mathsf{CW}} & 2q_1^{\mathsf{CW}} \\ 2q_0^{\mathsf{CW}} & -2q_3^{\mathsf{CW}} & 2q_2^{\mathsf{CW}} \\ 2q_1^{\mathsf{CW}} & 2q_2^{\mathsf{CW}} & 2q_3^{RW} \end{bmatrix} * (\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}})
\end{aligned}$$

A.3.2 Jacobian Matrix with Respect to the Measurement Noise

In this section we derive the Jacobian matrix $\mathbf{V} = \frac{\partial \mathbf{h}}{\partial \mathbf{v}}$ containing the partial derivatives of the measurement model $h(\mathbf{x}, \mathbf{v})$ with respect to the noise vector $\mathbf{v} = (v_x, v_y)^\top$. It is of the following form

$$\mathbf{V} = \begin{bmatrix} \frac{\partial x_1^{\mathsf{I}}}{\partial v_x} & \frac{\partial x_1^{\mathsf{I}}}{\partial v_y} \\ \frac{\partial y_1^{\mathsf{I}}}{\partial v_x} & \frac{\partial y_1^{\mathsf{I}}}{\partial v_y} \\ \vdots & \vdots \\ \frac{\partial x_j^{\mathsf{I}}}{\partial v_x} & \frac{\partial x_j^{\mathsf{I}}}{\partial v_y} \\ \frac{\partial y_j^{\mathsf{I}}}{\partial v_x} & \frac{\partial y_j^{\mathsf{I}}}{\partial v_y} \\ \vdots & \vdots \\ \frac{\partial x_l^{\mathsf{I}}}{\partial v_x} & \frac{\partial x_l^{\mathsf{I}}}{\partial v_y} \\ \frac{\partial y_l^{\mathsf{I}}}{\partial v_x} & \frac{\partial y_l^{\mathsf{I}}}{\partial v_y} \end{bmatrix}$$

According to Equation (A.8), the noise vector is an additive constant. Therefore, the Jacobian matrix $\mathbf{V}$ consists of l identity matrices $\frac{\partial h_i}{\mathbf{v}} = \mathbf{I}$

A.4 Measurement Model for SLAM with a Stereo Camera

The measurement model for SLAM with a stereo camera differs from the model for a single camera. It also depends on two independent variables: the state $\mathbf{x}$ and the measurement noise vector $\mathbf{v}$. Of course, the Jacobian matrices $\mathbf{H}$ and $\mathbf{V}$ containing the partial derivatives of the measurement model h with respect to $\mathbf{x}$ and $\mathbf{v}$ change.

As already mentioned in Section 5.3, the state $\mathbf{x}$ remains the same as for the monocular case except from the fact, that the position and orientation of the camera are related to the left camera system.

$$\mathbf{x} = \begin{bmatrix} \mathbf{t}^{\mathsf{W}} \\ \mathbf{q}^{\mathsf{CW}} \\ \mathbf{v}^{\mathsf{W}} \\ \omega^{\mathsf{CW}} \\ \mathbf{y}_1^{\mathsf{W}} \\ \vdots \\ \mathbf{y}_n^{\mathsf{W}} \end{bmatrix} = \begin{bmatrix} \text{Position of the left camera} \\ \text{Orientation of the left camera} \\ \text{Linear velocity} \\ \text{Angular velocity} \\ 1^{st} \text{ feature} \\ \vdots \\ n^{th} \text{ feature} \end{bmatrix}.$$

The measurement noise vector contains

$$\begin{bmatrix} v_{l,x} \\ v_{l,y} \\ v_d \end{bmatrix} = \begin{bmatrix} \text{Error in x}-\text{direction related to the left projection} \\ \text{Error in y}-\text{direction related to the left projection} \\ \text{Error in the Disparity Value} \end{bmatrix}$$

The measurement model relates the state of the system to the measurements of its output. The result is a measurement vector $\mathbf{z}$ containing the image coordinates of the left projection of each fully visible feature point and the according

disparity value. If the measurement related to the fully visible feature $\mathbf{y}_j^{\mathsf{W}}$ is denoted by z_j and l feature are fully visible, as measurement vector $\mathbf{z}$ we have

$$\mathbf{z} = h(\mathbf{x}, \mathbf{v}) = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_j \\ \vdots \\ \mathbf{z}_l \end{bmatrix}$$

where $\mathbf{z}_j$ is given by

$$\mathbf{z}_j = h_j(\mathbf{x}, \mathbf{v}) = \begin{bmatrix} x_{l,j}^{\mathsf{I}} \\ y_{l,j}^{\mathsf{I}} \\ d_j^{\mathsf{I}} \end{bmatrix}$$

. The left projection $\mathbf{y}_{l,j}^{\mathsf{I}}$ of the feature point $\mathbf{y}_j^{\mathsf{W}}$ is derived by

$$\mathbf{y}_{l,j}^{\mathsf{I}} = \begin{bmatrix} x_{l,j}^{\mathsf{I}} \\ y_{l,j}^{\mathsf{I}} \end{bmatrix} = \begin{bmatrix} \frac{x_{l,j}^{\mathsf{C}}}{z_{l,j}^{\mathsf{C}}} + v_x \\ \frac{y_{l,j}^{\mathsf{C}}}{z_{l,j}^{\mathsf{C}}} + v_y \end{bmatrix} \tag{A.14}$$

Because we do not know the individual values for the measurement noise at each point in time, we set it to zero. Then we derive $h(\mathbf{x}, 0)$ where the predicted measurements are estimates of the real measurements.

$$\hat{\mathbf{z}} = h(\mathbf{x}, \mathbf{v}) = \begin{bmatrix} \hat{\mathbf{z}}_1 \\ \vdots \\ \hat{\mathbf{z}}_j \\ \vdots \\ \hat{\mathbf{z}}_l \end{bmatrix}$$

where the image coordinates $\mathbf{y}_{l,j}^{\mathsf{I}}$ of the left projection are calculated without the noise values.

$$\hat{\mathbf{y}}_{l,j}^{\mathsf{I}} = \begin{bmatrix} x_{l,j}^{\mathsf{I}} \\ y_{l,j}^{\mathsf{I}} \end{bmatrix} = \begin{bmatrix} \frac{x_{l,j}^{\mathsf{C}}}{z_{l,j}^{\mathsf{C}}} \\ \frac{y_{l,j}^{\mathsf{C}}}{z_{l,j}^{\mathsf{C}}} \end{bmatrix} \tag{A.15}$$

At last we need the equation to derive the left camera coordinates $\mathbf{y}_{l,j}^{\mathsf{C}}$ from world coordinates of the feature point $\mathbf{y}_j^{\mathsf{W}}$. The demanded equation resembles Equation (A.11).

$$\mathbf{y}_{l,j}^{\mathsf{C}} = \mathbf{R}^{\mathsf{CW}}(\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}}) \tag{A.16}$$

The first two components of the measurement $\mathbf{z}_j$ can be calculated by the last mentioned equations. The third component is the disparity d_j^{I}. It represents the difference between the $\mathsf{x}-$coordinate of the left and right projection of the considered world point.

$$d_j^{\mathsf{I}} = x_{l,j}^{\mathsf{I}} - x_{r,j}^{\mathsf{I}}$$

Thus, we also need to compute the coordinates of the right projections. If the camera coordinates $\mathbf{y}_{r,j}^{\mathsf{C}}$ for one world point are already given, Equation (A.14) and Equation (A.15) can be applied to derive $\mathbf{y}_{r,j}^{\mathsf{I}}$ or $\hat{\mathbf{y}}_{r,j}^{\mathsf{I}}$. The difference to

the right projection is due to the equation for the transformation of world into right camera coordinates. The displacement of the right camera projection centre regarding to the left one, namely the length of the baseline b, must be considered. Then we have

$$\begin{aligned} \mathbf{y}_{r,j}^{\mathsf{C}} &= \mathbf{R}^{\mathsf{CW}}(\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}} - \mathbf{R}^{\mathsf{WC}}\mathbf{b}) && \text{(A.17)} \\ &= \mathbf{R}^{\mathsf{CW}}(\mathbf{y}_j^{\mathsf{W}} - \mathbf{t}^{\mathsf{W}}) - \mathbf{b} && \text{(A.18)} \end{aligned}$$

where $\mathbf{b}$ denotes the vector $(b, 0, 0)^\top$.

We know that Equation (A.16) resembles Equation (A.11). Thus, the according system of equations is also the same. We have

$$\begin{aligned} x_{l,j}^{\mathsf{C}} &= r_{11}(x_j^{\mathsf{W}} - t_x) + r_{12}(y_j^{\mathsf{W}} - t_y) + r_{13}(z_j^{\mathsf{W}} - t_z) \\ y_{l,j}^{\mathsf{C}} &= r_{21}(x_j^{\mathsf{W}} - t_x) + r_{22}(y_j^{\mathsf{W}} - t_y) + r_{23}(z_j^{\mathsf{W}} - t_z) \\ z_{l,j}^{\mathsf{C}} &= r_{31}(x_j^{\mathsf{W}} - t_x) + r_{32}(y_j^{\mathsf{W}} - t_y) + r_{33}(z_j^{\mathsf{W}} - t_z). \end{aligned}$$

where

$$\begin{aligned} r_{11} &= (q_0^{\mathsf{CW}})^2 + (q_1^{\mathsf{CW}})^2 - (q_2^{\mathsf{CW}})^2 - (q_3^{\mathsf{CW}})^2 \\ r_{21} &= 2(q_2^{\mathsf{CW}} q_1^{\mathsf{CW}} + q_0^{\mathsf{CW}} q_3^{\mathsf{CW}}) \\ r_{31} &= 2(q_3^{\mathsf{CW}} q_1^{\mathsf{CW}} - q_0^{\mathsf{CW}} q_2^{\mathsf{CW}}) \\ \\ r_{12} &= 2(q_1^{\mathsf{CW}} q_2^{\mathsf{CW}} - q_0^{\mathsf{CW}} q_3^{\mathsf{CW}}) \\ r_{22} &= (q_0^{\mathsf{CW}})^2 - (q_1^{\mathsf{CW}})^2 + (q_2^{\mathsf{CW}})^2 - (q_3^{\mathsf{CW}})^2 \\ r_{32} &= 2(q_0^{\mathsf{CW}} q_1^{\mathsf{CW}} + q_3^{\mathsf{CW}} q_2^{\mathsf{CW}}) \\ \\ r_{13} &= 2(q_1^{\mathsf{CW}} q_3^{\mathsf{CW}} + q_0^{\mathsf{CW}} q_2^{\mathsf{CW}}) \\ r_{23} &= 2(q_2^{\mathsf{CW}} q_3^{\mathsf{CW}} - q_0^{\mathsf{CW}} q_1^{\mathsf{CW}}) \\ r_{33} &= (q_0^{\mathsf{CW}})^2 - (q_1^{\mathsf{CW}})^2 - (q_2^{\mathsf{CW}})^2 + (q_3^{\mathsf{CW}})^2 \end{aligned} \qquad \text{(A.19)}$$

Equation (A.18) leads to a slightly different system of equations by involving the length of the baseline. By substituting Equation (A.16) into Equation (A.18) we derive:

$$\mathbf{y}_{r,j}^{\mathsf{C}} = \mathbf{y}_{l,j}^{\mathsf{C}} - \mathbf{b} \qquad \text{(A.20)}$$

Then, the system of equations for Equation (A.18) is

$$\begin{aligned} x_{r,j}^{\mathsf{C}} &= r_{11}(x_j^{\mathsf{W}} - t_x) + r_{12}(y_j^{\mathsf{W}} - t_y) + r_{13}(z_j^{\mathsf{W}} - t_z) - b && \text{(A.21)} \\ y_{r,j}^{\mathsf{C}} &= r_{21}(x_j^{\mathsf{W}} - t_x) + r_{22}(y_j^{\mathsf{W}} - t_y) + r_{23}(z_j^{\mathsf{W}} - t_z) && \text{(A.22)} \\ z_{r,j}^{\mathsf{C}} &= r_{31}(x_j^{\mathsf{W}} - t_x) + r_{32}(y_j^{\mathsf{W}} - t_y) + r_{33}(z_j^{\mathsf{W}} - t_z) && \text{(A.23)} \end{aligned}$$

The r's are given in Equation (A.19).

In this case, we are just interested in the components $x_{r,j}^{\mathsf{C}}$ and $z_{r,j}^{\mathsf{C}} = z_{l,j}^{\mathsf{C}}$ to derive $x_{r,j}^{\mathsf{I}}$.

A.4.1 Jacobian Matrix with Respect to the State

In this section we want to derive the Jacobian matrix $\mathbf{H} = \frac{\partial h}{\partial \mathbf{x}}$ containing the partial derivatives of the measurement model $h(\mathbf{x}, 0)$ with respect to the state

$\mathbf{x}$. The matrix $\mathbf{H}$ represented by several blocks of partial derivatives is of the following form

$$\mathbf{H} = \begin{bmatrix} \frac{\partial h_1}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial h_1}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial h_1}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial h_1}{\partial \omega^{\mathsf{CW}}} & \frac{\partial h_1}{\partial \mathbf{y}_1^{\mathsf{W}}} & \cdots & \frac{\partial h_1}{\partial \mathbf{y}_j^{\mathsf{W}}} & \cdots & \frac{\partial h_1}{\partial \mathbf{y}_l^{\mathsf{W}}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ \frac{\partial h_j}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial h_j}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial h_j}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial h_j}{\partial \omega^{\mathsf{CW}}} & \frac{\partial h_j}{\partial \mathbf{y}_1^{\mathsf{W}}} & \cdots & \frac{\partial h_j}{\partial \mathbf{y}_j^{\mathsf{W}}} & \cdots & \frac{\partial h_j}{\partial \mathbf{y}_l^{\mathsf{W}}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ \frac{\partial h_l}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial h_l}{\partial \mathbf{q}^{\mathsf{CW}}} & \frac{\partial h_l}{\partial \mathbf{v}^{\mathsf{W}}} & \frac{\partial h_l}{\partial \omega^{\mathsf{CW}}} & \frac{\partial h_l}{\partial \mathbf{y}_1^{\mathsf{W}}} & \cdots & \frac{\partial h_l}{\partial \mathbf{y}_j^{\mathsf{W}}} & \cdots & \frac{\partial h_l}{\partial \mathbf{y}_l^{\mathsf{W}}} \end{bmatrix}$$

If we have a closer look at the $j^t h$ row of the Jacobian matrix, we can determine, that some blocks are equal to the zero matrix.

$$\frac{\partial h_j}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h_j}{\partial \mathbf{t}^{\mathsf{W}}} & \frac{\partial h_j}{\partial \mathbf{q}^{\mathsf{CW}}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \frac{\partial h_j}{\partial \mathbf{y}_j^{\mathsf{W}}} & \dots & \mathbf{0} \end{bmatrix}$$

All the other blocks are non-trivial. But they can be again simplified by applying the quotient rule. Any partial derivative of $h(\mathbf{x}, 0)$ will always have the following form:

$$\begin{aligned} h'_j &= \begin{bmatrix} \frac{(x_{l,j}^{\mathsf{C}})' * z_{l,j}^{\mathsf{C}} - x_{l,j}^{\mathsf{C}} * (z_{l,j}^{\mathsf{C}})'}{(z_{l,j}^{\mathsf{C}})^2} \\ \frac{(y_{l,j}^{\mathsf{C}})' * z_{l,j}^{\mathsf{C}} - y_{l,j}^{\mathsf{C}} * (z_{l,j}^{\mathsf{C}})'}{(z_{l,j}^{\mathsf{C}})^2} \\ \frac{(x_{l,j}^{\mathsf{C}})' * z_{l,j}^{\mathsf{C}} - x_{l,j}^{\mathsf{C}} * (z_{l,j}^{\mathsf{C}})'}{(z_{l,j}^{\mathsf{C}})^2} - \frac{(x_{r,j}^{\mathsf{C}})' * z_{r,j}^{\mathsf{C}} - x_{r,j}^{\mathsf{C}} * (z_{r,j}^{\mathsf{C}})'}{(z_{r,j}^{\mathsf{C}})^2} \end{bmatrix} \\ &= \frac{1}{(z_{r,j}^{\mathsf{C}})^2} \begin{bmatrix} (x_{l,j}^{\mathsf{C}})' * z_{l,j}^{\mathsf{C}} - x_{l,j}^{\mathsf{C}} * (z_{l,j}^{\mathsf{C}})' \\ (y_{l,j}^{\mathsf{C}})' * z_{l,j}^{\mathsf{C}} - y_{l,j}^{\mathsf{C}} * (z_{l,j}^{\mathsf{C}})' \\ -b(z_{l,j}^{\mathsf{C}})' \end{bmatrix}. \end{aligned}$$

This can be conveniently expressed by a matrix multiplication.

$$(h_j)' = \mathbf{Q} * (\mathbf{y}_j^{\mathsf{C}})' = \frac{1}{(z_{r,j}^{\mathsf{C}})^2} \begin{bmatrix} z_{l,j}^{\mathsf{C}} & 0 & -x_{l,j}^{\mathsf{C}} \\ 0 & z_{l,j}^{\mathsf{C}} & -y_{l,j}^{\mathsf{C}} \\ 0 & 0 & -b \end{bmatrix} * \begin{bmatrix} (x_{l,j}^{\mathsf{C}})' \\ (y_{l,j}^{\mathsf{C}})' \\ (z_{l,j}^{\mathsf{C}})' \end{bmatrix}. \tag{A.24}$$

Thus, to derive the partial derivatives $\frac{\partial h_j(\mathbf{x},0)}{\partial \mathbf{x}}$ we need to differentiate the vector $\mathbf{y}_{l,j}^{\mathsf{C}}$ with respect to the state. $\mathbf{y}_{l,j}^{\mathsf{C}}$ is derived by Equation (A.16) which is the same as for the single camera model. Therefore, also the according partial derivatives with respect to the state are the same and given in Section A.3.1.

A.4.2 Jacobian Matrix with Respect to the Measurement Noise

In this section we derive the Jacobian matrix $\mathbf{V} = \frac{\partial h}{\partial \mathbf{v}}$ containing the partial derivatives of the measurement model $h(\mathbf{x}, 0)$ with respect to the noise vector

$\mathbf{v} = (v_{l,x}, v_{l,y})^\top$. It is of the following form

$$\mathbf{V} = \begin{bmatrix} \frac{\partial x_1^{\mathrm{I}}}{\partial v_{l,x}} & \frac{\partial x_1^{\mathrm{I}}}{\partial v_{l,y}} & \frac{\partial x_1^{\mathrm{I}}}{\partial v_d} \\ \frac{\partial y_1^{\mathrm{I}}}{\partial v_{l,x}} & \frac{\partial y_1^{\mathrm{I}}}{\partial v_{l,y}} & \frac{\partial y_1^{\mathrm{I}}}{\partial v_d} \\ \frac{\partial d_1^{\mathrm{I}}}{\partial v_{l,x}} & \frac{\partial d_1^{\mathrm{I}}}{\partial v_{l,y}} & \frac{\partial d_1^{\mathrm{I}}}{\partial v_d} \\ \vdots & \vdots & \vdots \\ \frac{\partial x_j^{\mathrm{I}}}{\partial v_{l,x}} & \frac{\partial x_j^{\mathrm{I}}}{\partial v_{l,y}} & \frac{\partial x_j^{\mathrm{I}}}{\partial v_d} \\ \frac{\partial y_j^{\mathrm{I}}}{\partial v_{l,x}} & \frac{\partial y_j^{\mathrm{I}}}{\partial v_{l,y}} & \frac{\partial y_j^{\mathrm{I}}}{\partial v_d} \\ \frac{\partial d_j^{\mathrm{I}}}{\partial v_{l,x}} & \frac{\partial d_j^{\mathrm{I}}}{\partial v_{l,y}} & \frac{\partial d_j^{\mathrm{I}}}{\partial v_d} \\ \vdots & \vdots & \vdots \\ \frac{\partial x_l^{\mathrm{I}}}{\partial v_{l,x}} & \frac{\partial x_l^{\mathrm{I}}}{\partial v_{l,y}} & \frac{\partial x_l^{\mathrm{I}}}{\partial v_d} \\ \frac{\partial y_l^{\mathrm{I}}}{\partial v_{l,x}} & \frac{\partial y_l^{\mathrm{I}}}{\partial v_{l,y}} & \frac{\partial y_l^{\mathrm{I}}}{\partial v_d} \\ \frac{\partial d_l^{\mathrm{I}}}{\partial v_{l,x}} & \frac{\partial d_l^{\mathrm{I}}}{\partial v_{l,y}} & \frac{\partial d_l^{\mathrm{I}}}{\partial v_d} \end{bmatrix}$$

According to Equation (A.8), the noise vector is an additive constant. Therefore, the Jacobian matrix $\mathbf{V}$ consists of l Identity Matrices: $\frac{\partial h_j}{\partial \mathbf{v}} = \mathbf{I}$.

Appendix B

Quaternion Rotations

In this work it often occurs, that we need to transform world into camera coordinates. This transformation includes translation as well as rotation. Rotations can be represented by various means. Each representation has specific characteristics and attributes that makes them more or less applicable for solving the SLAM problem. Here, we used *quaternions* to represents rotations due to several advantages. Nevertheless, other representations appear in this work: *Euler Angles*, *Axis & Angle* and *rotation matrices*.

In this appendix we will firstly present the reasons for our choice of quaternions . Then, some general operations with quaternions are introduced. We close with a description how to convert the different means into each other. The following explanations are based on [6] and [24].

B.1 Using Quaternions as a Representation of Rotations

A rotation has three degrees of freedom. These degrees can be conveniently represented by Euler Angles. These are three angles, specifying the rotation about the x, y and z-axis.

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$$

Euler angles are a minimal representation.

Another way to represent rotation would be to specify a rotation axis explicitly and the amount of rotation about that axis. Thus, we need a four-dimensional vector containing a three-dimensional unit vector $\mathbf{x}$ and an angle θ specifying the rotation around $\mathbf{x}$ in radians.

$$\begin{bmatrix} x \\ y \\ z \\ \theta \end{bmatrix}$$

Notations like Euler Angles and Axis & Angle are intuitively easy to understand, but there are problems.

- For some rotations there is not a unique representation using Euler Angles.
- When applying one rotation and then applying another rotation, the sum of the individual rotations is not equal to the total rotation. Thus, Axis & Angle is inapplicable to combine rotations.

Rotation matrices do not suffer from these disadvantages. They usually carry 3×3 elements, where each rotation is represented by a unique matrix. The combination of two rotations is done by matrix multiplication. Nevertheless, there are disadvantages. Since a rotation has only three degrees of freedom, the 9 components of the matrix are constrained by the requirement of orthogonality. When we are representing the rotation of a landmark then we want a matrix that represents a pure rotation, but not scaling, shear or reflections. To do this we need a subset of all possible matrices known as an orthogonal matrices. This implies , e.g. that any two columns must be mutually perpendicular. The determinant of a rotation matrix must be always 1. The magnitude of their eigenvalues must be also equal to 1.

In contrast to that, unit quaternions are four-dimensional vectors, that are just constrained by the requirement, that their magnitude is equal to 1.

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

Each rotation is associated with a unique unit quaternion. Analogous to Axis & Angle, a rotation is represented by the amount of rotation θ around a unit vector $\mathbf{x}$.

$$\mathbf{q} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2})\mathbf{x} \end{bmatrix} \tag{B.1}$$

Thus, both representations are easily convertible into each other, which is explained in detail in Section B.3. The combination of rotations can also easily be done with quaternion multiplication, which is described among other things in Section B.2.

Another advantage of using quaternions is that the Jacobians of the model equations are much easier to derive than with using rotation matrices.

B.2 Basic Quaternion Arithmetic

Antipodal Quaternions As already said above, a unit quaternion describes a rotation about $\mathbf{x}$ by θ. A rotation by $\theta - 2\pi$ yields the same result, but the other way round. According to Definition (B.1), the antipodal rotation to $\mathbf{q}$ can be derived by

$$\begin{bmatrix} \cos(\frac{\theta-2\pi}{2}) \\ \sin(\frac{\theta-2\pi}{2})\mathbf{x} \end{bmatrix} = \begin{bmatrix} \cos(\frac{\theta}{2} - \pi) \\ \sin(\frac{\theta}{2} - \pi)\mathbf{x} \end{bmatrix} = -\mathbf{q}$$

Quaternion multiplication If we have given two single quaternion rotations $\mathbf{q}_1$ and $\mathbf{q}_2$, we can obtain the composed rotation $\mathbf{q}$ by applying

$$\mathbf{q} = \mathbf{q}_1\mathbf{q}_2 = \begin{bmatrix} q_{01} \\ \mathbf{v}_1 \end{bmatrix} \begin{bmatrix} q_{02} \\ \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} q_{01}q_{02} - \mathbf{v}_1 \cdot \mathbf{v}_2 \\ q_{01}\mathbf{v}_2 + q_{02}\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2 \end{bmatrix} \tag{B.2}$$

Quaternion multiplication is an associative but not commutative operation.

To obtain the inverse rotation $\mathbf{q}^{-1}$ to $\mathbf{q}$ we utilise the fact that $\mathbf{q}\mathbf{q}^{-1} = \mathbf{q}^{-1}\mathbf{q} = (1,0,0,0)^\top$. Using Equation (B.2) the inverse rotation to $\mathbf{q} = (q_0, q_1, q_2, q_3)^\top$ is $\mathbf{q} = (q_0, -q_1, -q_2, -q_3)^\top$.

Rotation of a Vector In this work it often occurs that we like to rotate certain points given as three-dimensional vectors $\mathbf{x}$. If we want to apply a quaternion rotation $\mathbf{q}$ than the rotated vector $\mathbf{x}'$ is obtained by

$$\begin{bmatrix} 0 \\ \mathbf{x}' \end{bmatrix} = \mathbf{q} \begin{bmatrix} 0 \\ \mathbf{x} \end{bmatrix} \mathbf{q^{-1}}.$$

B.3 Conversions

Relation between Axis & Angle and Quaternions Each of both representations describes a rotation by θ radians about the unit vector $\mathbf{x}$, the rotation axis. This rotation as Axis & Angle $\mathbf{a}$ is

$$\mathbf{a} = \begin{bmatrix} \mathbf{x} \\ \theta \end{bmatrix}$$

This rotation as quaternion $\mathbf{q}$ is

$$\mathbf{q} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2})\mathbf{x} \end{bmatrix}$$

Relation between Rotation Matrices and Quaternions The columns of a rotation matrix form the base vector of the frame which should be rotated. Thus, the rotation matrix $\mathbf{R}_q$ associated with the quaternion $\mathbf{q} = (q_0, q_1, q_2 q_3)^\top$ can be derived by rotating the columns of a 3×3 identity matrix.

$$\mathbf{R}_q = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_2 q_1 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_3 q_1 - q_0 q_2) & 2(q_0 q_1 + q_3 q_2) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

The conversion of a rotation matrix into a quaternion is quite easy too. Therefore, we define the following abbreviations:

$$\begin{aligned} r_{11} &= q_0^2 + q_1^2 - q_2^2 - q_3^2 \\ r_{12} &= 2(q_1 q_2 - q_0 q_3) \\ r_{13} &= 2(q_1 q_3 + q_0 q_2) \\ r_{21} &= 2(q_2 q_1 + q_0 q_3) \\ r_{22} &= q_0^2 - q_1^2 + q_2^2 - q_3^2 \\ r_{23} &= 2(q_2 q_3 - q_0 q_1) \\ r_{31} &= 2(q_3 q_1 - q_0 q_2) \\ r_{32} &= 2(q_0 q_1 + q_3 q_2) \\ r_{33} &= q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{aligned}$$

As already said above, the sole constrained on quaternions is $\|\mathbf{q}\| = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$. Then we have

$$\begin{aligned}
q_0^2 &= 1 - q_1^2 - q_2^2 - q_3^2 \\
4q_0^2 &= 4 - 4q_1^2 - 4q_2^2 - 4q_3^2 \\
4q_0^2 &= 3q_0^2 + 1 - q_1^2 - q_2^2 - q_3^2 \\
4q_0^2 &= 1 - (-q_0^2 + q_1^2 + q_2^2 - q_3^2) + q_0^2 - q_1^2 + q_2^2 - q_3^2 + q_0^2 + q_1^2 - q_2^2 - q_3^2 \\
4q_0^2 &= 1 - r_{11} + r_{22} + r_{33} \\
&= \\
q_0 &= \pm\frac{\sqrt{1 - r_{11} + r_{22} + r_{33}}}{2}
\end{aligned}$$

The choice of the sign for w indicates the direction of rotation. The other components can be easily derived, too, as shown in the following:

$$\begin{aligned}
r_{32} - r_{23} &= 2q_2q_3 + 2q_1q_0 - 2q_2q_3 + 2q_1q_0 \\
r_{32} - r_{23} &= 4q_1q_0 \\
q_1 &= \frac{r_{32} - r_{23}}{4q_0} \\
q_1 &= \frac{r_{32} - r_{23}}{2\sqrt{1 - r_{11} + r_{22} + r_{33}}}
\end{aligned}$$

$$\begin{aligned}
r_{13} - r_{31} &= 2q_1q_3 + 2q_2q_0 - 2q_1q_3 + 2q_2q_0 \\
r_{13} - r_{31} &= 4q_2q_0 \\
q_2 &= \frac{r_{13} - r_{31}}{4q_0} \\
q_2 &= \frac{r_{13} - r_{31}}{2\sqrt{1 - r_{11} + r_{22} + r_{33}}}
\end{aligned}$$

$$\begin{aligned}
r_{21} - r_{12} &= 2q_1q_2 + 2q_3q_0 - 2q_1q_2 + 2q_3q_0 \\
r_{21} - r_{12} &= 4q_3q_0 \\
q_3 &= \frac{r_{21} - r_{12}}{4q_0} \\
q_3 &= \frac{r_{21} - r_{12}}{2\sqrt{1 - r_{11} + r_{22} + r_{33}}}
\end{aligned}$$

You might have noticed, that there are other possibilities to derive the components of the quaternion $\mathbf{q}$ from a rotation matrix. Starting from the constraint, that the magnitude of the quaternion is equal to 1, we also could evaluate q_1, q_2 or q_3 instead of q_0 and then go on evaluating the respective other three components as shown before.

Bibliography

[1] Fraunhofer AIS. Available via http://www.ais.fraunhofer.de.

[2] Point Grey::Home. Available via http://www.ptgrey.com.

[3] A. Chiuso, P. Favaro, H. Jin, S. Soatto. Structure from Motion Causally Integrated Over Time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4), April 2004.

[4] A. Nüchter. Autonome Exploration und Modellierung von 3D Umgebungen. Master's thesis, Fraunhofer Institut für Autonome Intelligente Systeme, 2002.

[5] A. Wiedemann, A. Wehr. Fusion of Photogrammetric and Laser Scanner Data. In *CIPA Int. Symposium '99, Photogrammetry in Architecture, Archaeology and Urban Conservation*, 1999.

[6] M. Baker. Quaternions. Available via http://www.euclideanspace.com /maths/algebra/realNormedAlgebra/quaternions/.

[7] G. H. Bendels, P. Degener, R. Wahl, M. Körtgen, and R. Klein. Image-based registration of 3d-range data using feature surface elements. In Y. Chrysanthou, K. Cain, N. Silberman, and F. Niccolucci, editors, *5th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST 2004)*, pages 115–124. Eurographics, December 2004.

[8] A. J. Davison. *Mobile Robot Navigation Using Active Vision*. Phd thesis, Robotics Research Group, Department of Engineering Science, University of Oxford, 1998.

[9] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. Technical report, Robotics Research Group, Department of Engineering Science, University of Oxford, 2003.

[10] Frank Dellaert and Ashley Stroupe. Linear 2d localization and mapping for single and multiple robots. In *Proceedings of the IEEE International Conference on Robotics and Automation, 2002*. IEEE, May 2002.

[11] H.J. Ferreau. Einführung in die Automatische Differentiation. Handout, Seminar Automatische Differentiation, 5 2004.

[12] Gary Bishop G.Welch. An Introduction to Kalman Filter, 2001. Siggraph, Course 8.

[13] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, (239):133–135, September 1981.

[14] D. Hähnel, D. Schulz, and W. Burgard. Mobile robot mapping in populated environments. *Advanced Robotics*, 17(7):579–598, 2003.

[15] I.Mahon, S.Williams. Three-Dimensional Robotic Mapping. Technical report, ARC Centre of excellence for Autonomous Systems, School of Aerospace, Mechanical and Mechatronic Engineering, University of Sydney, 2003.

[16] I.N. Bronstein, K.A. Semendjajew, G.Musiol, H. Mühlig. *Handbuch der Mathematik*. Verlag Harry Deutsch, 2001.

[17] R.E. Kalman. An new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[18] T. Luhmann. *Nahbereichsphotogrammetrie*. Herbert Wichman Verlag, 2003.

[19] M.Fichtner, A.Großmann, Michael Thielscher. Intelligent execution monitoring in dynamic environments. Technical report, Knowledge Representation and Reasoning Group, Department of Computer Science, Dresden University of Technology, 2003.

[20] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pages 593–598, 2002.

[21] H. Moravec. *Robot: Mere Machine to Transcendent Mind*, chapter 2:Caution! Robot Vehicle!, page 27. Oxford University Press, 1998.

[22] N. Karlsson, M.E. Munich, L. Goncalves, J. Ostrowski, E. Di Bernardo, P. Pirjanian. Core Technologies for Service Robotics. *Proceedings International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[23] E. Nebot. Simultaneous Localization and Mapping. Australian Centre of Field Robotics, The University of Sydney, July 2002. Summer School.

[24] T. Pietzsch. *Application of a monocular camera as a motion sensor for mobile robots*. Diplomarbeit, Fakulät Informatik, Technische Universität Dresden, 2004.

[25] P.Newman,J.Leonard, J.D.Tardos, J.Neira. Explore and Return: Experimental Valication of Real-Time Concurrent Mapping and Localization. In *IEEE International Conference on Robotics and Automation*, pages 1802–1809, 2002.

[26] P.S.Maybeck. *Stochastic models, estimation and control*, volume 1. Academic Press, 1979.

[27] L.B. Rall. *Automatic Differentiation: Techniques and Applications*. Lecture Notes in Computer Science. Springer-Verlag, 1981.

[28] Smith Randall, Self Matthew, and Cheesman Peter. Estimating uncertain spatial relationships in robotics. In *Proceedings of the 1st Annual Conference on Uncertainty in Artificial Intelligence (UAI-85)*, New York, NY, 1985. Elsevier Science Publishing Comapny, Inc.

[29] R.Klette, K.Schlüns, A. Koschan. *Computer Vision, Three-Dimensional Data From Images*. Springer-Verlag, 1998.

[30] S.Russel, P.Norvig. *Artificial Intelligence-A Modern Approach*. Prentice Hall, 2003.

[31] Hartmut Surmann, Kai Lingemann, Andreas Nüchter, and Joachim Hertzberg. *Aufbau eines 3D-Laserscanners für autonome mobile Roboter*. GMD - Forschungszentrum Informationstechnik GmbH, Sankt Augustin, Germany, März 2001.

[32] Sebastian Thrun. Robotic mapping: a survey. pages 1–35, 2003.

[33] B. Triggs, P. McLauchlan, R. Hartley, and A.Fitzgibbon. Bundle adjustment – A modern synthesis. In *Vision Algorithms: Theory and Practice*, Lecture Notes in Computer Science, pages 298–375. Springer Verlag, 2000.

[34] U. Nehmzow. *Mobile Robotik*. Springer-Verlag, 2002.

[35] Greg Welch and Gary Bishop. Scaat: incremental tracking with incomplete information. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 333–344. ACM Press/Addison-Wesley Publishing Co., 1997.

[36] Peter Whaite and Frank P. Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(3):193–205, 1997.

[37] Michael Ming-Yuen Chang Ying-Kin Yu, Kin-Hong Wong. Recursive 3D Model Reconstruction Based on Kalman Filtering. Technical report, Chinese University of Hong Kong, Hong Kong, 2003.

[38] Michael Ming-Yuen Chang Ying-Kin Yu, Kin-Hong Wong. A fast recursive 3d model reconstruction algorithm for multimedia applications. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, 2004.