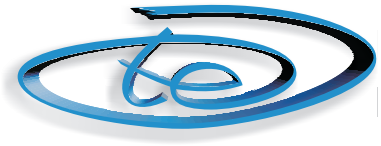
Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät

Lehrstuhl für Technische Elektronik

Prof. Dr.-Ing. Dr.-Ing. habil. Robert Weigel



Diplomarbeit mit dem Thema:

Simultaneous Localization and Mapping for Micro Air Vehicles using a Single Camera and an Inertial Measurement Unit

vorgelegt von

Hauke Stähle
geb. am 10.01.1984
in Aschaffenburg

Betreuer:	Dr. Michael Angermann (DLR) Alexander Götz (LTE)
Ausgehändigt am:	1. November 2009
Einzureichen bis:	3. Mai 2010

Friedrich-Alexander-Universität Erlangen-Nürnberg
TECHNISCHE FAKULTÄT
DLR Deutsches Zentrum für Luft- und Raumfahrt e.V. in der Helmholtz-Gemeinschaft

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde.
Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Hauke Stähle
Erlangen, den 3. Mai 2010

Kurzfassung

Die autonome Navigation von mobilen Maschinen in unbekannten Umgebungen wird als eines der fundamentalsten Probleme in der Robotertechnik angesehen [TBF05]. Um in solchen Umgebungen navigieren zu können, muss eine autonome Plattform die grundlegenden Fragen “Wo bin ich?” und “Wie sieht meine Umgebung aus?” beantworten. Die Antwort wird eine Herausforderung wenn weder globale Referenzen noch vorheriges Wissen der Umgebung zur Verfügung stehen. Eine Lösung zu diesem Problem stellt die Implementierung von Methoden aus dem Bereich “Simultane Lokalisierung und Kartenerstellung” (Simultaneous Localization and Mapping, SLAM) dar. Diese Techniken beziehen sich auf Prozeduren um eine autonome Plattform in die Lage zu versetzen, sich gleichzeitig selbst zu lokalisieren und eine Karte der Umgebung zu erstellen ohne auf vorheriges Wissen zurückzugreifen.

Diese Arbeit analysiert ein SLAM Verfahren für miniaturisierte Flugobjekte (Micro Air Vehicles, MAV), welches auf sequentiellen Bayesschen Filtern basiert. Um Messungen der Umgebung zu erlangen ist eine einzige Kamera auf die Plattform montiert. Die Daten der Bilder werden abstrahiert und benutzt um eine Karte der Umgebung zu erstellen und die Plattform innerhalb dieser Karte zu lokalisieren. Um den Schätzungsprozess weiterhin zu stützen, ist die Plattform mit Inertialsensorik (Inertial Measurement Unit, IMU) ausgestattet, welche die Beschleunigungen und Rotationsgeschwindigkeiten messen kann, welchen die Plattform ausgesetzt ist.

Die Eigenschaften der Fehler der benutzten Sensoren werden analysiert und die entsprechenden Modelle hergeleitet. Ein Verfahren zur Fusion der Messungen wird diskutiert, welches eine Kombination von einem nicht-parametrischen stochastischen Filter (Partikelfilter) für die Positionsschätzung und einem Gauss-basierten Filter (erweiterter Kalman Filter) für die Kartenerstellung benutzt. Die Implementierung des Systems wird durch Experimente ausgewertet und zum erwarteten Verhalten des Algorithmus verglichen.

Abstract

The autonomous navigation of mobile machines in unkown environments is considered to be one of the most fundamental problems in robotics [TBF05]. To be able to navigate in such environments, the basic questions that have to been answered by an autonomous platform are “Where am I?” and “How does my environment look like?”. The answer becomes a challenge if neither global references nor a-priori knowledge of the environment is available. One solution to this problem is to implement methods of simultaneous localization and mapping (SLAM). These techniques refer to procedures where an autonomous platform is put into position to localize itself and map the proximity at the same time without need for previous knowledge.

This thesis analyses a SLAM approach for micro air vehicles (MAV) based on sequential Bayesian filters. To gain measurements of the environment, a single camera is mounted on the platform. The data of the images is abstracted and used to build a map of the environment as well as to localize the autonomous platform within this map. To further support the estimation process, the platform is equipped with an inertial measurement unit (IMU) that measures forces and rotations the platform is exposed to.

The properties and errors of the involved sensors are analyzed and the according models are derived. An approach of merging the measurements is discussed, that uses a combined nonparametric stochastic filter (particle filter) for the position estimation and a Gaussian based filter (extended Kalman filter) for the mapping process. Experiments are evaluated and compared to the expected behavior of the algorithm.

Contents

1 Introduction

The autonomous navigation of mobile machines in unknown environments is considered to be one of the most fundamental problems in robotics [TBF05]. To give a machine the ability to make decisions about its further movement it needs to answer the basic questions “Where am I?”, “How does my environment look like?” and “How to reach my destination?”. This means that the platform needs information about its current location, a definition of the destination it wants to reach and a strategy to combine this data to calculate a route it can follow. In outdoor environments the global positioning system (GPS) can be used to estimate the location of an object with high accuracy as well as to define a destination on the earth. A system following this concept is limited as it relies on a free line-of-sight between the GPS receiver and a certain number of GPS satellites. In situations where the reception is occluded, other approaches have to be used to enable autonomous navigation.

To enable the navigation in environments where no global reference is available, the autonomous platform has to use measurements obtained from internal sensors. With these sensors, the machine can build a map of its proximity and use this to localize itself again in the future. This is referred to as *simultaneous localization and mapping (SLAM)* and helps the autonomous platform to answer the questions “Where am I?” and “How does my environment look like?” without the need for any prior knowledge. It is a typical recursive problem as a localization is needed for the mapping process, and a map is needed for the localization process. SLAM describes the problem of solving both tasks at the same time.

SLAM is strongly related to the behavior of human beings that solve this problem without effort. A typical example is the behavior in an unknown building. When entering an environment for the first time, we intuitively memorize the area. While it may take us a long time to find a certain room on the first attempt, we improve our knowledge about the map of the building every time we reobserve an area. This allows us to orient ourself better with time.

1.1 Motivation

The focus of this work is to solve the SLAM problem for micro air vehicles. A set of such units can be used to deploy a flying network by equipping each of them with wireless communication capabilities. This is useful in disaster scenarios, where an infrastructure is not available and has to be built up quickly to support rescue teams on site.

Furthermore, these units can be used to explore unknown areas without the need for user interaction. With the help of such autonomous platforms, people in danger can be localized or even small payloads can be brought to them in areas where it seems to be

too unsafe for a human to pass. For such scenarios, a flying platform is the first choice as it is not limited by obstacles on the ground.

Both examples have in common, that a micro air vehicle has to gain knowledge about its environment to fulfill its tasks. This is necessary to avoid the contact with obstacles as well as to correct the planned route because of imprecision in the movement and external forces. This might happen as the unit is for example exposed to the influence of wind.

If it is possible to solve the SLAM problem for a micro air vehicle, the gained information can be used to decide about the further movement of the machine and to correct errors between the calculated route and its real trajectory. With an according strategy for the destination choice, such an autonomous platform can be used in civil, disaster and industrial situations for many applications.

1.2 Approach

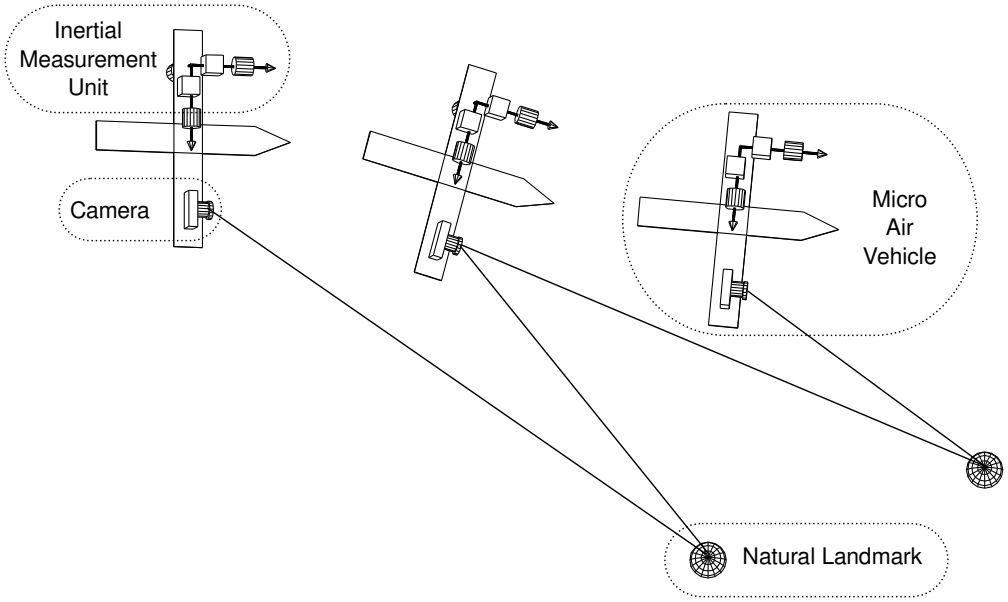


Figure 1.1: Illustration showing the scope of this approach. A micro air vehicle is equipped with a camera and an inertial measurement unit that is rigidly attached to the flying platform. The measurements from both sensors are merged in order to build a map and localize the platform within this map. The camera is used to extract and track natural landmarks that give a locally stable reference.

To obtain measurements of the environment, we use a single camera that captures the visible light and an inertial measurement unit (IMU) that provides information about the acceleration and rotation of the body. Both sensors will be mounted on the micro air vehicle and move with it. In order to solve the SLAM problem, we use a probabilistic

approach that models the mapping and localization process as a combined estimation problem. With the help of Bayesian estimation techniques, we merge the measurements of the camera and the IMU with the momentary estimation of the map to get a stable and accurate system.

While only using the information of the IMU leads to an unbounded growing of the position error, the camera can give a locally stable reference. With the images from the camera, it is possible to identify and to track the relative movement of the platform with respect to points in the environment. The points are used to build the map and to localize the platform relative to its environment. The points for tracking are extracted from the image while the system is operating and thus they are not previously known. This enables the system to work in unprepared environments, without the need to deploy artificial landmarks. As we use a single camera for measurements of the environment we refer to this approach as *visual simultaneous localization and mapping (vSLAM).*

Using a micro air vehicle as platform leads to multiple challenges. The state of the platform is high dimensional, it has to be described by at least six degrees of freedom: Three dimensions for the position and three dimensions for the attitude. This leads to a high computational effort when solving the estimation problem. Another issue involved in this approach is the projective transformation characteristic of the image sensor. As we are only using one camera, it is not possible to estimate the distance of any point in the scenery from only one image. This specific property of a camera has to be considered to gain a stable system.

1.3 Problem Formulation

The aim of solving the SLAM problem is to estimate the posterior of the platform state $\mathbf{x}_{0:k}$ along with the map $\mathbf{m}$ where k is a discrete time index. We assume that the platform is in a static environment and thus the map has no time index because it will never change. The available information to estimate the posterior are the obtained measurements $\mathbf{z}_{1:k}$ and the control inputs $\mathbf{u}_{1:k}$. While the measurements refer to sensor data that includes knowledge about the map, the control inputs usually relates to other internal or external information that is available to improve the estimation process. This includes commands that are sent by an operator to the platform or the controls that are produced by a strategy planer to follow a certain route.

The problem can be formulated as determining a conditional probability density function p:

$$p(\mathbf{x}_{0:k}, \mathbf{m} | \mathbf{z}_{1:k}, \mathbf{u}_{1:k}) \tag{1.1}$$

what is referred to as the *full SLAM problem* [TBF05] as it estimates the whole posterior $\mathbf{x}_{0:k}$ of the states of the platform. Another formulation is called the *online SLAM problem* and it estimates the posterior of only the momentary platform state $\mathbf{x}_k$. The difference is the set of algorithms that can be applied to solve the estimation problems.

The Bayesian network showing the dependencies between the random variables for the full SLAM problem is drawn in figure 1.2. From this diagram, it is obvious that the control input $\mathbf{u}$ is neither dependent on the history of the trajectory nor on the map.

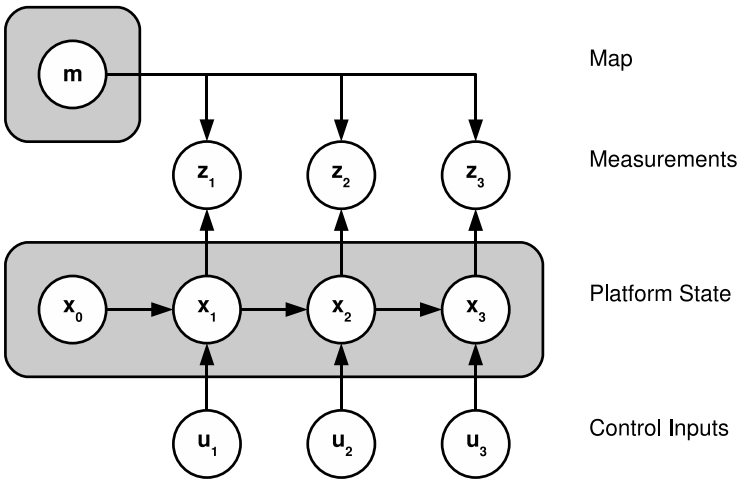


Figure 1.2: Bayesian network showing the dependencies of the involved random variables for the simultaneous localization and mapping problem. The aim is to estimate the posterior of the platform states $\mathbf{x}$ and the map $\mathbf{m}$ without prior knowledge. The measurements $\mathbf{z}$ correlate the map with the platform states while the control inputs $\mathbf{u}$ refer to further information that is available for the state propagation of the platform.

Furthermore, it is visible that the map **m** and the state of the platform **x** is correlated via the measurements **z**.

To solve this problem, basically two models are needed:

- State evolution model
- Measurement model

The state evolution model describes the evolution of the system given the momentary state and control input. It has to take the reliability of the control inputs into account as well as the dynamics of the platform. The measurement model connects the scenery with the internal representation. It has to consider the errors introduced by the sensors and the confidence of the internal map representation. The measurements are used to update the state of the map as well as to localize the platform.

In order to achieve the aim of solving the SLAM problem, a state evolution and a measurement model is required. The models have to reflect the properties of the used sensors with high accuracy in order to get a stable system.

1.4 Contributions

This thesis describes and discusses, based on the work of [MTKW02], the basic theoretical foundations to solve the SLAM problem with the help of a nonparametric filter (particle filter) for the localization estimation and a Gaussian based filter (extended Kalman filter) for the mapping process.

Models are developed to characterize the errors and properties of the inertial measurement unit as well as the camera. A method is derived how to incorporate the individual

measurements to the whole estimation process. Furthermore, the complete image chain is analyzed that is used to identify and to track natural landmarks.

To evaluate the results, measurements were obtained in a tracking system and analyzed with a flexible software framework that was implemented from scratch for this thesis. It is able to visualize and numerically analyze captured data as well as to run in real-time scenarios where the localization information is used for other purposes. This software is supposed to be the basis for future work on this topic.

1.5 Related Work

According to [Thr02], the first statistical framework for simultaneously solving the mapping problem and the induced problem of localizing an autonomous platform relative to its growing map was introduced by a series of papers by Smith, Self and Cheeseman [SSC90][SC86]. From that time on, the problem was referred to as either simultaneous localization and mapping (SLAM) or concurrent mapping and localization (CML).

With further scientific research, other solutions were developed to solve the SLAM problem. These differ in the stochastic approach, the dimensionality of the platform state as well as the used sensors. For this thesis, we will focus on solutions that use bearing-only sensors as primary source for measurements.

An impressive system was developed by Davison [DRMS07] that uses a single camera as sensor. Features are identified in the image by a Harris corner detector and a sparse map of landmarks is built. The contribution of his work lies in the real-time capability of his system. This work was the first to show a solution to the SLAM problem with a single camera that works with the usage of off-the-shelf hardware in real-time. He has chosen to use an extended Kalman filter that represents the landmarks in the map as well as the state of the system. This is a typical approach for SLAM problems. The drawback of this implementation lies in the quadratic complexity with the number of landmarks. This means that this approach is limited to sparse maps with a size of up to some hundreds of landmarks. Furthermore, Davison also shows a solution to the problem of the projective property of a camera by using an inverse parameterization of the distance [CDM08]. With this parameterization it is possible to easily represent a landmark with unknown distance. Also a method was developed to limit the area of the search window for the matching process of features by using a probabilistic approach [CD09]. This approach helps to transform the uncertainty involved in the position of a landmark to a region on the image sensor.

The first implementation of Monte Carlo based methods [MTKW02] for vision based SLAM was introduced by a group in Vancouver [SLL01][SEG05]. They used a stereo camera system from which they extracted visual odometry information and used those as a control input for the actual algorithm. They used very distinctive and transformation invariant SIFT [Low04] descriptors to match the features in the image. The high computational time of extracting those descriptors leads to a system, that was far away from operation in real time. On average ten seconds were spent per frame to extract the features. Nevertheless, the system showed a high accuracy over a long time if used in a post-processing approach.

A group at Linkoeping University investigated a modification of the Monte Carlo based filter to exploit the fact that some parameters are linear within the system model and can be marginalized out of the state space [SGN05]. This leads to a highly efficient system, that is not limited in the dimensionality of the platform state. Usually, the number of needed hypotheses is said to grow exponential with the number of states a system has. This group was able to show, that their method works for a helicopter in an outdoor environment that was equipped with an inertial measurement unit, a barometer sensor and a single camera that was facing downwards [TSKG09]. Although their approach yields only a two dimensional map of the ground, they could show a high accuracy.

Another group from the University of Freiburg uses a similar approach with a unmanned aerial vehicle and two down-looking cameras [SGSB08]. Instead of a Monte Carlo or EKF based implementation, they used Graph-Based SLAM that represents the trajectory of the platform in a network of nodes and edges. The idea is to go back in time and to maximize the certainty of the posterior by alternating the state estimations that the system had in previous time steps.

A different implementation was chosen by a group from the Chemnitz University of Technology [SLP07]. They utilized a flying system with a camera as well but used an unscented Kalman filter (UKF) to incorporate measurements and update the system state. This filter is an extension to the Kalman filter that enables the usage of nonlinear models by calculating the transformation of characteristic points.

1.6 Outline

In chapter 2, the prerequisites are explained that are necessary to follow the derivations within this thesis. This includes the definition of the used coordinate frames and their relations, the representation of rotations and the basic equations needed for two different implementations of Bayesian filters, namely particle filters and extended Kalman filters. The models and characteristics of the sensors are discussed in chapter 3 including the cause and mathematical description of sensor errors and distortions. Chapter 4 is devoted to the processing and abstraction of the image data in order to be able to track natural landmarks. This is comprised of the detection of interesting points on the image plane and the matching with the help of descriptors of those points in order to redetect the natural landmarks in future images. In chapter 5 the integration of the individual components for the visual SLAM approach is introduced. The complete measurement and state evolution models are discussed and the exact strategies related to the map management and landmark parameterization are described in detail. In chapter 6, the individual implementation details are described that were chosen in order to derive a flexible and computational efficient system. The experimental results of chapter 7 show the performance of the system. The influences of different parameters to the entire estimation process is analyzed and evaluated. Chapter 8 concludes this thesis and gives an abstract of further possible improvements of the system.

2 Prerequisites

In this chapter, the basic principles needed to represent the system state are described. This includes the possibilities to express rotations in three dimensional space and the used coordinate systems within the algorithm. Furthermore, two implementations of Bayesian filters are presented that are necessary within our approach to solve the simultaneous localization and mapping problem.

2.1 Euclidean Coordinate Systems

An Euclidean coordinate system is defined by an origin and base axes that are orthogonal to each other to span an n-dimensional space. A point $\mathbf{p}$ within this n-dimensional system $\mathbb{R}^n$ is defined by an n-tuple $\mathbf{p} = (x_1, \cdots, x_n)^T$. The values of this n-tuple are referred to as coordinates of a point within that coordinate system. These coordinates define the length of the individual vectors of the base that have to be combined to define a certain point within the system.

Coordinate systems can be related to each other in the sense that one coordinate system is defined within another one. In this thesis, the latter system is referred to as child system while the other system is referred to as parent system. These systems are also called local system and global system, respectively.

2.2 Rotations

Rotations are transformations of objects that keep one point fixed in two dimensional space $\mathbb{R}^2$ and a line fixed in three dimensional space $\mathbb{R}^3$ while transforming a set of points or vectors. A rotation preserves the length of all rotated vectors and preserves the inner structure of an object e.g. the object under consideration does not change its size.

2.2.1 Rotation Matrix

A rotation can be expressed as a matrix. Each matrix that is orthonormal and has a determinant equal to one defines a rotation matrix. All matrices fulfilling these conditions are in the set of the special orthogonal group $SO(n)$ where n defines the dimensionality.

A rotation of a point $\mathbf{p}^{(A)}$ by a rotation matrix $\mathbf{C}_{AB} \in SO(n)$ to a point $\mathbf{p}^{(B)}$ is described as:

$$\mathbf{p}^{(B)} = \mathbf{C}_{AB}\mathbf{p}^{(A)} \tag{2.1}$$

An important property of rotation matrices is, that they can be combined by left multiplying several rotations:

$$\mathbf{p}^{(C)} = \mathbf{C}_{BC}\mathbf{C}_{AB}\mathbf{p}^{(A)} = \mathbf{C}_{AC}\mathbf{p}^{(A)} \tag{2.2}$$

This means that the point first gets rotated from the system denoted as A to a system B and afterwards to a system C.

As rotation matrices are orthonormal, the inverse transformation equals the transpose:

$$\mathbf{p}^{(A)} = \mathbf{C}_{AB}^{-1}\mathbf{p}^{(B)} = \mathbf{C}_{AB}^{T}\mathbf{p}^{(B)} \tag{2.3}$$

In particular, a rotation matrix for two dimensional space may be written as

$$\mathbf{C} = \begin{pmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{pmatrix} \tag{2.4}$$

and for three dimensional space as

$$\mathbf{C} = \begin{pmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{pmatrix} \tag{2.5}$$

Notice, that the rotation matrix for the two dimensional case has only one degree of freedom while the rotation matrix for the three dimensional case only has three degrees of freedom. This is contributed to the condition that the matrices have to be orthonormal and a determinant equal to one.

2.2.2 Euler Angles

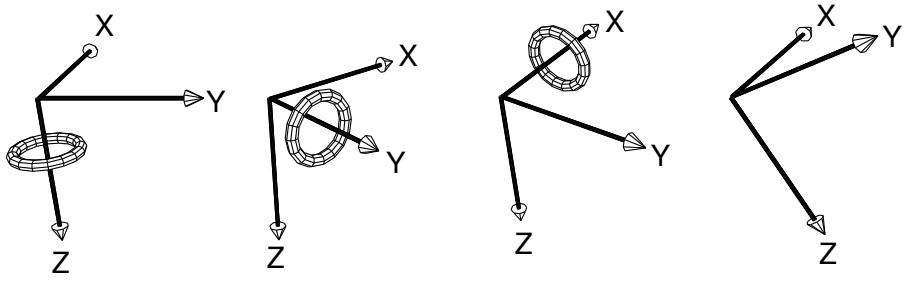


Figure 2.1: Illustration of the definition of the Euler angles used in this thesis. A rotation described by Euler angles is a combination of three individual rotations that are applied one after each other to a coordinate system. In this thesis, the first rotation is a rotation around the z-axis of the system (yaw), the second a rotation around the y-axis (pitch), followed by a rotation around the x-axis (roll).

Euler angles describe a rotation in three dimensional space by three individual rotations around different axes. The angles are called roll, pitch and yaw. Although Euler angles

give an illustrative description of the attitude, care has to be taken not to confuse the different possibilities to describe a rotation with Euler angles. The final attitude of an object using Euler angles depends on: The order of the rotations, the axes of the rotations, whether the axes turn with the object or not, the positive direction of a rotation and the direction of the axes.

The following representation is used in this thesis:

- The axes of the coordinate system are right handed, z-axis is facing downwards
- The order of the rotations is $\mathbf{C}_z$ (Yaw), $\mathbf{C}_y$ (Pitch), $\mathbf{C}_x$ (Roll)
- A positive rotation is a right handed rotation
- The axes are moving with the object

The individual rotation matrices are defined as, where $\cos(x) = \mathrm{c}\,x$ and $\sin(x) = \mathrm{s}\,x$:

$$\mathbf{C}_z(y) = \begin{pmatrix} \mathrm{c}\,y & \mathrm{s}\,y & 0 \\ -\mathrm{s}\,y & \mathrm{c}\,y & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.6a}$$

$$\mathbf{C}_y(p) = \begin{pmatrix} \mathrm{c}\,p & 0 & -\mathrm{s}\,p \\ 0 & 1 & 0 \\ \mathrm{s}\,p & 0 & \mathrm{c}\,p \end{pmatrix} \tag{2.6b}$$

$$\mathbf{C}_x(r) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\,r & \mathrm{s}\,r \\ 0 & -\mathrm{s}\,r & \mathrm{c}\,r \end{pmatrix} \tag{2.6c}$$

The combination of those rotations leads to the final rotation matrix:

$$\mathbf{C} = \mathbf{C}_x\mathbf{C}_y\mathbf{C}_z = \begin{pmatrix} \mathrm{c}\,p\mathrm{c}\,y & \mathrm{c}\,p\mathrm{s}\,y & -\mathrm{s}\,p \\ \mathrm{s}\,r\mathrm{s}\,p\mathrm{c}\,y - \mathrm{c}\,r\mathrm{s}\,y & \mathrm{s}\,r\mathrm{s}\,p\mathrm{s}\,y + \mathrm{c}\,r\mathrm{c}\,y & \mathrm{s}\,r\mathrm{c}\,p \\ \mathrm{s}\,r\mathrm{s}\,y + \mathrm{c}\,r\mathrm{s}\,p\mathrm{c}\,y & \mathrm{c}\,r\mathrm{s}\,p\mathrm{s}\,y - \mathrm{s}\,r\mathrm{c}\,y & \mathrm{c}\,r\mathrm{c}\,p \end{pmatrix} \tag{2.7}$$

This first rotates around the z-axis (yawing) then around the already rotated y-axis (pitching) and finally around the twice rotated x-axis (rolling).

To calculate the Euler angles from a rotation matrix, a direct coefficient comparison is sufficient:

$$\begin{aligned} p &= \arcsin(\mathrm{s}\,p) = \arcsin(-\mathbf{c}_{02}) \\ r &= \arctan 2(\mathrm{s}\,r\mathrm{c}\,p, \mathrm{c}\,r\mathrm{c}\,p) = \arctan 2(\mathbf{c}_{12}, \mathbf{c}_{22}) \\ y &= \arctan 2(\mathrm{c}\,p\mathrm{s}\,y, \mathrm{c}\,p\mathrm{c}\,y) = \arctan 2(\mathbf{c}_{01}, \mathbf{c}_{00}) \end{aligned} \tag{2.8}$$

A special problem with the representation of a rotation with Euler angles is the so called gimbal-lock. This occurs for a certain rotation that leads to a loss of one degree of freedom. For our representation, this happens if the absolute value of the pitch is exactly $\pm\frac{\pi}{2}$. In this situation, the axes for the roll and yaw movement become the same.

Mathematically, this can be seen if setting the pitch value for example to $\frac{\pi}{2}$. This means $\mathrm{c}\,p = 0$ and $\mathrm{s}\,p = 1$:

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & -1 \\ \mathrm{s}r\mathrm{c}y - \mathrm{c}r\mathrm{s}y & \mathrm{s}r\mathrm{s}y + \mathrm{c}r\mathrm{c}y & 0 \\ \mathrm{s}r\mathrm{s}y + \mathrm{c}r\mathrm{c}y & \mathrm{c}r\mathrm{s}y - \mathrm{s}r\mathrm{c}y & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1 \\ \mathrm{s}(r-y) & \mathrm{c}(r-y) & 0 \\ \mathrm{c}(r-y) & -\mathrm{s}(r-y) & 0 \end{pmatrix} \tag{2.9}$$

It is clearly seen, that the z-coordinate of a point is not dependent on the roll and yaw angles anymore in this situation what directly represents the loss of one degree of freedom.

This does not mean that a representation with Euler angles is not sufficient to define the attitude of a flying platform. On the earth, a pitch of $\pm\frac{\pi}{2}$ means that the nose of the vehicle is parallel respectively anti-parallel to the gravity vector. This is an uncommon attitude for any plane.

2.3 Coordinate Frames

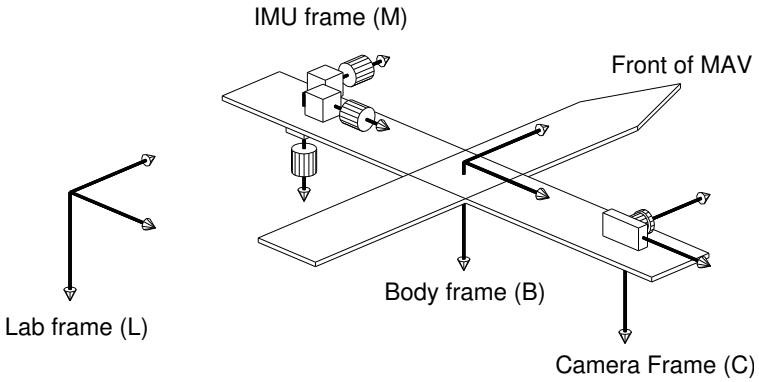


Figure 2.2: Illustration of the used coordinate frames in this thesis and their origin and attitude. The origin of the body frame is at an arbitrary point of the platform. The optical center of the camera and the optical axis define the attitude and origin of the camera frame. The origin of the frame of the inertial measurement unit is in the center of the sensor. The attitude is defined by the alignment of the axes of the sensors. The lab frame is used for a global reference for the mapping and localization process. When the system starts, the body and the lab frames are identical.

Several frames are used in this thesis. A frame defines the origin and attitude of a coordinate system in reference to another system:

- *Lab frame* $^{(L)}$
  The landmarks as well as the position of the platform are defined in the lab frame. The origin of this frame at time $k = 0$ is equal to the origin of the body frame. This is the global frame that our system uses to be able to relate the state of the platform with the map.

- *Body frame* $^{(B)}$
  The body frame is child of the lab frame and has its origin at a certain point on the platform. For a micro air vehicle, this might be the rotational center of the platform. The rotation and translation between the lab and the body frame is referred to as $\mathbf{C}_{LB}$ and $\mathbf{t}_b^{(L)}$.
- *IMU frame* $^{(M)}$
  The IMU frame is child of the body frame and its origin is the center of the acceleration and turn rate sensors. As the body and the inertial measurement unit form a rigid body, the translation between the body and IMU is assumed to be constant over the whole time the system runs. The rotation and translation between the body and IMU frame are written as $\mathbf{C}_{BM}$ and $\mathbf{t}_m^{(B)}$. The letter I was not chosen to name this frame to not confuse it with the common notation I for the inertial frame.
- *Camera frame* $^{(C)}$
  The camera frame is child of the body frame with an origin equal to the projection center of the camera. As for the IMU frame, the translation and rotation between body and camera frame is assumed to be constant and referred to as $\mathbf{C}_{BC}$ and $\mathbf{t}_c^{(B)}$.

The frames are defined as right handed systems. The initial z-axis is facing downwards, for the lab frame this means that the z-axis points in the same direction as the gravity vector.

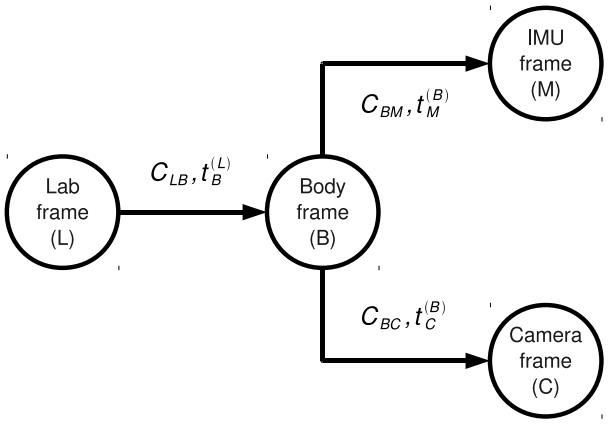


Figure 2.3: Illustration of the relations of the coordinate frames used in this thesis. As the lab frame is the global reference, the body is defined as a child of it. The further split into an IMU and a camera frame allows an easy extension of this definition. The rotation and translation between the body and IMU respectively camera frame is assumed to be fixed as it is a rigid body.

To transfer a point $\mathbf{p}$ from a parent to a child frame, first the translation between the two frames is compensated and afterwards the rotation:

$$\mathbf{p}^{(C)} = \mathbf{C}_{PC}(\mathbf{p}^{(P)} - \mathbf{t}_c^{(P)}) \tag{2.10}$$

where $\mathbf{p}^{(C)}$ and $\mathbf{p}^{(P)}$ refer to a point in the child respectively parent frame, $\mathbf{t}_c^{(P)}$ defines the translation between the child and parent frame in the coordinate system of the parent and $\mathbf{C}_{PC}$ is the rotation between the parent and child system as seen from the parent system.

To transform a point from a child to a parent frame, first the rotation is compensated to align the axes of the child frame parallel to the axes of the parent frame and then the translation between the two frames is added:

$$\mathbf{p}^{(P)} = \mathbf{C}_{PC}^{T}\mathbf{p}^{(C)} + \mathbf{t}_c^{(P)} \tag{2.11}$$

2.4 Bayesian Estimation

Bayesian estimation refers to a probabilistic approach for estimating an unknown probability distribution using measurements. The unknown distribution as well as the available measurements are modeled as parametric or non-parametric distributions. With the help of mathematical models, the individual measurements are used to gain confidence about the unknown distribution.

The scientific usage is the incorporation of incorrect measurements to a believed state of a system. As sensors are subject to certain errors like thermal noise or quantization, the measured values are to some degree incorrect. While directly using these measurements to get a point estimate might lead to wrong results, a Bayesian estimator models the certainty of the measurements to update the state of the system. With each new measurement, the certainty of the whole system state increases even if single measurements are subject to errors.

To use Bayesian estimation, the system state as well as the measurements are modeled as random variables. During the estimation process, the parameters of the associated random distributions of these random variables are updated.

In our case we get noisy measurements from the camera as well as from the inertial measurement unit. With the help of Bayesian estimators it is possible, to calculate the pose of the platform as well as the map of landmarks out of these measurements up to a specific certainty.

2.4.1 Recursive Bayesian Filtering

Before explaining the particular implementations of a Bayesian estimator, the basic equations are derived in this section. For the derivation we will consider a recursive Bayesian filter, that estimates the system state $\mathbf{x}_k$ at the discrete time k using all the available measurements $\mathbf{z}_{1:k-1}$ and control inputs $\mathbf{u}_{1:k-1}$ up to this time. As the system is initialized at time $k = 0$, it is assumed that neither measurements nor control inputs are present for this time step.

The aim is to estimate the posterior

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}, \mathbf{u}_{1:k}) = \widehat{bel}(\mathbf{x}_k) \tag{2.12}$$

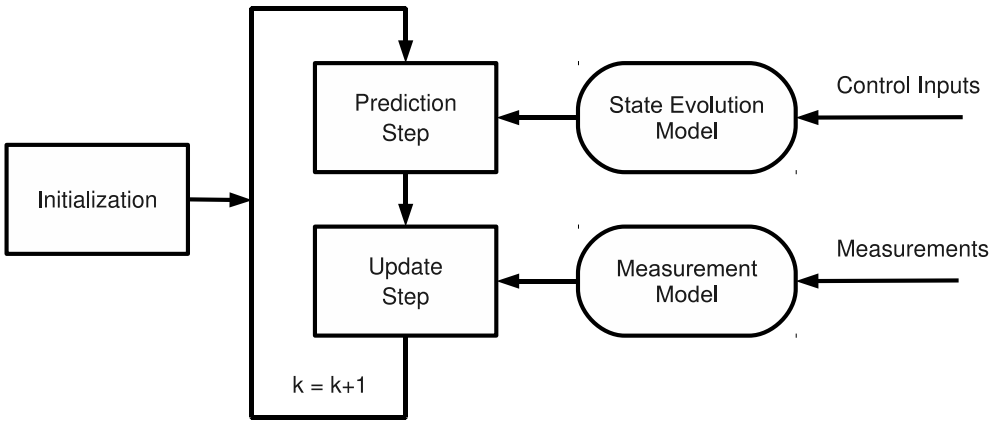


Figure 2.4: Illustration of the basic steps involved in recursive Bayesian filtering. After the random distribution of the state was initialized, the recursive Bayes filter follows a prediction and update scheme. During the prediction step, the distribution is altered according to the dynamics of the system and the available control inputs with the help of the state evolution model. In the update step, the momentary measurements are incorporated into the state distribution with the help of the measurement model.

representing the estimation of the current system state given all measurements and control inputs. For the SLAM problem, the system state $\mathbf{x}$ consists here of the platform state as well as the state of the individual landmarks in the map. For the derivation of the Bayesian filter it is sufficient to model the complete system state with one variable. In later chapters it is shown how to separate this state again in a way to optimize the computational efficiency.

Using Bayes rule, this can be transformed to

$$p(\mathbf{x}_k|\mathbf{z}_{1:k},\mathbf{u}_{1:k}) = p(\mathbf{x}_k|\mathbf{z}_{1:k-1},\mathbf{z}_k,\mathbf{u}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k,\mathbf{z}_{1:k-1},\mathbf{u}_{1:k})p(\mathbf{x}_k|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k})} \quad (2.13)$$

We assume that the system state follows a Markov process. This means that the current system state does only depend on the very last state of the system. Furthermore, the current measurement does not depend on previous measurements nor on the control input.

Thus, we can rewrite the equation to

$$\frac{p(\mathbf{z}_k|\mathbf{x}_k,\mathbf{z}_{1:k-1},\mathbf{u}_{1:k})p(\mathbf{x}_k|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k})} = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k})}{p(\mathbf{z}_k)} \quad (2.14)$$

To calculate an estimation of the current state given all previous measurements, we incorporate the previous estimate as follows:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_{1:k})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k})d\mathbf{x}_{k-1} \quad (2.15)$$

Assuming that the state $\mathbf{x}_{k-1}$ is complete, the current state only depends on the previous state and the current control input $\mathbf{u}_k$. Furthermore, we assume that the control input is randomly distributed and thus does not influence the estimation from the previous step [TBF05]:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k-1})d\mathbf{x}_{k-1} = \overline{bel}(\mathbf{x}_k) \quad (2.16)$$

This step is usually referred to as prediction step yielding a belief $\overline{bel}$ of the system state before incorporating a measurement, this is also referred to as a-priori estimation. In contrast to this, the update step yields the posterior $\widehat{bel}(\mathbf{x}_k)$ that considers the current measurement.

Concluding, the basic structure of a recursive Bayesian filter is given as:

$$\widehat{bel}(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{z}_{1:k},\mathbf{u}_{1:k}) = \alpha p(\mathbf{z}_k|\mathbf{x}_k)\overline{bel}(\mathbf{x}_k) \quad (2.17)$$

A concrete implementation will have to provide three probability distributions:

- The measurement probability $p(\mathbf{z}_k|\mathbf{x}_k)$
- The state evolution probability $p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)$
- The initial belief of the state $bel(\mathbf{x}_0)$

Furthermore, an algorithm has to be chosen that is able to numerically calculate the probabilities. The decision about the algorithm depends on the particular distributions of the random variables and the linearity of the system state and the measurements.

Two different implementations are discussed now, that will be used later for the realization of the visual SLAM system: The Kalman filter and the particle filter.

2.4.2 Kalman Filter

A standard Kalman filter is an implementation of a recursive Bayesian filter and used to calculate the state of a system from noisy measurements. All random variables are modeled as normal distributed and it is assumed that linear models exist for the measurement model as well as for the state evolution model. The original equations were derived in [Kal60].

The special representation of all the random variables limits this filter to process only unimodal distributions. Whenever the variables are distributed in another way, an equivalent Gaussian distribution has to be used what can lead to a wrong estimation process.

Although the standard Kalman filter is not designed to work with nonlinear models, a modified version exists that is described in the next section that is capable of using such models as well.

The state of a Kalman filter can be fully described by a multivariate normal distribution, consisting of a covariance matrix $\mathbf{P}$ and a vector for the mean estimation $\mathbf{x}$. While the Kalman filter is running, the mean estimation gets updated to match the real state in an optimal way. The covariance matrix describes the confidence about the system state.

The state evolution model is described in a discrete state-space representation that relates the system states between two consecutive time steps incorporating the control input:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \tag{2.18}$$

where $\mathbf{x}$ is the state vector, $\mathbf{F}$ the state-transition model that relates the previous state to the current one, $\mathbf{u}$ is a vector of control inputs, $\mathbf{B}$ the control input model and $\mathbf{w}$ is the process noise drawn from a multivariate normal distribution $\mathbf{w} \sim N(\mathbf{0}, \mathbf{Q}_k)$.

The state transition model directly describes the evolution of the system with the assumption that the previous state was perfectly known and the system was not exposed to any influence modifying the state. As a system usually changes its states very frequently, the control input model is used to represent the desired actions. If the system state is always correct and the control inputs perfectly describe the change of the state, the model would already been completed.

But as the system state is not perfectly known and the control inputs are erroneous, additional noise has to be added to reflect this uncertainty. This process noise is used to reflect all the changes of the system, that are not directly covered by the state transition and control input model. This might be caused by an external force that changes the system state or by an erroneous calculation of the control inputs. The latter means that the system might chose to follow a certain path, but it is not able to because of physical limitations.

With this state evolution model, the uncertainty about the system state will always increase as the real system state and the estimation diverge. Although this makes the filter unusable if just using the state evolution model, the uncertainty is correctly represented in the filter state.

To gain further knowledge about the system state, the filter offers the ability to incorporate measurements. These observations are used to update the estimation and will lead to a reduction of the uncertainty of the system. These measurements are only useful, if the system state is observable via them. Although the system state has not to be directly observable, a mathematical model has to exist to map the observations to the internal estimation.

The observations are incorporated with the help of the measurement model:

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{2.19}$$

where $\mathbf{z}$ it the vector of measurements, $\mathbf{x}$ is the system state, $\mathbf{H}$ is referred to as measurement matrix and $\mathbf{v}$ is the measurement noise drawn from a multivariate normal distribution $\mathbf{v} \sim N(0, \mathbf{R}_k)$.

The measurement matrix maps the readings from the sensors to the system state. As the sensors are considered to be imperfect, additional noise is added to the measurements similar to the process noise in the state evolution model. This noise represents the uncertainty that is associated with the readings from the sensors.

According to the derivation of the recursive Bayesian filter, the Kalman filter works in two steps to estimate the system state with the help of the measurements in a recursive way. These steps are namely the prediction step, where the state evolution model is

used to yield an a-priori believe of the system state, and the update step, where the measurements are used to correct the estimate to yield the posteriori estimation of the system state. Before the filter can be used, it has to be initialized to reflect the momentary certainty of the system state.

Prediction Step

During the prediction step of a Kalman Filter, an a-priori believe of the system is derived from the previous state and the control input. This is used as the base for the incorporation of new measurements during the update step.

The equation for the prediction of the estimated system state is

$$\bar{\mathbf{x}}_k = \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \mathbf{u}_k \tag{2.20}$$

what is the direct application of the state evolution model without the random variable for the noise.

As the Kalman filter is a Bayesian filter, also the certainty of the system state is predicted. During the prediction step, the certainty about the system state will decrease, never increase. This is because the prediction is done without measurements. Thus, the state of the system might have changed what is not observed in the prediction step. During the prediction step the uncertainty is increased according to the process noise in order to model such changes:

$$\bar{\mathbf{P}}_k = \mathbf{F}_k \hat{\mathbf{P}}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{2.21}$$

This prediction only reflects the affine transformation of the state vector $\mathbf{x}_{k-1}$ with the state-transition model $\mathbf{F}_k$ and the addition of the random noise variable $\mathbf{w}$. As the control vector is considered to be error-free, no uncertainty is introduced by it. This is of course in practice not true and the uncertainty inherent with the control vector is modeled with the process noise as well.

Update Step

In the update step the predictions of the state vector and the covariance matrix are corrected according to the measurements from the sensors.

The innovation residual defined as

$$\tilde{\mathbf{y}}_k = \tilde{\mathbf{z}}_k - \mathbf{H}_k \bar{\mathbf{x}}_k = \tilde{\mathbf{z}}_k - \bar{\mathbf{z}}_k \tag{2.22}$$

is used to correct the state estimation. As a Kalman filter in its basic form assumes a linear system, the correction is defined to be proportional to the difference between the estimated measurement $\bar{\mathbf{z}}_k$ and the current measurement $\tilde{\mathbf{z}}_k$. It has to be highlighted that $\tilde{\mathbf{z}}_k$ is a true measurement from the sensors while $\bar{\mathbf{z}}_k$ is the prediction of the measurement from the system state.

The innovation covariance defined as

$$\mathbf{S}_k = \mathbf{H}_k \overline{\mathbf{P}} \mathbf{H}_k^\mathsf{T} + \mathbf{R}_k \tag{2.23}$$

is used to find the magnitude of correction. It is an affine transformation of the predicted state covariance $\overline{\mathbf{P}}$ according to the measurement model. Furthermore, the noise covariance $\mathbf{R}_k$ is added to reflect the noisy behavior of the measurements, modeled by $\mathbf{v}_k$ in the measurement model.

For the correction of the state and the covariance, the Kalman gain $\mathbf{K}$ is calculated. The Kalman gain defines the magnitude of correction.

Finally, the state estimate is updated according to

$$\widehat{\mathbf{x}} = \overline{\mathbf{x}}_k + \mathbf{K}_k \tilde{\mathbf{y}} \tag{2.24}$$

and the estimated covariance is updated according to

$$\widehat{\mathbf{P}_k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \overline{\mathbf{P}}_k \tag{2.25}$$

This update of the covariance is only valid for choosing the optimal Kalman gain. This means that the Kalman gain is chosen in order to minimize the mean square error $E((\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T)$. Thus, in that form the Kalman filter is an MMSE estimator and the Kalman gain $\mathbf{K}$ can be calculated as:

$$\mathbf{K}_k = \overline{\mathbf{P}}_k \mathbf{H}_k^\mathsf{T} \mathbf{S}_k^{-1} \tag{2.26}$$

A detailed derivation of the optimal Kalman gain is discussed in [Kal60].

Other definitions of the Kalman gain are possible but not very common. The update of the estimated covariance has to be modified if using another definition.

Initialization

The very first step when using a Kalman filter is its initialization. This refers to an initial estimation of the state vector $\mathbf{x}$ and the covariance matrix $\mathbf{P}$. Initialization is a critical part. If the certainty about the system state is overestimated, then it is very likely that the Kalman Filter will get instable, what means that the estimation of the state does not converge to the real value. An underestimation of the certainty will slow down the converging process. Although this is not a desired situation as well, it is less critical than the previous case.

To initialize a Kalman filter, basically two possibilities exist: Initialization without using any information of the sensors and the initialization using already available measurements.

The initialization without any sensor readings is possible as long as the uncertainty correctly covers all possible states of the system and these states can be modeled as normal distributed. Sometimes, these conditions do not apply and another method has to be chosen. For the case, where measurements are available when starting the system, these can be used to initialize the system. This will help to reduce the variance

of the initial believe and thus lead to a faster convergence of the Kalman filter compared to the initialization process without measurements.

To initialize the system with the help of sensor readings, the inversion of the measurement function can be used:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{H}_k^{-1}\mathbf{z}_k \\ \bar{\mathbf{P}}_k &= \left(\mathbf{H}_k^{-1}\right)^T \mathbf{Q}_k \mathbf{H}_k^{-1} \end{aligned} \tag{2.27}$$

To use this equation, the measurement equation has to be invertible as well as the measurement model and the momentary noise has to be known. Usually, the latter two assumed to be constant and thus do not introduce difficulties.

If the measurement model is not invertible, another function for the initialization has to be developed. This is especially true for the projective transformation of a single image sensor that is unable to estimate the distance of a certain point.

Computational Complexity

A remarkable drawback of the Kalman filter is its almost cubic complexity in the number of states it estimates. The update step involves the inversion of a square matrix with a dimensionality equal to the number of system states. This is necessary in order to calculate the optimal Kalman gain.

This complexity makes the Kalman filter unusable for real-time applications that involve a huge dimensionality of the system state. This especially applies for a SLAM implementation based on a single Kalman filter as the system state consists of the platform state as well as the complete map with a lot of landmarks.

2.4.3 Extended Kalman Filter

The extended Kalman filter modifies the standard Kalman filter to be able to use nonlinear state-space representations for the state evolution model and the measurement model. The functions of both models have to be differentiable, as the extended Kalman filter linearizes about the current mean and covariance with a first order Taylor expansion of the nonlinear functions.

The state-space description of the system changes to

$$\begin{aligned} \mathbf{x}_k &= f(\mathbf{x}_{x-1}, \mathbf{u}_k) + \mathbf{w}_k \\ \mathbf{z}_k &= h(\mathbf{x}_k) + \mathbf{v}_k \end{aligned} \tag{2.28}$$

The state evolution model and the measurement model are now represented by an arbitrary, differentiable function.

Prediction Step

Compared to the basic Kalman filter, the prediction step is modified according to the nonlinear transition function yielding:

$$\begin{aligned} \bar{\mathbf{x}}_k &= f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) \\ \bar{\mathbf{P}}_k &= \mathbf{F}_k \hat{\mathbf{P}}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k \end{aligned} \tag{2.29}$$

where $\mathbf{F}_k$ describes the Jacobian of the state transition function at the current estimate of the system state:

$$\mathbf{F}_k = \left. \frac{\partial f}{\partial x} \right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k} \tag{2.30}$$

Update Step

In the update step, the measurement $\tilde{\mathbf{z}}_k$ is now incorporated according to a nonlinear function:

$$\begin{aligned} \tilde{\mathbf{y}}_k &= \tilde{\mathbf{z}}_k - h(\bar{\mathbf{x}}_k) \\ \mathbf{S}_k &= \mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{R}_k \\ \mathbf{K}_k &= \bar{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{S}_k^{-1} \\ \hat{\mathbf{x}}_k &= \bar{\mathbf{x}}_k + \mathbf{K}_k \tilde{\mathbf{y}}_k \\ \hat{\mathbf{P}}_k &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \bar{\mathbf{P}}_k \end{aligned} \tag{2.31}$$

where $\mathbf{H}_k$ describes the Jacobian of the state transition function at the predicted estimate of the system state:

$$\mathbf{H}_k = \left. \frac{\partial h}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}_k} \tag{2.32}$$

Problems

The extended Kalman filter linearizes the state evolution model and the measurement model with a first order Taylor expansion at the momentary estimation of the system state. This linearization introduces errors in the estimation process that have to be thought of.

The main problem caused by the extended Kalman filter is an overestimation of the certainty of the system state. The linearization leads to an update of the covariance matrix that does not reflect the real certainty. In this case it is very likely that further measurements will lead to an instability of the whole filter.

To overcome this problem, usually the process noise of the system state and the measurement noise are increased to reflect the errors introduced by the linearization process. Some groups even claim that the extended Kalman filter will always fail [BB03], especially in the context of SLAM problems [JU01]. Despite that, the extended Kalman filter has shown its reliability in a huge amount of applications although the estimation suffers from the linearization process.

2.4.4 Particle Filter

The particle filter, also called sampling importance resampling (SIR) filter, is another implementation of a Bayesian estimation filter. Compared to the Kalman Filter, the

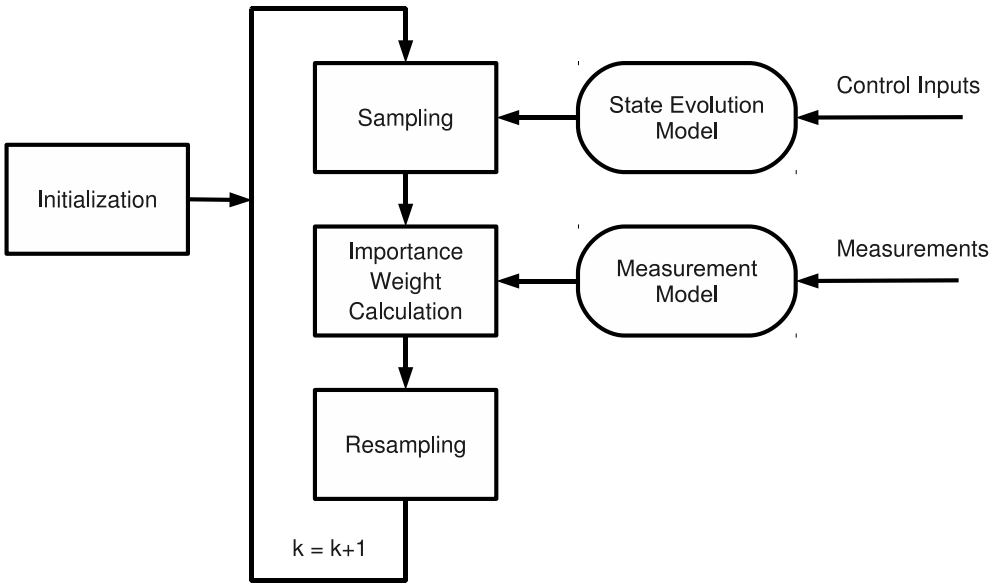


Figure 2.5: Illustration of the steps involved in a particle filter. After the initialization, a set of particles is sampled according to the distribution given by the state evolution model. After that, the importance weight of each particle is calculated with the help of the measurement model. The importance weight describes the ratio between the proposal and the target distribution. In the resampling step, particles with a high importance weight are duplicated while those with a low weight are discarded.

particle filter does not model the random variables as Gaussian distributed. Instead, the particle filter is a nonparametric filter and can model any arbitrary distribution.

The particle filter follows the steps sampling, importance weighting and resampling. In the sampling step, particles are drawn from a proposal function q. After this step, each particle represents a hypothesis of an a-priori state of the system. To incorporate the measurements, the importance weight ω of each particle is calculated. The result is a set of particles with different weights that approximate the posterior probability density function according to:

$$p(\mathbf{x}) \approx \sum_i \omega^{[i]} \delta(\mathbf{x} - \mathbf{x}^{[i]}) \tag{2.33}$$

The resampling step follows. During the resampling step, a new set of particles is created based on the old one. Particles for the new set are selected according to their weight while multiple selection is possible. This means that a hypothesis that does not match the measurement gets discarded while one matching the measurement better gets duplicated.

The approximation is only valid if the number of particles tends to infinity. For practical use, the number of particles is finite and errors are introduced into the model. However, with an appropriate number of particles, the filter is still able to approximate the distribution exactly enough so that it can be used for real applications.

Sampling Step

As the Kalman filter, the particle filter estimates the posterior:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k},\mathbf{u}_{1:k}) = \alpha p(\mathbf{z}_k|\mathbf{x}_k)\int p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k-1})d\mathbf{x}_{k-1} \tag{2.34}$$

what means estimating the state $\mathbf{x}$ at time k given all available measurements $\mathbf{z}_{1:k}$ and control inputs $\mathbf{u}_{1:k}$ up to the momentary time step.

Instead of directly calculating this posterior, the particle filter works with a proposal q and a target distribution p. The proposal distribution usually describes the predicted believe of the system what reflects the estimation of the system state without the measurements. The target distribution is the actual posterior distribution that should be estimated. To describe the relation between those two distributions, a weighting factor ω is introduced that reflects the ratio. The idea behind this split is to be able to use a proposal function that can easily be calculated. Although the proposal usually refers to the prediction, any function can be chosen as long as it is not zero. If choosing a proposal function with zero values, the weighting factors are not able to describe the ratio between the functions anymore. Nevertheless, as the number of particles is limited, the quality of the estimation process will improve if the proposal function describes the system state with high accuracy.

Our proposal function is defined as the estimation according to the state evolution model:

$$q(\mathbf{x}_k|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k-1})d\mathbf{x}_{k-1} \tag{2.35}$$

As the previous posterior $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k-1})$, the proposal distribution is represented as a weighted particle set. This directly means, that the sampling stage can be done by a transformation of the previous hypotheses according to the state evolution model. To reflect the uncertainty introduced by the process noise, a sampled value of the distribution of this noise is added to each particle when calculating the proposal function. Thus, the distribution of the proposal directly reflects the predicted system state including the noise to cover unmodeled changes of the state.

As for the Kalman filter, a state-space representation is chosen to model the system dynamics. This time, the state transition model might be highly nonlinear as the particle filter is a non-parametric filter and thus does not need a model to directly transform the parameters of a distribution.

The state-space representation for the particle filter is given as:

$$\mathbf{x}_k = f(\mathbf{x}_{x-1},\mathbf{u}_k) + \mathbf{w}_k \tag{2.36}$$

where $f(\mathbf{x}_{x-1},\mathbf{u}_k)$ represents a nonlinear state transition model and $\mathbf{w}_k$ is the process noise.

To obtain the proposal distribution, the particle set is directly transformed according to:

$$\forall \mathbf{x}_{k-1}^{[i]} : \mathbf{x}_k^{[i]} = f(\mathbf{x}_{k-1}^{[i]},\mathbf{u}_k) + \mathbf{w}_k \tag{2.37}$$

where $\mathbf{w}_k$ represents one sample of the process noise. Although we haven chosen a Gaussian noise process here, it is possible to incorporate any process as long as it is possible to sample values from it.

Importance Weighting

As the aim is to estimate the posterior and not the proposal function, the ratio between both functions is calculated and used as the weight ω of each hypothesis:

$$\omega_k^{[i]} = \frac{target}{proposal} = \frac{p(\mathbf{x}_k^{[i]}|\mathbf{z}_{1:k}, \mathbf{u}_{1:k})}{q(\mathbf{x}_k^{[i]}|\mathbf{z}_{1:k-1}, \mathbf{u}_{1:k})} = \alpha p(\mathbf{z}_k|\mathbf{x}_k^{[i]})\omega_{k-1}^{[i]} \tag{2.38}$$

where α is a normalization constant. It is obvious, that with this definition of the proposal distribution, the importance weights only depend on the measurement model. If choosing another proposal, this equation has to be altered accordingly. The importance weight from the previous iteration $\omega_{k-1}^{[i]}$ is only necessary if the resampling step was skipped.

Nevertheless, the better the proposal distribution matches the target distribution, the better the estimation process will be in the end. If no particle occupies a place reflecting the true state of the system, it is not possible to approximate the posterior anymore.

Resampling Step

The resampling step is the actual trick of the particle filter. After the evaluation of the hypotheses in the importance weighting step, hypotheses that do not match the measurements are discarded as they do not account for the approximation of the posterior. A particle with the weight zero has no influence on the posterior but as it is part of the fixed size set of particles, the hypothesis is actually lost to represent the state of the system. To overcome this issue, particles that do not contribute to the approximation are discarded while others, that are in a place matching the measurements, get duplicated. This is obvious as many particles should be available in the peak position of the posterior as they very likely reflect the true system state and have to be available to represent a hypothesis of the noise in the further iterations of the particle filter.

Thus, during resampling a new particle set is created out of the old one. Particles are chosen to be member of the new set proportional to their weight. As a particle might be chosen more than once for duplication, this equals an exchange of weight to density. Areas that include particles with a high weight will get a higher density of particles after the resampling while areas with particles with a low weight will have a lower density.

After that, the weights of all particles are normalized according to

$$\sum_i \omega^{[i]} = 1 \tag{2.39}$$

Although the normalization to the constant 1 is only necessary to reflect the properties of a probability density function, any constant might be chosen here. The important fact is only that all particles are set to an equal constant.

Several strategies exist for the resampling step as problems might occur especially if the number of particles is small. As the variance of the system state is reflected by the distribution of the particles, the resampling has to avoid reducing this uncertainty of the state by selecting one particle over-proportional to its weight. This problem is called degeneracy of weights and can happen as the resampling is a random process and one realization might not reflect an optimal distributed selection. Especially with a small number of particles special approaches have to be followed to overcome the degeneration problem.

According to [DC05], four basic resampling strategies are in use:

- Multinomial resampling
  In this method, the particles are stacked in a line and occupy a length reflecting their weight. To form the new set, N samples are drawn from an uniform distribution that spans the area of the stacked particle weights and a particle is chosen every time its interval includes a sample.
- Residual resampling
  For residual resampling, the normalized weights of the particles are multiplied with the fixed size of the set N. This directly means $\sum_i \omega^{[i]} = N$. The modified weights are afterwards split into an integer and a residual part. Each particle is chosen deterministically according to the integer part of its weight. The rest of the particles that are necessary to fill up the complete set are chosen with the multinomial method but by using the residuals of the weights.
- Stratified resampling
  To implement the approach of stratified resampling, the stack of particles is partitioned into a fixed number u of disjoint sets. After that, $\frac{N}{u}$ particles are chosen according to multinomial resampling out of each partition. As $\frac{N}{u}$ might not be an integer number, the missing particles have to be filled up with some strategy.
- Systematic resampling
  Systematic resampling stacks the normalized weights of the particles in a line as well. After that, one sample of a uniform distribution $s = (0 : \frac{1}{N}]$ is chosen, that corresponds to a start value. Beginning from this value, the particles are chosen deterministically with a distance of $\frac{1}{N}$ to yield N particles.

For this thesis, the systematic resampling approach was chosen as it is computational simple and has a good empirical performance [DC05]. Only one random number has to be drawn what makes it faster than the other three approaches. Furthermore, the stack of particle weights is only traversed in one direction what can be exploited to speed up the selection process.

Another important decision is the time of resampling. As the resampling step exchanges weights for density, the actual probability density function is not altered. This means that it is possible to skip the resampling step for some iterations. But the resampling can not be postponed to long, as hypotheses are lost to represent the uncertainty.

The resampling is usually triggered by external as well as internal knowledge. If an autonomous platform does not move, the system state should not change either, what means that the resampling should be stopped. While the platform is moving and the system state changes, the number of hypotheses that actually contribute to the prob-

ability density function has to be observed. If the whole posterior only relies on a few hypotheses, the resampling should be triggered.

The number of effective particles is used as a metric to represent the number of particles that contribute to the posterior:

$$N_{eff} = \frac{1}{\sum_i \left(\frac{1}{\omega^{[i]}}\right)^2} \tag{2.40}$$

This function will go to 1 if only one particle with the normalized weight 1 exists and the others have weight 0. This means that the whole posterior is dominated by just one hypothesis. On the other hand, if all particles contribute equally to the posterior, what means $\forall i : \omega^{[i]} = \frac{1}{N}$, the result of this function will be N. Both these cases represent extreme situations and are not likely to happen in the implementation of a real system. Nevertheless, the values between these extremes give a good estimation about the actual number of particles contributing to the posterior. If N_{eff} drops below a certain threshold, the resampling should be triggered.

3 Sensors and Sensor Models

The camera and the inertial measurement unit are the sensors used in this approach to gain knowledge about the environment and the platform state. As the sensors are physical devices, the measurements are subject to noise and distortion effects. For a properly working system, models are needed to be able to incorporate the raw data that the sensors provide. To increase the accuracy, the models should reflect the behavior of the real devices as good as possible. In this chapter, the basic properties of the used sensors are described and based on those, the according models are developed.

3.1 Camera

We have chosen to use a typical low-cost off-the-shelf camera utilizing a charge-coupled devices image sensor for our approach. The information of the camera is used to detect and track points in the provided images that serve as measurements for the Bayesian filter. A camera can be referred to as a bearing-only sensor as well. As the environment is projected onto the image plane, no distance information is available. In this section, a mathematical model is developed to describe this behavior. Furthermore, the projection is subject to distortions caused by imperfect manufacturing of the camera and lens systems. A method is shown, to derive parameters to be able to describe those effects.

3.1.1 Image Sensor

The image sensor consists of an array of photoactive cells that accumulate an electric charge proportional to the light intensity. Before an image is taken all cells are discharged and after the exposure time the charges are fed into an charge amplifier and the resulting voltage into an analog to digital converter. Usually, the cells are connected in a column-wise fashion and the process of digitalizing the signal happens only in the last column. After the last column was completely digitized, the charges of the remaining columns are all shifted by one towards the last column. The shift register for the charges are called charged-coupled devices (CCD). This process happens until all data was processed and a new image is taken. Although CCD is often used as a synonym for an image sensor it only refers to the underlying process for shifting the charges.

Errors

The values obtained for the individual photoactive cells are subject to quantization and thermal noise during digitalization. As the cells are integrating the light intensity, a long exposure time yields a blurry image dependent on the speed of movement of the

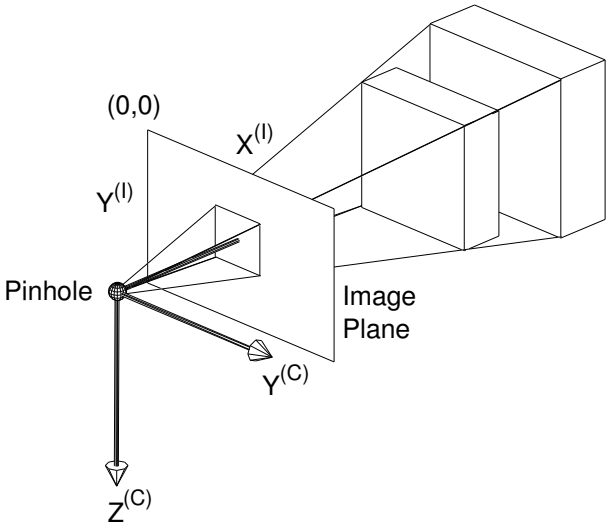


Figure 3.1: Three dimensional illustration of a pinhole camera model. The relation between the coordinate systems of the camera and the image plane are visible. The two boxes result in the same projection on the image plane because of the projective property of a pinhole camera that prevents the discrimination of distances.

camera. The exposure time identifies the amount of time the photoactive cells are exposed to light. The thermal and quantization noise is usually modeled with a normal distributed process.

3.2 Pinhole Camera Model

The pinhole camera model is a basic model to describe the projective transformation of a camera. This model requires assumptions that are not fulfilled by real cameras. Although it is only an approximation it is widely used because of its simplicity and mathematical convenience.

3.2.1 Basic Model

The basic model describes a projection from a point in 3 dimensional space to a 2 dimensional plane, the sensor plane. The basic property of a pinhole camera is the fact, that the rays between all points in 3 dimensional space and their projected points on the sensor plane intersect in one single point, the pinhole or also known as optical center. From every point of the environment, only one ray of light is assumed to pass this optical center. The axis are chosen in a way that the image plane is parallel to the plane spanned by the y- and z-axes of the camera coordinate system. Thus the x-axis, also called optical axis, corresponds to the viewing direction of the camera. The distance between the optical center and the image plane is known as focal length f of the camera while the intersection of the x-axis and the image plane is called the

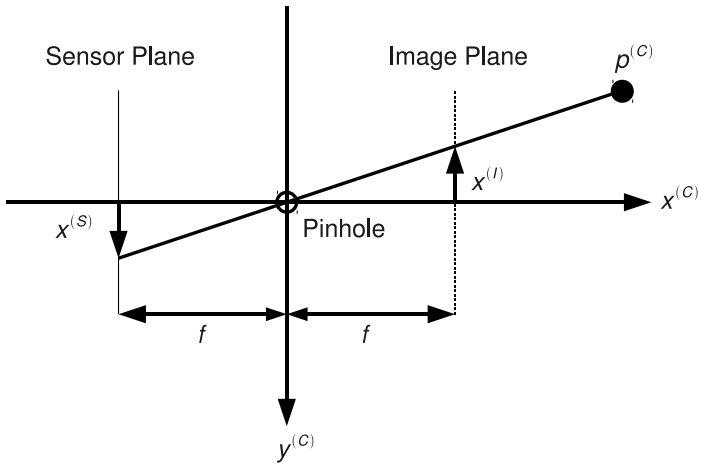


Figure 3.2: Illustration of the projection of the pinhole camera model. The point $p^{(C)}$ in the camera frame gets projected onto the sensor plane and the image plane. The derived coordinates depend on the focal length f that defines the distance between the sensor plane and the pinhole.

principal point. The model only describes the projection of points with a positive x-coordinate in the camera coordinate system, for negative or zero distance of a point the projection is not defined.

The projection of a point in the camera coordinate system $\mathbf{p}^{(C)} = (x_c, y_c, z_c)^T, x_c > 0$ to a point on the sensor plane $\mathbf{p}^{(S)} = (x_s, y_s)^T$ is defined as follows:

$$\begin{pmatrix} -x_s \\ -y_s \end{pmatrix} = \begin{pmatrix} \frac{y_c}{x_c} * f \\ \frac{z_c}{x_c} * f \end{pmatrix} \tag{3.1}$$

A further simplification introduces another plane, the image plane. This plane is parallel to the sensor plane with the same distance, the focal length f, to the origin but intersects with the x-axis at a positive value. The advantage of the image plane is that the projected points onto this plane do not have negative coordinates compared to their counterparts in the coordinate system of the sensor. Thus the projection of the environment onto the image plane will not appear inverted.

The projection of a point $\mathbf{p}^{(C)}$ to a point $\mathbf{p}^{(\bar{I})} = (x_{\bar{i}}, y_{\bar{i}})^T$ on the image plane without considering the rasterization is given as:

$$\begin{pmatrix} x_{\bar{i}} \\ y_{\bar{i}} \end{pmatrix} = \begin{pmatrix} \frac{y_c}{x_c} * f \\ \frac{z_c}{x_c} * f \end{pmatrix} \tag{3.2}$$

As it makes the mathematical derivation more compact and easier to follow, the projection onto the image plane is chosen in this thesis to be the place where the capturing of the images happens for further processing. In this context it has to be mentioned, that this image plane is a mathematical concept and the actual sensor lies on the sensor plane. Because of this fact, the image plane is sometimes referred to as virtual image plane as well.

Pixel Space

The basic model describes a projection for all points with a positive x-coordinate x_c. As the physical size of an image sensor is limited, some points with a positive x-coordinate will not get captured by the sensor. Furthermore, the projection onto the image sensor is rasterized and translated.

An image sensor consists of an array of photoactive cells, that can capture the emitted light of objects. The rasterization occurs as the light sensitive entity at an image sensor has a physical size. Every cell yields a single point, called pixel, that contributes to the rasterized picture of the projected environment. The number of columns and rows are called the resolution of the image sensor and referred to as res_X and res_Y

In computer vision, the coordinates of pixels are usually positive. To satisfy this condition, the coordinates of the projected points are translated that the position of the pixel located at the top and left corner equals $\mathbf{p}_{TL} = (0,0)^T$ and the position of the pixel in the down and right corner equals $\mathbf{p}_{DR} = (res_X - 1, res_Y - 1)^T$. This translation is modeled by the focal offset factors f_{ox} and f_{oy}.

Given a point in the camera coordinate system $\mathbf{p}^{(C)} = (x_c, y_c, z_c)^T, x_c > 0$, the pixel coordinates $\mathbf{p}^{(I)} = (x_i, y_i)^T$ can be calculated as follows:

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \left(\frac{y_c}{x_c} * f + f_{ox}\right) * \frac{res_X}{size_X} \\ \left(\frac{z_c}{x_c} * f + f_{oy}\right) * \frac{res_Y}{size_Y} \end{pmatrix} \tag{3.3}$$

The factors for resolution, sensor size and focal length can be combined to a single factor what leads to a more compact representation. The mapping from meters to pixels in that representation is already included in the factors for the focal pixel lengths $f_{x,px}$, $f_{y,px}$ and the focal pixel offsets $f_{ox,px}$, $f_{oy,px}$. As the image sensor is usually not quadratic, two different focal lengths are now necessary:

$$x_c \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \begin{pmatrix} f_{ox,px} & f_{x,px} & s_{px} \\ f_{oy,px} & 0 & f_{y,px} \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \tag{3.4}$$

The involved matrix $\mathbf{K}$ for the projection is called intrinsic camera calibration matrix. With the help of this, the projection process can be represented in a mathematical convenient way. The result are so-called homogeneous coordinates that represent a line. This can be easily seen as an alteration of the coordinate x_c does not invalidate the equation. To actually calculate the according pixel on the image plane, the division by the distance is inevitable. This operation is called perspective divide or homogeneous divide. Notice, that the coordinate x_c actually does not directly reflect the distance of the point to the projection center. To obtain the distance, the other coordinates have to be taken into account as well.

The additional parameter s of the intrinsic camera calibration matrix $\mathbf{K}$ describes one type of distortion the image is exposed to. It is referred to as skew parameter and models not orthogonally aligned sensor cells.

Unit Plane

Another important plane in the model of the pinhole camera is the so-called unit plane. This is a plane parallel to the image plane, that intersects with the optical axis at a distance of 1 to the pinhole. For some undistortion models, the point has to be represented at this plane.

Inverse Projection

Inverse projection reverses the calculation of a point in three dimensional space $\mathbf{p}^{(C)} = (x_c, y_c, z_c)^T, x_c > 0$ from the pixel coordinates on the image plane $\mathbf{p}^{(I)} = (x_i, y_i)^T$. The mapping is not directly possible, because it is a transformation of a point from three degrees of freedom to two degrees of freedom. Nevertheless, an inverse projection is possible up to an unknown x-coordinate of the point $\mathbf{p}^{(C)}$.

The inverse projection given a known coordinate x_c is:

$$\frac{1}{x_c}\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ \frac{1}{f_{x,px}} & \frac{-s}{f_{x,px}f_{y,px}} & \frac{sf_{oy,px}}{f_{x,px}f_{y,px}} - \frac{f_{ox,px}}{f_{x,px}} \\ 0 & \frac{1}{f_{y,px}} & -\frac{f_{oy,px}}{f_{y,px}} \end{pmatrix}\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{3.5}$$

3.2.2 Geometric Image Distortion

The calculated positions of projected points on the image plane are subject to distortions. A distortion means that a point is influenced by a shift to the position it should have according to the basic pinhole camera model. Distortion in this context only refers to a geometrical alteration of the rays and does not include effects that change the energy of the signal like fog or clouds.

Several sources lead to such distortions: Inaccuracies in the manufacturing of the image sensor, inaccuracies during attachment of the image sensor to the camera chassis and distortions introduced by imperfect lens systems. Furthermore, a distortion might also be introduced because the ray of light passes different kind of media on its way to the camera that can include disturbances introduced by heat sources or refraction caused by glass. The latter distortions are not further considered in this thesis as it is assumed, that the environment does not include conditions that lead to those distortions. Nevertheless, these have to be considered if working in according areas.

The first kind of error is introduced by the mounting and manufacturing of the image sensor. The pixels of the image sensor might be subject to an error in the orthogonal alignment, that lead to a skew transformation on the image plane, represented by the parameter s in the camera calibration matrix. During attachment of the image sensor to the camera chassis a misalignment might be introduced as well. The effect is, that the image sensor is not centered in the chassis and might be tilted in a way, that it is not orthogonal to the optical axis anymore.

Another source of distortion is introduced by the lens system as the individual optical elements might be shifted and tilted towards each other. The lens system mentioned here

is a difference to the introduced pinhole camera model. Real cameras are equipped with several kind of lenses that make it possible to bundle the light rays emitted from a certain point in the environment. This makes it possible to reduce the exposure time what first allows the real-time usage of cameras. The reduction of the exposure times also directly avoids blurry images as the image sensor does not move much during capturing of one image.

Like proposed by [WCH92], the distortions excluding the image sensor can be categorized into three groups:

- Radial distortion caused by the lens system
- Decentering distortion caused by the misalignment of the optical centers of individual elements of the lens system
- Thin prism distortion caused by imperfections during manufacturing of the lens system and the misalignment of the CCD sensor within the camera chassis

Although models for each of these groups exists, [WCH92] could show that the distortion is governed by the offset of the principal point on the image sensor and a radial distortion introduced by the lens system. For our implementation, the distortion model combines a second order radial distortion and the distortion of the principal point.

The shift of the principal point from its ideal position can be modeled by a change of the focal offset factor. The radial distortion model depends on the distance of a projected point to the principal point.

Second Order Radial Distortion

Given an undistorted point $\mathbf{p}^{(U)}$ at the unit plane and considering the first coefficient for radial distortion k_1, the dependence can be described according to [WCH92]:

$$\tilde{\mathbf{p}}^{(D)} = \mathbf{p}^{(U)}(1 + k_1||\mathbf{p}^{(U)}||^2) \tag{3.6}$$

where $||\mathbf{p}^{(U)}||$ describes the Euclidean distance of the considered point on the unit plane to the principal point and $\tilde{\mathbf{p}}^{(U)}$ is the distorted point, that includes the shift caused by the radial distortion. This kind of distortion happens before the projection to the image plane. These equations describe the distortion for a point on the unit plane what makes it independent of the physical geometry of the image sensor.

The analytical inverse of this equation leads to several solutions, that are computationally complex. Usually an approximation is used to undistort a point. As proposed by [MW04] the undistortion can be approximated by:

$$\mathbf{p}^{(U)} = \tilde{\mathbf{p}}^{(U)} \left(1 - \frac{k_1||\tilde{\mathbf{p}}^{(U)}||^2 + k_1^2||\tilde{\mathbf{p}}^{(U)}||^4}{1 + 4k_1||\tilde{\mathbf{p}}^{(U)}||^2} \right) \tag{3.7}$$

3.2.3 Calibration Process

To estimate the parameters of a lens-camera system, a calibration procedure is needed. The estimated parameters usually include the five intrinsic parameters needed to form

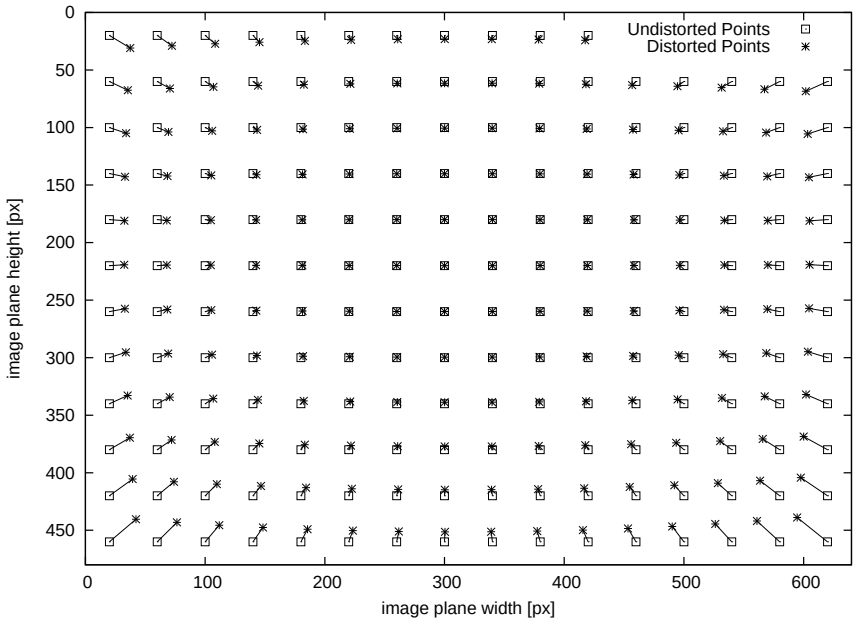


Figure 3.3: Illustration of the radial distortion caused by the lens system of the camera used in this approach. The radial distortion depends on the distance of a point from the principal point and causes the projection of a point from the environment to shift compared to an ideal pinhole camera model.

the camera calibration matrix $\mathbf{K}$, which are the focal lengths $f_{x,px}$, $f_{y,px}$, the focal offsets $f_{ox,px}$, $f_{oy,px}$ and the skew parameter s. Furthermore, the parameters needed for modeling the distortion of the lens are estimated.

Within this thesis, existing software was used to estimate the calibration parameters. Applications for the calibration process include the Calibration Toolbox for Matlab, CalDe and CalLab from DLR and the Calibration Application of OpenCV. All these tools work in a similar way. First, a test pattern has to be printed whose dimensions and shape are known by the tool. In a next step, several images showing this test pattern under different angles and distances are taken. The software identifies the points of the test patterns on the images and afterwards estimates the orientation and translation of the camera in reference to this pattern. These are called the extrinsic parameters of the camera. After that, the measured points can be compared to the projected points according to a pinhole camera model with distortions. In the last step, the image error is minimized considering the parameters of this model. The image error is defined as a cost function that relates the distances of the measured points to the projected ones.

3.3 Inertial Measurement Unit

An inertial measurement unit is equipped with one or more sensors to measure accelerations and turn rates of objects. The aim is to be able to calculate the attitude, velocity and position of an object at every time. To be able to calculate those in three dimensional space, three acceleration as well as three turn rate sensors are needed.

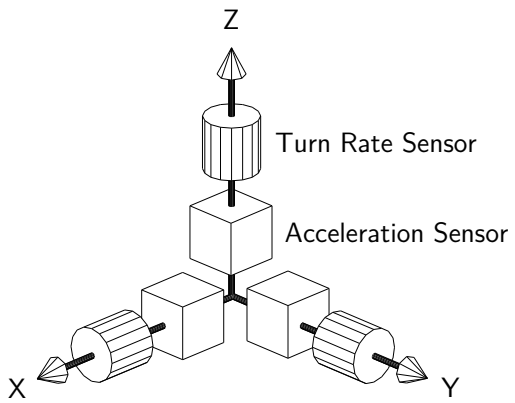


Figure 3.4: Illustration of a strapdown inertial measurement unit. The unit consists of three turn rate sensors and three acceleration sensors that are orthogonal to each other. The measurements are used to describe the dynamics of an object moving and rotating in three dimensional space.

Two types of inertial measurement systems are distinguished: Stable platform systems that keep their attitude while an object is moving and strapdown systems that rotate with an object. While stable platform systems offer a longer stable estimation of the attitude, the rotation of the body has to be compensated what makes these systems mechanically more complex. For the experiments described in this thesis a strapdown system was used. The measurements of those systems reflect the change of the already rotated attitude of the body they are attached too. To be able to calculate the attitude and position in reference to a frame outside the body, a strapdown algorithm is required that keeps track of the attitude, velocity and position of an object and is able to incorporate new measurements.

3.3.1 Turn Rate Sensors

The turn rate sensors measure the velocity of a rotation around a specific axis. Several types for the realization exist, that include micro-electro-mechanical systems (MEMS), systems based on optical effects and systems based on mechanical effects. Other systems are not described here, a complete overview can be found in [Tit05].

Mechanical based systems actually do not measure turn rates, but the orientation of an object. They are built with a set of three gimbal rings, that allow free movement around all axes, and a spinning wheel in the center. These kind of systems are not part of a strapdown system but mentioned here for completeness. When the housing of this system rotates, the spinning wheel will resist to follow this movement because of the effect of conservation of angular momentum. Thus, the absolute change of the orientation in reference to the starting attitude can be directly measured from the angles between the gimbal rings. The drawback of those systems are the fact that rotating parts are involved as well as the relatively large size of such a system.

Optical based systems are using lasers to measure turn rates. The basic idea is to emit two similar beams of a laser at the same point within a circular loop in opposite directions. When the beams exit the loop again, they are combined and if the loop

was undergoing a rotation, a phase shift is measurable as one beam traveled a longer way compared to the other one. This behavior is known as Sagnac-Effect. Although optical based systems work reliable and accurate, they are expensive and a realization occupies a relatively large area.

The drawbacks of the optical and mechanical based systems make them unusable for micro air vehicles. Thus, micro-mechanical systems are used. They are cheap and lightweight on the one hand, but on the other they also provide the worst quality of the measurements.

MEMS turn rate sensors make use of the Coriolis effect. This effect states that a mass m moving with velocity $\mathbf{v}$ experiences a force if the frame of reference is rotating at an angular velocity ω [Woo07]:

$$\mathbf{F}_c = -2m(\boldsymbol{\omega} \times \mathbf{v}) \tag{3.8}$$

This force can be measured and used as a basis for the calculation of the turn rates. As this effect is only present for moving masses, the micro-mechanical electrical system uses vibrating elements to be able to make usage of this effect.

Error Characteristics of MEMS Turn Rate Sensors

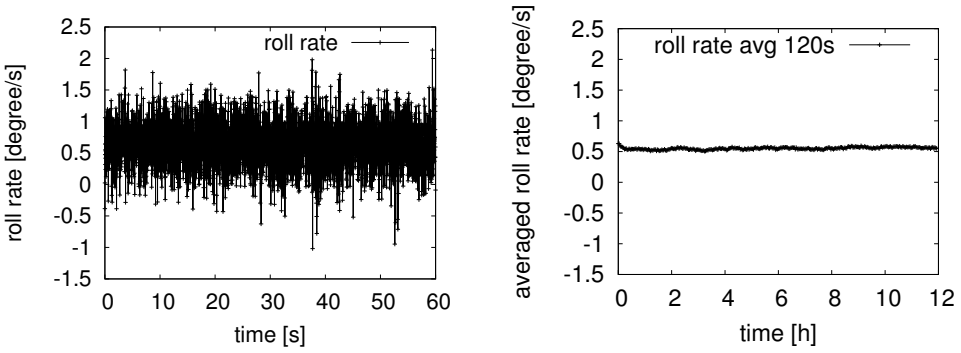


Figure 3.5: Two plots showing the sensor noise respectively the evolution of the bias for the turn rate sensor of the x-axis at the same scaling. The first plot shows the individual samples within a one minute window. The second one shows the evolution of the bias by dividing the captured data for 12 hours into bins of each 120 seconds and calculating the average of each bin. It is clearly visible that the measurements are dominated by the sensor noise and the constant part of the bias. Nevertheless, it is also visible that the bias is moving slowly especially among the first values of the plot. This is due to the temperature change of the inertial measurement unit that was not warmed up when starting the capture.

Considering three MEMS turn rate sensors, that are supposed to be orthogonally aligned, a generic error model is [Wen07]:

$$\tilde{\omega} = \mathbf{M}_{Gyro}\omega + \mathbf{b}_{Gyro} + \mathbf{n}_{Gyro} \tag{3.9}$$

where $\tilde{\omega}$ describes a vector of the measured turn rates while ω are the true turn rates. The summands $\mathbf{b}_{Gyro}$ and $\mathbf{n}_{Gyro}$ model induced errors and are referred to as bias respectively sensor noise. The matrix $\mathbf{M}_{Gyro}$ is called the misalignment matrix of the turn rate sensor.

The misalignment matrix $\mathbf{M}_{gyro}$ describes the misalignment of the turn rate sensors from a perfect orthogonal alignment. Furthermore, scaling factors s are included in this matrix that map the measurements from the sensors to usable values. The misalignment of the axes is assumed to be constant and usually already considered by the firmware of the inertial measurement unit. The manufacturer estimates those parameters and stores them into non-volatile memory. Before the data of a measurement is sent, the readings are corrected for misalignment errors. The scaling factors are split into a constant and a dynamic part. While the constant part is considered by the calibration process of the manufacture as well, the dynamic part models errors introduced by non-linearities of the analog to digital converter and depends on the momentary turn rate.

The bias is split into a constant, a drifting and a residual part. The constant part $\mathbf{b}_{const}$ is assumed to be static among a measurement. The drifting part $\mathbf{b}_{drift}$ models the slow changes of the bias that will lead to a random walk in the bias. This effect is caused by flicker noise in the electronics [Woo07]. The residual bias $\mathbf{b}_{residual}$ models the influence of the temperature to the MEMS. Notice, that the dependence between the temperature and the drift is highly non-linear. Some sensors already include a rough compensation for this temperature effect.

The sensor noise is caused by thermo-mechanical effects that influence the readings from the sensors. This noise fluctuates at a higher frequency than the sampling rate and thus will lead to a white noise sequence that influences the values of the individual samples. The integration of those values will lead to an angular random walk in the end.

The most important error sources for MEMS turn rate sensors are the white thermal noise and the constant bias. The error in the misalignment can usually be neglected. To model the errors, we have chosen to use a Gaussian white noise process as a representation for the influence of the sensor noise. The constant bias is estimated before each experiment by averaging measurements over several samples.

3.3.2 Acceleration Sensors

Acceleration sensors measure the specific forces an object is exposed to. They include mechanical based systems and solid state systems. These are examples, a more completed overview can be found in [Tit05].

Mechanical systems consist of a mass that is attached in a flexible way to a holder. Because of the inertia effect, this mass resists immediately changes of the position. This effect can be measured and is directly proportional to the force the sensor was exposed to. Newton's second law models this behavior as it says, that the acceleration of an object is direct proportional to the exposed force.

Solid state based systems use different effects to measure the acceleration. Among others, they utilize the properties of quartz elements, acoustic waves or vibrating el-

ements. As an example, the vibrating beam accelerometer is described here. This sensor consists of masses that are attached to two quartz elements in a way that an acceleration will cause one quartz element to be stretched while the other one is bended. The oscillation frequency of the quartz elements will change according to the force the element is exposed to. This change can be measured and is a direct indication for the force the sensor was exposed to.

No differences in the principles between full-scale systems and micro-mechanical acceleration sensors exist. They exploit the same techniques to derive acceleration measurements.

Inherent to all acceleration sensors is the fact, that the specific force measurements include the gravity. Depending on the attitude of the sensor, the gravity influence on the axes of the acceleration sensor is different. This effect has to be compensated in order to gain measurements that only represent the accelerations the object was exposed to.

Error Characteristics of MEMS Acceleration Sensors

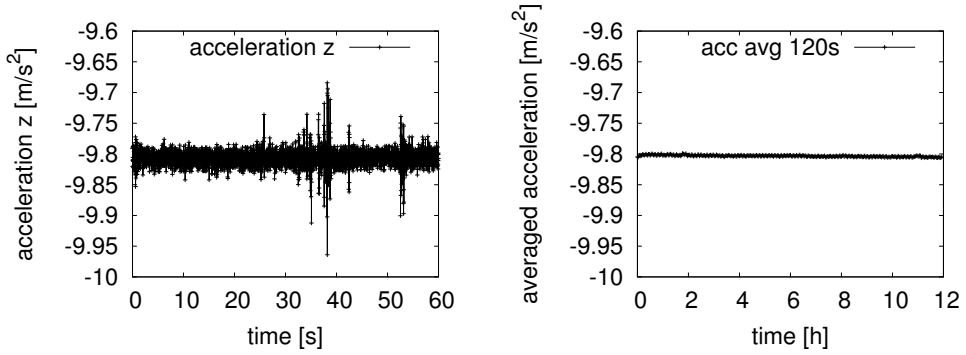


Figure 3.6: A plot similar to the plot of the turn rate sensor showing the sensor noise and bias evolution of the acceleration sensor for the z-axis. As this axis was oriented parallel to the direction of the gravity vector, the resulting force was measured by the sensor. Although the direction of the gravity vector and the acceleration sensor are the same, the force is seen as a negative acceleration. As for the turn rate sensor, it can be seen that the measurements are dominated by the sensor noise.

A generic error model for the acceleration sensors follows the same structure as for the turn rate sensors [Wen07]:

$$\tilde{\mathbf{a}} = \mathbf{M}_{Acc}\mathbf{a} + \mathbf{b}_{Acc} + \mathbf{n}_{Acc} \tag{3.10}$$

where $\tilde{\mathbf{a}}$ represents the measured acceleration, $\mathbf{a}$ is the true acceleration, $\mathbf{M}_{Acc}$ is the misalignment matrix of the acceleration sensors, $\mathbf{b}_{Acc}$ is the bias and $\mathbf{n}_{Acc}$ represents the thermal sensor noise.

The induced errors are similar to those of the turn rate sensors. Instead of an angular random walk, the integration of the readings from an acceleration sensor will lead to a random walk in the velocity respectively a second order random walk in the position.

A special error source related to the acceleration sensors is called size effect. This effect describes the translation between the turn rate sensors and the acceleration sensors. Because of the finite dimensions, it is not possible that both sensors share a common origin. Thus, the acceleration sensors will measure a force, if the sensor is spinning around the axes of the turn rate sensors. An integration of the measurements in this case will result in a pseudo-velocity of the object. For our approach, we neglected this effect, as the contribution to the whole error is small.

3.4 Strapdown Inertial Navigation Algorithm

Sensors of an inertial strapdown system are aligned in orthogonal axes and measure the acceleration or rotation towards respectively around these axes. The strapdown system is directly connected to the system that has to be tracked and forms a rigid body with that system. A strapdown algorithm incorporates the measurements of an inertial measurement unit that changes its attitude with the body to yield the attitude, position and velocity within a global reference frame.

For the interpretation of the measurements from the inertial measurement unit, we assume that the earth is flat within the range of the platform. This is an approximation and only true for small distances. This also includes that no reference coordinate system is present that rotates what leads to a simplification for the derivation of the strapdown algorithm.

The strapdown inertial navigation algorithm is split into one part calculating the attitude of an object and into one part calculating the position and velocity of an object. To obtain the attitude, the readings of the turn rate sensors are integrated. This orientation is used to project the data from the acceleration sensors to global axes where the gravity is compensated. The remaining force, that describes the accelerations of the object in a gravity-free world, are integrated to gain the velocity and integrated again to get the position. The algorithm has to be initialized with start values for the orientation, velocity and position.

3.4.1 Attitude Calculation

The attitude calculation is one part of the strapdown algorithm. As the axis of the IMU are moving with the object, a direct integration of the turn rate is not sufficient. Instead, the turn rates describe the rotational velocity of the already rotated body. This means, that the axes for the measurement change with the orientation of the object. This fact has to be considered during the integration process.

Attitude Computation with Euler Angles

To be able to use Euler angles for the calculation of the attitude, a relation has to be found, that describes how the Euler angles change if rotating the axes of an already

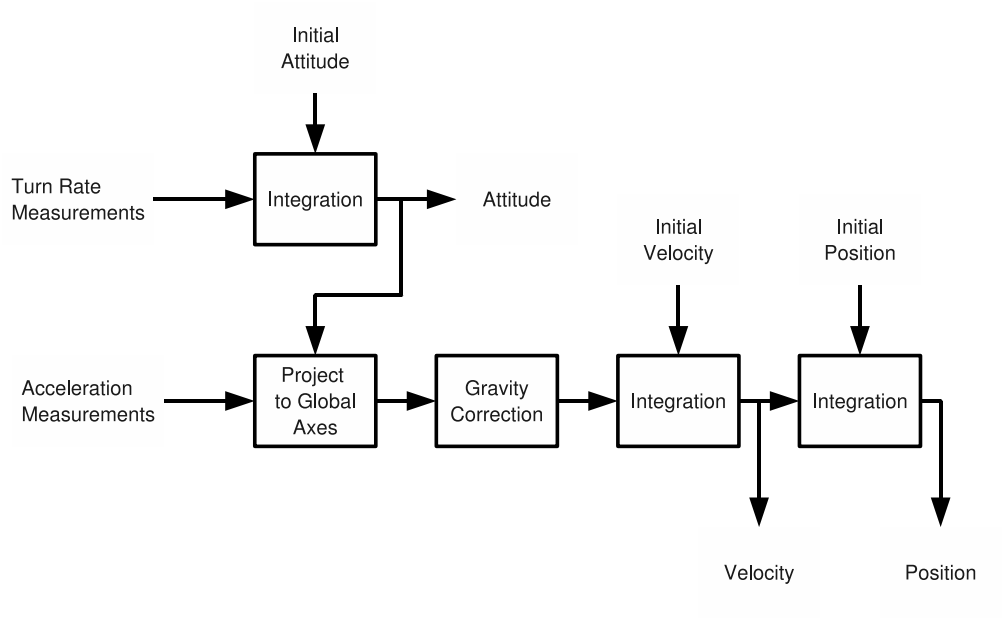


Figure 3.7: A basic strapdown algorithm. While the final attitude can be calculated by integrating the values of the turn rate sensors, the acceleration measurements have to be transformed to a global reference. This is necessary as the acceleration sensors measure the gravity which has to be compensated to derive the true acceleration of the body. The acceleration of the body is integrated once to get the velocity and integrated twice to get the position.

rotated system what is measured by the turn rate sensors. It is obvious that it is not sufficient to integrate these values as the axes of the rotation differ.

The mapping of turn rates ω to changes of Euler angles can be described with an differential equation [Tit05]. This description utilizes the turn rate transformation matrix $\mathbf{E}(r,p,y)$:

$$\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix} = \begin{pmatrix}\dot{r}\\ 0\\ 0\end{pmatrix} + \mathbf{C}_x \begin{pmatrix}0\\ \dot{p}\\ 0\end{pmatrix} + \mathbf{C}_x\mathbf{C}_y \begin{pmatrix}0\\ 0\\ \dot{y}\end{pmatrix} = \mathbf{E}(r,p,y) \begin{pmatrix}\dot{r}\\ \dot{p}\\ \dot{y}\end{pmatrix} \tag{3.11}$$

Notice that this equation is only valid for our definition of the Euler angles. If the order of the Euler angles is changed the differential equation will change as well. This equation reflects the fact that the yaw angle is influenced by the rotation of roll and pitch and the pitch angle is influenced by the rotation in the roll axis. The measured turn rate of the x-axis directly reflects the change of the roll Euler angle because it is the last rotation.

To calculate the attitude we are interested in the inverse function that relates the measured turn rates to a change of the Euler angles of the system:

$$\begin{pmatrix} \dot{r} \\ \dot{p} \\ \dot{y} \end{pmatrix} = \mathbf{E}(r,p,y)^{-1} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} 1 & \sin(r)\tan(p) & \cos(r)\tan(p) \\ 0 & \cos(r) & -\sin(r) \\ 0 & \frac{\sin(r)}{\cos(p)} & \frac{\cos(r)}{\cos(p)} \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{3.12}$$

The drawback of this equation is that singularities are present when the Euler pitch angle reaches $\pm\frac{\pi}{2}$. For a hovering platform, this case should never happen as it is usually not possible to stabilize the platform in this attitude.

To gain the final attitude of the system, the following integration is used:

$$\mathbf{r}_k = \mathbf{r}_{k-1} + \int_{t_{k-1}}^{t_k} \mathbf{E}(r_t, p_t, y_t)^{-1} \omega(t) dt \tag{3.13}$$

As the turn rates are not available as a continuous signal, this equation is approximated by

$$\mathbf{r}_k \approx \mathbf{r}_{k-1} + \mathbf{E}(r_{k-1}, p_{k-1}, y_{k-1})^{-1} \omega T \tag{3.14}$$

where T is the sampling rate of the IMU, ω refers to the measured turn rates and $\mathbf{E}(r_{k-1}, p_{k-1}, y_{k-1})$ is the turn rate transformation matrix calculated with the values for roll, pitch and yaw from the previous estimation step. This approximation assumes that the turn rates are constant during one sampling interval. Instead of a rectangular integration, other approaches might be followed.

Attitude Computation with Rotation Matrices

Another way to keep track of the attitude is by using an orientation matrix. The benefit is that no singularities will occur with this representation.

Given an attitude of an object by its rotation Matrix $\mathbf{C}(t)$ for a certain time t, the rate of change of this matrix can be expressed as

$$\dot{\mathbf{C}}(t) = \lim_{\delta t \to 0} \frac{\mathbf{C}(t+\delta t) - \mathbf{C}(t)}{\delta t} \tag{3.15}$$

where $\mathbf{C}(t+\delta t)$ describes the attitude of the object at time $t+\delta t$ and can be rewritten as a combination of the initial rotation and a rotation $\mathbf{D}(t)$

$$\mathbf{C}(t+\delta t) = \mathbf{C}(t)\mathbf{D}(t) \tag{3.16}$$

To obtain $\mathbf{D}(t)$, the small angle approximation is used. It states, that for small changes in the attitude, the direction cosine matrix can be approximated. Starting from the definition of the orientation matrix with Euler angles, the approximation is given as:

$$\mathbf{C}(\delta r, \delta p, \delta y) = \mathbf{C}_x(\delta r)\mathbf{C}_y(\delta p)\mathbf{C}_z(\delta y) \approx \begin{pmatrix} 1 & \delta y & -\delta p \\ -\delta y & 1 & \delta r \\ \delta p & -\delta r & 1 \end{pmatrix} = \mathbf{D}(t) \tag{3.17}$$

This is possible because for small angles the approximation $\cos(\delta) = 1$, $\sin(\delta) = \delta$ and $\delta \times \delta = 0$ is true. This representation directly reflects the fact, that the order of rotations does not matter if the magnitudes of those are small. Regarding Euler angles, the same matrix will be derived for another order of the individual rotations. Nevertheless, it is dependent on the definition of the axes.

Splitting the matrix $\mathbf{D}(t)$ according to

$$\mathbf{D}(t) = \mathbf{I} + \delta\mathbf{A} \tag{3.18}$$

where $\mathbf{I}$ is the identity matrix and $\delta\mathbf{A}$ describes the part that is dependent on the change of the angles, the substitution into equation 3.15 yields

$$\dot{\mathbf{C}}(t) = \mathbf{C}(\mathbf{t}) \lim_{\delta t \to 0} \frac{\delta\mathbf{A}}{\delta t} \tag{3.19}$$

As the turn rates directly describe the change of the angles, this equation will be in the limit $t \to 0$

$$\dot{\mathbf{C}}(t) = \mathbf{C}(t)\Omega(t) \tag{3.20}$$

where

$$\Omega(t) = \begin{pmatrix} 0 & \omega_z(t) & -\omega_y(t) \\ -\omega_z(t) & 0 & \omega_x(t) \\ \omega_y(t) & -\omega_x(t) & 0 \end{pmatrix} \tag{3.21}$$

The matrix $\Omega(t)$ is also referred to as a skew symmetric matrix.

To be able to calculate the attitude, a solution for the differential equation is needed. For the relation between the times k and $k-1$ the solution is given as

$$\mathbf{C}_k = \mathbf{C}_{k-1} e^{\int_{t_{k-1}}^{t} \Omega(t)dt} \tag{3.22}$$

As we assume that the turn rates are constant during a sampling interval, the equation can be written as:

$$\mathbf{C}_k = \mathbf{C}_{k-1} e^{\Omega_k T} \tag{3.23}$$

To calculate this equation, the exponential function is expressed as an infinite power series:

$$e^{\Omega_k T} = \sum_{n=0}^{\infty} \frac{(\Omega_k T)^n}{n!} \tag{3.24}$$

what leads for the special case of our matrix Ω to the solution [Tit05]

$$\mathbf{C_k} = \mathbf{C}_{k-1}(\mathbf{I} + \frac{\sin(||\omega_k T||)}{||\omega_k T||}\Omega_k T + \frac{1-\cos(||\omega_k T||)}{||\omega_k T||^2}\Omega_k^2 T^2) \tag{3.25}$$

where $\mathbf{C}_{k-1}$ is the rotation of the previous time step, $\mathbf{I}$ is the identity Matrix and $||\omega_k T||$ is the second norm of the turn rate vector at time k multiplied with the sampling rate.

3.4.2 Position Calculation

As the inertial measurement unit is only sensitive to accelerations, these measurements have to be integrated once to gain the velocity and twice to get the current position. As the axis of the IMU moves with the body, the measurements have to be rotated from the IMU to a global frame in order to incorporate new measurements.

The differential equation to relate the the acceleration measured in the IMU frame to the velocity in the lab frame is

$$\dot{\mathbf{v}}^{(L)}(t) = \mathbf{C}_{LM}^{T}(t)\tilde{\mathbf{a}}^{(M)}(t) + \mathbf{g}^{(L)}(t) \tag{3.26}$$

where $\mathbf{C}_{LM}$ defines the momentary rotation between the lab and IMU frame and $\mathbf{g}^{(L)}$ refers to the gravity vector. As the gravity $\mathbf{g}^{(L)}(t)$ is assumed to be constant in the area the system operates, it is not time variant and $\mathbf{g}^{(L)}(t) = \mathbf{g}^{(L)}$ is true.

For the propagation of the velocity, the equation is integrated:

$$\mathbf{v}_k^{(L)} = \mathbf{v}_{k-1}^{(L)} + \int_{t_{k-1}}^{t_k} \mathbf{C}_{LM}^{T}(t)\mathbf{a}^{(M)}(t) + \mathbf{g}^{(L)} dt \tag{3.27}$$

Equally to the turn rate sensors, the values of the acceleration sensors are sampled and only available at discrete points in time. Thus, the velocity calculation is approximated by:

$$\mathbf{v}_k^{(L)} \approx \mathbf{v}_{k-1}^{(L)} + \mathbf{C}_{LM}^{T}(k-1)\left(\mathbf{a}_k^{(M)} + (\frac{1}{2}\omega(k) \times \mathbf{a}_k^{(M)})T\right)T + \mathbf{g}^{(L)}T \tag{3.28}$$

The second summand in the brackets compensates the fact, that the orientation is changing between two time steps. As we know the change from the turn rate sensor, it is possible to compensate this although the final contribution to the velocity update is small and it is usually neglected.

For the update of the position, the velocity has to be integrated

$$\mathbf{p}_k^{(L)} = \mathbf{p}_{k-1}^{(L)} + \int_{t_{k-1}}^{t_k} \mathbf{v}^{(L)}(t) dt \tag{3.29}$$

what results after the discretization in the equation

$$\mathbf{p}_k^{(L)} = \mathbf{p}_{k-1}^{(L)} + \mathbf{v}^{(L)}(k)T \tag{3.30}$$

3.4.3 Errors Related to the Strapdown Algorithm

The most critical path in a strapdown algorithm is the propagation of errors in the attitude calculation to errors in the position calculation. As the measured accelerations are projected to global axes where the compensation for the gravity occurs, any error in the attitude will lead to a wrong interpretation of the accelerations of the body. Assuming

the body is not rotated in reference to the lab frame, but the estimation differs by δr, δp and δy, the projection difference can be expressed as

$$\begin{aligned}\delta\mathbf{g}^{(L)} &= \mathbf{C}(0,0,0)\begin{pmatrix}0\\0\\-g\end{pmatrix} - \mathbf{C}(\delta r,\delta p,\delta y)\begin{pmatrix}0\\0\\-g\end{pmatrix}\\ &= \begin{pmatrix}g\sin(\delta p)\\-g\sin(\delta r)\cos(\delta p)\\-g\cos(\delta r)\cos(\delta p)\end{pmatrix} - \begin{pmatrix}0\\0\\-g\end{pmatrix}\end{aligned} \tag{3.31}$$

For a small angle approximation, the difference between the true gravity vector and incorrect projected one is

$$\delta\mathbf{g}^{(L)} = \begin{pmatrix}g\delta p\\-g\delta r\\0\end{pmatrix} \tag{3.32}$$

This error $\delta\mathbf{g}^{(L)}$ is directly interpreted as an acceleration of the body and will be twice integrated in the strapdown algorithm, what leads to a quadratic error with the time. Interestingly, this error is only relevant for the horizontal axes, the axes that are orthogonal to the gravity. The effect on the vertical axis is smaller causing the system to have higher stability in the height coordinate.

3.4.4 Initialization of the Strapdown Algorithm

For an successful initialization of the strapdown algorithm, the initial velocity, position and attitude has to be known. For our approach, we assume that the system starts while the platform is static. This means, the velocity is zero. Although the position is completely unknown, we initialize the position to zero as well. This is possible as the SLAM algorithm works with relative measurements. Thus, a translation of the start position will only lead to a shift in the final map.

As an error in the attitude is the most important issue for the strapdown algorithm, we initialize the orientation of the platform by exploiting the fact, that the gravity is always measured by the acceleration sensors. As the orientation of the gravity vector is known we are able to obtain the initial attitude.

Starting from the expected projection of the gravity vector to the axes of a rotated inertial measurement unit

$$\begin{pmatrix}a_x\\a_y\\a_z\end{pmatrix} = \mathbf{C}(r,p,y)_{LM}\begin{pmatrix}0\\0\\-g\end{pmatrix} = \begin{pmatrix}g\sin(p)\\-g\sin(r)\cos(p)\\-g\cos(r)\cos(p)\end{pmatrix} \tag{3.33}$$

it is obvious that the initial attitude can be calculated by

$$\begin{aligned}roll_{init} &= \arctan\tfrac{\bar{a}_y}{\bar{a}_z}\\ pitch_{init} &= \arcsin\tfrac{\bar{a}_x}{g}\\ yaw_{init} &= y_0\end{aligned} \tag{3.34}$$

where $\bar{a}$ are the averaged measurements from the acceleration sensors and g is the expected acceleration caused by the gravity of the earth.

As the yaw angle defines a rotation around an axis parallel to the gravity vector, the angle is not observable by using this method for the initialization. For a SLAM approach, that uses relative measurements, a wrong yaw angle will only lead to a rotation of the map. Thus, it can be chosen arbitrarily.

This estimation of the initial attitude is subject to the error characteristics of the acceleration sensors. While it possible to avoid the influence of the sensor noise by averaging over a number of samples, the bias will lead to a wrong result. This uncertainty has to be modeled when initializing the whole system.

4 Image Processing

We assume that the world exists of point landmarks that can be identified by their projections to the image plane. The aim is to follow these projections over several time steps to use them as measurements for the Bayesian estimator. As the distance of the landmarks is not directly visible via their projections, it has to be estimated using several measurements. A map is built with those landmarks that is simultaneously used for the localization of the platform.

This chapter describes how these projections are identified on a particular image of the camera, how they are stored in memory and used later in order to find the according landmark again in further images.

4.1 Image Processing Overview

We want to use natural landmarks in our approach. Instead of deploying artificial beacons, we will use structures of the environment as landmarks. This means that landmarks have to be identified on the image plane. This is done by a feature detector, that estimates areas of an image that can easily be tracked among several frames. We define the point in the environment, that caused the feature to appear, as the according landmark.

Once features are extracted from the image, the projection of the landmark has to be found in other images. As the measurements in a SLAM system are relative, a single observation does not provide any information about the propagation of the system state. To be able to find landmarks again, information of their projection are abstracted and stored into a feature descriptor.

With the help of the feature descriptor, the landmark can be recognized at a later point in time. The process of estimating the position of the projection with the help of the feature descriptor is called feature matching. It uses a metric to define the likelihood that a certain point on the image plane was caused by the same landmark.

4.2 Feature Detection

Feature detection refers to a method for abstracting image information. Instead of tracking all points seen in an image, only a subset of points is chosen that fulfills properties defined by a feature detector. Feature detectors can be split into three major groups:

- Corner detectors
- Edge detectors

- Blob detectors

The aim of feature detectors is usually to select structures of an image that can be easily recognized in other images. Corner detectors define an interesting point as a point with a high gradient on the image in each direction. Edge detectors respond to a gradient on a line in only one direction and blob detectors respond to areas with a smooth gradient in each direction. The difference between corner and blob detectors is vague, although blob detectors usually gain a metric by analyzing an area while corner detectors are more restricted to the direct proximity of a point. In this work, points of the environment should be tracked thus edge detectors are not further considered.

Blob detectors build a scale-space representation of an image that consists of several copies of the original image that are scaled and smoothed with different parameters. For each image within the scale-space a filter is used to detect interesting points, usually a Gaussian filter or an approximation of it. With an according descriptor, blob detectors can gain invariance for affine transformations of a feature. Common blob detectors are scale-invariant feature detector (SIFT) [Low04] and speeded up robust feature detector (SURF) [BTGL06]. They are both highly related to each other. SURF uses an approximation compared to SIFT to gain a faster computational speed but it was also shown that the matching results are worse. Despite their advantages, both still need a high computation time. As the real-time behavior is a constraint in this thesis, they were not chosen for feature detection.

In particular, we have chosen to use the Shi and Tomasi detector that is an improvement of the Harris and Stephensen corner detector. Both detectors are used frequently and showed to select points in an image that are reliable for tracking.

4.2.1 Harris and Stephensen Corner Detector

The idea of the Harris and Stephensen Corner Detector is to measure the self-similarity of a patch if it is shifted by small amounts around its original location, where a patch is a copy of a rectangular area of an image [HS88]. If the similarity measurement of the shifted patch changes noticeable from the center position, a corner is detected. In this case the patch is said to have a low self-similarity. As the metric of the similarity between a patch and the image at a certain position, the sum of squared distances (SSD) is used.

Given an image patch $\mathbf{P}$ with fixed size, the sum of squared distances (SSD) between the original patch and the shifted patch can be calculated as:

$$SSD(\delta x, \delta y) = \sum_{(x,y)\in\mathbf{P}} \omega(x,y)(\mathbf{I}(x,y) - \mathbf{I}(x+\delta x, y+\delta y))^2 \tag{4.1}$$

Where $\omega(x,y)$ defines a smoothing function e.g. a Gaussian window and $\mathbf{I}(x,y)$ is the intensity of the pixel. $\omega(x,y)$ can also be a constant in order to speed up the calculation of this metric. But in this case, the detector is more sensitive to image noise.

With this function, a corner can already be identified by searching for the minimum SSD of a patch in a defined proximity. If this minimum is above a threshold, the self-similarity of the patch is low and an interesting point is detected. This method was first proposed

by [Mor77] on which the Harris and Stephensen Corner detector is based. The drawback with this method is that it only takes discrete shifts of the patch into account and thus it depends on the direction of the gradients within the area. To overcome this, Harris and Stephensen developed a method that takes the differentials into account instead of shifted patches to gain a direction-invariant metric.

In order to incorporate continuous shifts, a first order Taylor expansion is used

$$\mathbf{I}(x+\delta x, y+\delta y) \approx \mathbf{I}(x,y) + \mathbf{I}_x(x,y)\delta x + \mathbf{I}_y(x,y)\delta y \tag{4.2}$$

where $\mathbf{I}_x$ and $\mathbf{I}_y$ represent the first differentiation of the image. Substituted into the SSD function gives

$$SSD(\delta x, \delta y) \approx \sum_{(x,y)\in\mathbf{P}} \omega(x,y)(\mathbf{I}_x(x,y)\delta x + \mathbf{I}_y(x,y)\delta y)^2 \tag{4.3}$$

This can be written as a matrix as

$$SSD(\delta x, \delta y) \approx \begin{pmatrix}\delta x & \delta y\end{pmatrix} \mathbf{T} \begin{pmatrix}\delta x \\ \delta y\end{pmatrix} \tag{4.4}$$

where $\mathbf{T}$ is called structure tensor and is defined as

$$\mathbf{T} = \begin{pmatrix} \sum_{(x,y)\in\mathbf{P}} \omega(x,y)\mathbf{I}_x(x,y)^2 & \sum_{(x,y)\in\mathbf{P}} \omega(x,y)\mathbf{I}_x(x,y)\mathbf{I}_y(x,y) \\ \sum_{(x,y)\in\mathbf{P}} \omega(x,y)\mathbf{I}_x(x,y)\mathbf{I}_y(x,y) & \sum_{(x,y)\in\mathbf{P}} \omega(x,y)\mathbf{I}_y(x,y)^2 \end{pmatrix} \tag{4.5}$$

The structure tensor $\mathbf{T}$ encodes the change of the metric due to a continuous shift of the patch. The higher the change of the metric caused by shifting, the more interesting the considered point. For the actual calculation of the magnitude of this change, the eigenvalues λ_1 and λ_2 of $\mathbf{T}$ are reviewed:

- If $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$, the point is not regarded as a feature
- If $\lambda_1 \approx 0$ and $\lambda_2 \gg 0$ then an edge is found
- If $\lambda_1 \gg 0$ and $\lambda_2 \gg 0$ a corner is found

As the calculation of the eigenvalues for this matrix includes two square roots, another function was suggested for the calculation [HS88]:

$$M_c = \lambda_1\lambda_2 - \kappa(\lambda_1+\lambda_2)^2 = \det(T) - \kappa\,\mathrm{trace}^2(T) \tag{4.6}$$

where κ donates a tunable parameter. If M_c is over a certain threshold, the point is regarded as an interesting point.

4.2.2 Shi and Tomasi Corner Detector

Shi and Tomasi [ST94] revised Harris and Stephens corner detector and proofed, that another metric is better if the according image area is used for tracking. To use an area for tracking means, that the according patch should be reliable recognized even if the

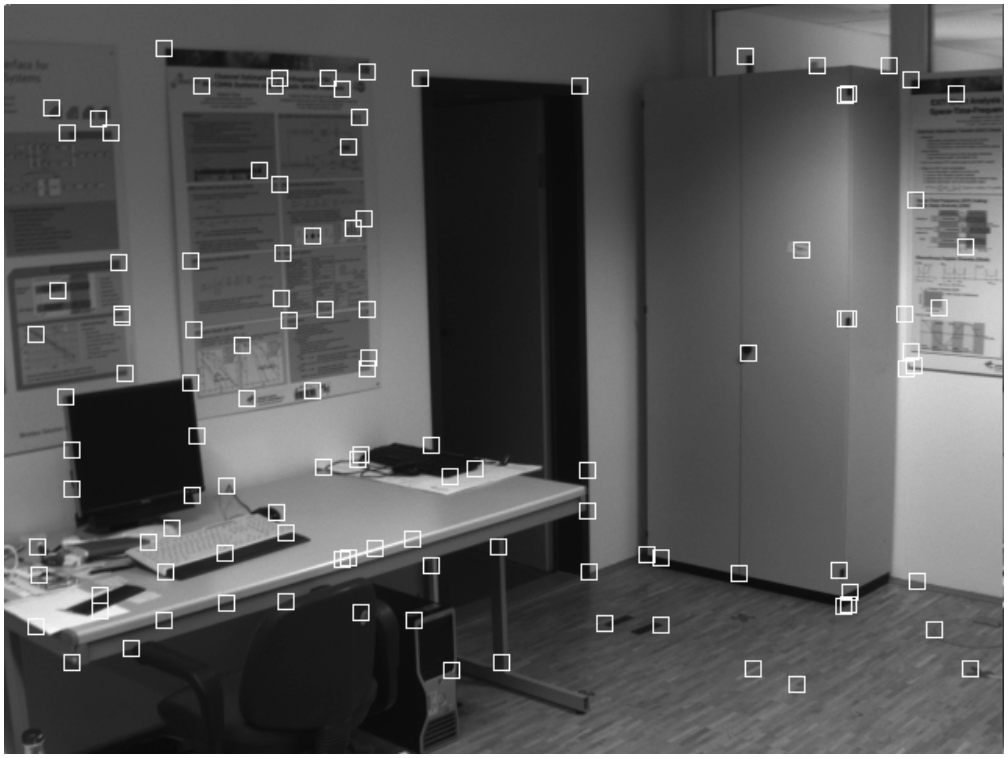


Figure 4.1: An example of the results of the Shi and Tomasi corner detection algorithm. The rectangles refer to points that the algorithm identified as corners. Before the corner detection was run, the image was divided into a grid with cells of the size 30x40 pixels. Only the maxima within those cells is plotted.

viewpoint changes. They proposed, that a point should be regarded as an interesting point if the smaller eigenvalue exceeds a certain threshold

$$min(\lambda_1, \lambda_2) > \theta \tag{4.7}$$

what lowers the probability of finding edges instead of corners.

The drawback of the Shi and Tomasi as well as the Harris and Stephensen detector is that they will only detect interesting points within one scale of an image compared to blob detectors. If the gradient is smooth, the point will not be detected. Furthermore, they also respond to particular kind of edges.

4.3 Feature Description

A feature descriptor is used to describe visual features of images. A feature descriptor ideally describes an image feature in a way, that it is robust in the sense of noise and affine transformation if tried to be detected again in following images. Feature descriptors might include abstractions of the color, texture, gradients and other characteristics

of a certain area of the image. For our approach, we have chosen to use a patch as a feature descriptor because of its simplicity and its fast processing.

4.3.1 Patch as a Feature Descriptor

A patch is a very simple feature descriptor and contains the image data itself in an area. No further abstractions are made. The advantage is a relative small computational effort to calculate the matching score between an image and a patch and no further calculations are needed to gain a patch from an image. Just the image data itself is copied and stored in memory. On the other hand, patch features in their simplest form are not invariant to any transformation. Nevertheless, they are heavily used because of their simplicity.

4.3.2 Rotation Compensation

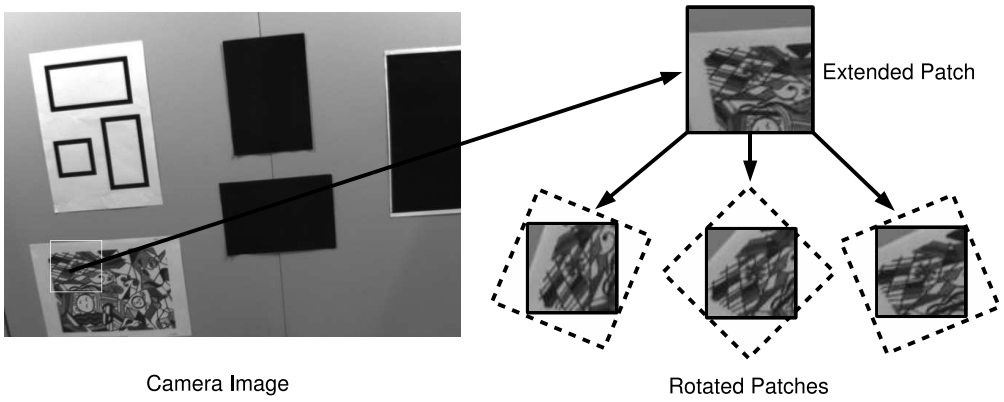


Figure 4.2: Patches are used as feature descriptors in order to track points among several frames. Instead of storing the size that is used for the correlation metric, an extended area is hold in memory. This makes it possible to retrieve rotated versions of the patch that are parallel to the image border in order to exploit certain strategies to speed up the matching process.

Although the viewpoint to a feature may not change much especially in scenarios where the aim is to stabilize a micro air vehicle, a rotation around the optical axis usually discriminates the results heavily. Thus, a compensation of the rotation of the view is needed. For this reason, the initial rotation α_{init} for each patch is stored among the data. When it comes to matching, the current estimation of the rotation α is taken to rotate the patch accordingly. This is a simple rotation in two dimensions and has to be done once for a patch before the matching process starts.

Another possibility might be the rotation of the image instead of the patch, but this leads to a higher computational time as the search area that has to be rotated is larger than the patch and more pixels have to transformed. Another approach might be the usage of a mapping function what also increases the computational time as this function has

to be used every time a pixel from the patch is related to a pixel in the image. This means that a rotation of the patch itself is preferable.

The rotated patch should have the same dimensions like the original patch with the borders parallel to the image borders. This is necessary to exploit integral images, a technique used to speed up the matching process. If just rotating a rectangular patch, the values of some pixels within the original area are undefined. Thus, an extended patch is used that stores a larger area of the original image than is actually needed for the matching process. With the extra pixels of the extended patch, all pixels can finally be calculated.

The pixels of the rotated patch $\mathbf{P}_{rot}(x,y)$ out of the data of the extended patch $\mathbf{P}_{ext}(x,y)$ are calculated as follows:

$$\forall (x,y) \in \mathbf{P}_{rot} \mapsto \mathbf{P}_{rot}(x,y) = \mathbf{P}_{ext}(\lfloor u+0.5 \rfloor, \lfloor v+0.5 \rfloor)$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cos(\alpha - \alpha_{init}) & -\sin(\alpha - \alpha_{init}) \\ \sin(\alpha - \alpha_{init}) & \cos(\alpha - \alpha_{init}) \end{pmatrix} \begin{pmatrix} x - \frac{\dim_x(\mathbf{P}_{rot})}{2} \\ y - \frac{\dim_y(\mathbf{P}_{rot})}{2} \end{pmatrix} + \begin{pmatrix} \frac{\dim_x(\mathbf{P}_{ext})}{2} \\ \frac{\dim_y(\mathbf{P}_{ext})}{2} \end{pmatrix} \tag{4.8}$$

where α_{init} and α define the rotation when the patch was initialized respectively the momentary estimation of the rotation, $\dim_x$ and $\dim_y$ are operators to calculate the width and height of the patches in pixels, u and v define coordinates of the extended patch and x as well as y define coordinates of the rotated patch.

This approach for the rotation compensation uses a nearest neighbor approximation to calculate the actual pixel value for the rotated patch. A further improvement is the usage of an interpolation function to enhance the quality of the rotation.

Instead of performing this interpolation for each pixel while calculating the rotated patch according to equation 4.8, it is proposed here to use an already interpolated version of the extended patch. This representation has to be calculated only once, when the patch is initialized, and thus the computational effort for the rotation compensation will not change while the results are improved.

4.4 Feature Matching

Feature matching is the process of finding one point of the scenery in two different images using a feature descriptor and a metric that defines the likelihood of a match. Different positions of the image are examined for a match what means that feature matching always includes the search over some area in the image.

To actually do the matching a metric for the comparison of feature descriptors is needed. Depending on this metric several results are distinguished:

- Correct match
- Absence match
- False match
- Ambiguous match

A match is called correct if the point in the scenery was identified correctly. An absence match means that the descriptor could not been found in the searched area. That is usually detected by a matching metric that falls under a certain threshold. False matches are points that are erroneously identified as matches. This kind of matches can lead to serious problems in any estimation algorithm. In consequence, false matches are prevented or detected. Matches are named ambiguous if two positions in the search area lead to almost the same extremes considering the metric. This can happen because of the properties of the texture of an object or the position for a feature was badly chosen. E.g. on a line the feature position is hard to estimate orthogonal to the gradient.

In general, feature matching for a set of features can be done in two different ways:

- Bottom-up approach
- Top-down approach

The bottom-up approach calculates the feature descriptors of all interesting points in an image in order to compare it to the descriptors in a database. This approach is computational very expensive as many descriptors are calculated at positions where a match is absolutely not likely. This approach is useful if no information about the positions of the feature within an image is available.

The top-down approach works with regions of interest (ROI). These regions define areas where a point of the scenery is estimated. Thus, only the area within the region of interest is explored for a feature. This approach is also called active matching and can save a lot of computational time. Depending on the certainty about the position the search area is greatly reduced. To gain the certainty, the knowledge about the position of the point in the scenery has to be converted to an area of the image plane. Furthermore, false matches are prevented as the search area is limited and less candidates for a false matching result are available.

In our approach, the top-down approach is used as we have a continuous stream of image data and the data from the inertial measurement unit and are thus able to calculate a region of interest for the matching of each feature depending on its certainty.

4.4.1 Patch Matching

As a patch is an extract of an image, the metric for matching is directly a function of the image data and the data stored in a patch. Among those metrics are:

- Cross correlation (CC)
- Normalized cross correlation (NCC)
- Sum of squared differences (SSD)
- Normalized sum of squared differences (NSSD)
- Sum of absolute differences (SAD)

While the approach of cross correlation, sum of squared differences and sum of absolute differences is sensitive to changes in the illumination of the scenery, the normalized

versions do not suffer from this phenomenon. Thus, the normalized versions of the metrics are preferable if image data from natural environment is used where an illumination change is very likely.

It was shown that the normalized cross correlation can better handle affine transformations of an image while the normalized sum of squared differences usually gives better results if no transformations of the image occurs except translation. So it depends on the expected trajectory which similarity metric is used. The computational complexity of both metrics is similar and the overall computing time can be speeded up by the usage of so called integral images.

For our approach, we used the normalized cross correlation to gain a metric of the similarity. The next section describes how it is defined.

Normalized Cross Correlation

The normalized cross correlation of an image Patch $\mathbf{P}(x,y)$ of size n with an image $\mathbf{I}(x,y)$ at position $(\delta x, \delta y)$ in the image is defined as:

$$NCC(\delta x, \delta y) = \frac{1}{n} \sum_{(x,y)\in\mathbf{P}} \frac{(\mathbf{P}(x,y) - \bar{\mathbf{P}})(\mathbf{I}(x+\delta x, y+\delta y) - \bar{\mathbf{I}})}{\sigma_P \sigma_I} \tag{4.9}$$

where

$$\begin{aligned} \bar{\mathbf{P}} &= \tfrac{1}{n} \sum_{(x,y)\in P} \mathbf{P}(x,y) & \sigma_{\mathbf{P}} &= \sqrt{\tfrac{1}{n} \sum_{(x,y)\in P} (\mathbf{P}(x,y) - \bar{\mathbf{P}})^2} \\ \bar{\mathbf{I}} &= \tfrac{1}{n} \sum_{(x,y)\in P} \mathbf{I}(x+\delta x, y+\delta y) & \sigma_{\mathbf{I}} &= \sqrt{\tfrac{1}{n} \sum_{(x,y)\in P} (\mathbf{I}(x+\delta x, y+\delta y) - \bar{\mathbf{I}})^2} \end{aligned} \tag{4.10}$$

Especially for patches with a high size this calculation can quickly become the bottleneck of the whole system. The function has to be evaluated for each feature that is expected in an image within the whole area of interest.

To optimize the calculation, the function can be rewritten as

$$NCC(\delta x, \delta y) = \frac{\sum_{(x,y)\in\mathbf{P}} (\mathbf{P}(x,y)\mathbf{I}(x+\delta x, y+\delta y)) - \bar{\mathbf{P}}\,\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y})}{\sigma_{\mathbf{P}} \sqrt{(n\,\mathrm{Sum}^2_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) - (\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}))^2)}} \tag{4.11}$$

where

$$\begin{aligned} \mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) &= \sum_{(x,y)\in\mathbf{P}} \mathbf{I}(x+\delta x, y+\delta y) \\ \mathrm{Sum}^2_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) &= \sum_{(x,y)\in\mathbf{P}} \mathbf{I}^2(x+\delta x, y+\delta y) \end{aligned} \tag{4.12}$$

A detailed derivation is described in appendix A.1.

If the patch itself does not change $\bar{\mathbf{P}}$ and $\sigma_{\mathbf{P}}$ have only to be evaluated once. The advantage of this rearrangement of the equation 4.9 is that the terms $\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y})$ and $\mathrm{Sum}^2_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y})$ can be calculated with just four operations for each term after the integral

image was calculated. Thus, the only term that has to be evaluated for each point and slows down the normalized cross correlation is

$$\sum_{(x,y)\in\mathbf{P}} (\mathbf{P}(x,y)\mathbf{I}(x+\delta x, y+\delta y)) \tag{4.13}$$

what makes it almost as fast as the standard cross correlation (CC) but with better results.

Integral Image

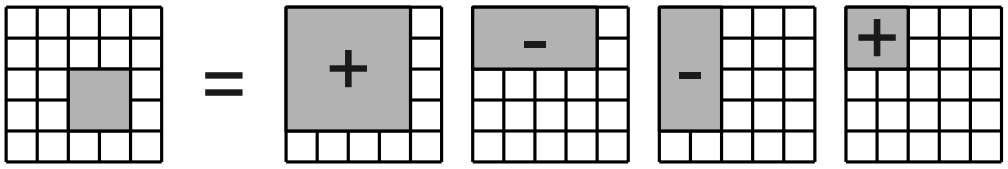


Figure 4.3: Illustration showing the operations involved to calculate the sum of a rectangular area with the help of integral images.

The terms $\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y})$ and $\mathrm{Sum}^2_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y})$ are calculated in the integral $\mathbf{Int}_{\mathbf{I}}(x,y)$ respectively integral squared $\mathbf{Int}^2_{\mathbf{I}}(x,y)$ representation of an image. These representations offer a fast way to calculate any sum of the values or squared values of a rectangular area of an image. This was also an argument to represent even rotated patches within an rectangular area. This representation was first introduced by [Cro84] that used it for fast texture mapping.

The integral representations are defined as

$$\mathbf{Int}_{\mathbf{I}}(x,y) = \sum_{\substack{0\le u\le x \\ 0\le v\le y}} \mathbf{I}(u,v) \qquad \mathbf{Int}^2_{\mathbf{I}}(x,y) = \sum_{\substack{0\le u\le x \\ 0\le v\le y}} \mathbf{I}^2(u,v) \tag{4.14}$$

To calculate the integral and integral squared representation, a dynamic programming approach is used. For each single entry, only two additions and one subtraction is needed if done in an unoptimized way. This can be further reduced to two additions per entry [Cro84].

The sum of the values or squared values within an rectangle $(x_1,y_1)\times(x_2,y_2)$ is calculated as:

$$\begin{aligned} \sum_{\substack{x_1\le x\le x_2 \\ y_1\le y\le y_2}} \mathbf{I}(x,y) &= \mathrm{Int}(x_2,y_2) - \mathrm{Int}(x_1-1,y_2) - \mathrm{Int}(x_2,y_1-1) + \mathrm{Int}(x_1-1,y_1-1) \\ \sum_{\substack{x_1\le x\le x_2 \\ y_1\le y\le y_2}} \mathbf{I}^2(x,y) &= \mathrm{Int}^2(x_2,y_2) - \mathrm{Int}^2(x_1-1,y_2) - \mathrm{Int}^2(x_2,y_1-1) + \mathrm{Int}^2(x_1-1,y_1-1) \end{aligned} \tag{4.15}$$

It has to be stressed, that the integral image is not calculated for the whole image in our approach. Instead, it is only calculated for the region of interest where a matching candidate is searched for.

5 Visual SLAM

In this chapter the integration of the different components to derive a system capable of solving the simultaneous localization and mapping problem is shown. Based on an approach using a particle filter and extended Kalman filters to solve the combined estimation problem, the according state evolution and measurement models are discussed in detail. This enables in the end the merging of the measurements from the camera and the inertial measurement unit.

5.1 Rao-Blackwellized Particle Filter

To estimate the system state, we use a Rao-Blackwellized particle filter, also called Rao-Blackwellized Sampling Importance Resampling filter. This Bayesian filter splits the estimation process into several filters depending on the representation of the uncertainty. A particle filter is used to estimate the part of the system that is not representable by a Gaussian distribution. This mainly includes the pose of the platform. The other parts of the system, namely the landmarks in our case, are estimated using an Extended Kalman Filter (EKF) that models uncertainties with Gaussian distributions.

The aim of full SLAM is to estimate the posterior of the system state $\mathbf{y}_{0:k}$ for all points in time with the available measurements $\mathbf{z}_{1:k}$ and control inputs $\mathbf{u}_{1:k}$ up to this time. The system state is combined of the platform state $\mathbf{x}_{0:k}$ and the map $\mathbf{m}$ of point landmarks. For the Rao-Blackwellized particle filter approach, the posterior is split according to

$$\begin{aligned} p(\mathbf{y}_{0:k}|\mathbf{z}_{1:k},\mathbf{u}_{1:k}) &= p(\mathbf{x}_{0:k},\mathbf{m}|\mathbf{z}_{1:k},\mathbf{u}_{1:k}) \\ &= \underbrace{p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k},\mathbf{u}_{1:k})}_{\text{platform state}} \underbrace{p(\mathbf{m}|\mathbf{x}_{0:k},\mathbf{z}_{1:k})}_{\text{map state}} \end{aligned} \tag{5.1}$$

As the probability density function of the platform state is modeled as a set of hypotheses, the map estimation can be factorized. This is possible because each hypothesis models a certain evolution of the state of the platform without uncertainty. The uncertainty itself is modeled with the distribution of the individual hypotheses. If the state of the platform is exactly known at every certain point in time, the individual measurements of the landmarks get independent as can be visualized with a Bayesian network.

The Bayesian network 5.1 shows the dependencies of the individual random variables if the state of the platform is perfectly known for every point in time. It can be seen that the individual measurements $\mathbf{z}$ become independent in this case. This is only possible if the whole evolution of the platform state is known for which reason this approach solves the full SLAM problem. On the other hand, as the whole set of particles represents only the distribution of a single state in time, the underlying algorithm has to be an

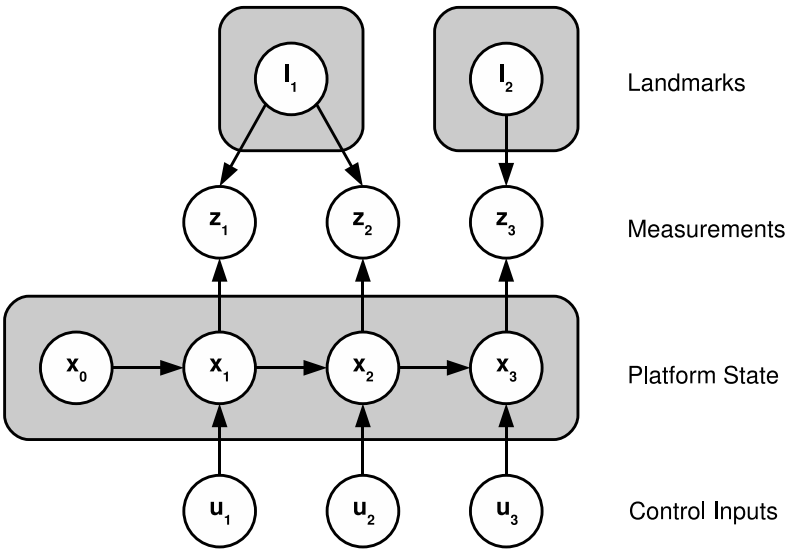


Figure 5.1: Bayesian network showing the dependencies of the random variables for the factored solution of the simultaneous localization and mapping problem (FastSLAM). As the platform states are modeled without uncertainty for an individual particle, the measurements of the landmarks are conditionally independent.

online algorithm and thus the online SLAM problem is solved as well. The idea that the knowledge of all the past platform states leads to a conditionally independent estimation of the landmarks was first exploited by Thrun who introduced it as factored solution to the SLAM problem (FastSLAM) [MTKW02] and proofed that

$$p(\mathbf{y}_{0:k}|\mathbf{z}_{1:k},\mathbf{u}_{1:k}) = \underbrace{p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k},\mathbf{u}_{1:k})}_{1 \times \text{ particle filter}} \prod_i \underbrace{p(\mathbf{l}^{(i)}|\mathbf{x}_{0:k},\mathbf{z}_{1:k})}_{\text{i} \times \text{ EKF filter}} \tag{5.2}$$

is true if a particle filter is used for the platform state. The idea is that each single particle has its own map where the landmarks are tracked with an extended Kalman filter. Thus, the whole posterior consists of actually $1+NL$ filter, one particle filter for the path, and L EKF filter for each landmark of the N particles.

5.1.1 Platform State Estimation

The particle filter used to estimate the platform state is similar to the described one in section 2.4.4 except the fact that the whole history is part of the posterior for the derivation.

It follows the same three steps: Sampling, importance weighting and resampling. During the sampling, the proposal function is created that uses the state evolution model to describe the propagation of the system state as

$$q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k}) = p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1},\mathbf{u}_{1:k-1}) \tag{5.3}$$

where $p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}, \mathbf{u}_{1:k-1})$ is represented as the set of particles from the previous time step and $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$ refers to a sampling process of the state evolution distribution that only depends on the last state and the current control input. As we use the inertial measurement unit for the propagation of the platform state, the state evolution model mainly consists of the strapdown algorithm and a sampling procedure for the expected errors of the sensors readings.

The importance weights represent the ratio between the proposal function q and the target function p, where the target function incorporates the current measurement $\mathbf{z}_k$. This means in the case of Rao-Blackwellized SLAM for a single hypothesis:

$$\omega_k^{[i]} = \frac{target}{proposal} = \frac{p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}, \mathbf{u}_{1:k})}{q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k-1}, \mathbf{u}_{1:k})} = \alpha p(\mathbf{z}_k|\mathbf{x}_k^{[i]})\omega_{k-1}^{[i]} \tag{5.4}$$

The probability $p(\mathbf{z}_k|\mathbf{x}_k^{[i]})$ is calculated by marginalizing the current estimation of the map and derived in a later section.

5.1.2 Map Estimation

The individual landmarks within the map are estimated by an extended Kalman filter. As the landmarks are assumed to be static, the process noise is zero and the state transition model simplifies to an identity matrix. This means that the prediction step is obsolete.

The update of a landmark incorporates a new measurement that is available at time k:

$$p(\mathbf{l}|\mathbf{x}_{0:k}, \mathbf{z}_{1:k}) = \alpha p(\mathbf{z}_k|\mathbf{l}, \mathbf{x}_k) p(\mathbf{l}|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k-1}) \tag{5.5}$$

This update is implemented by following the standard equations for the extended Kalman filter.

5.1.3 Comparison to EKF Based SLAM

The advantage of the possible factorization when using a Rao-Blackwellized particle filter is a gain of computational efficiency especially for SLAM. Other approaches are using a single, high-dimensional EKF filter that estimates the whole state, consisting of all the landmarks and platform states, usually referred to as EKF SLAM. If the number of landmarks is L, where each is represented with three coordinates, and the number of platform states is P, then the filter estimates in total the parameters for a $T = P + 3L$ dimensional Gaussian distribution. When exploiting the fact, that only a subset of the landmarks are measured, the whole complexity for one update step is quadratic in the number of landmarks $O(L^2)$. This fact limits the application of SLAM approaches that use an approach only based on extended Kalman filters as they are not able to handle maps that exceed a certain amount of landmarks.

5.2 System State Representation

Each particle represents one hypothesis of the system state at a particular point in time. This system state is combined of the platform state and the map state.

5.2.1 Platform State

The platform state is split into the pose and the dynamic state. The pose is the position and attitude of the platform in a global reference frame, in our case the lab frame. All other parameters contributing to the platform state are composed in the dynamic state. In our case this is the velocity of the platform.

In our approach, the platform state at time k is defined as:

$$\mathbf{x}_k = (\underbrace{t^{(L)}_{x,m,k}, t^{(L)}_{y,m,k}, t^{(L)}_{z,m,k}}_{\text{Platform position}} \underbrace{\dot{t}^{(L)}_{x,m,k}, \dot{t}^{(L)}_{y,m,k}, \dot{t}^{(L)}_{z,m,k}}_{\text{Platform velocity}} \underbrace{r_k, p_k, y_k,}_{\text{Platform attitude}})^T = \begin{pmatrix} \mathbf{t}^{(L)}_{m,k} \\ \dot{\mathbf{t}}^{(L)}_{m,k} \\ \mathbf{r}_{LM,k} \end{pmatrix} \tag{5.6}$$

Instead of the body, the platform state refers to the translation and rotation between the IMU and the lab frame. We have chosen this representation for the platform state as the measurements from the IMU are available at a higher rate compared to the data from the camera. Whenever an operation in the camera frame or the body frame is necessary, the translation and rotation has to be transformed to the according coordinate systems.

The platform state can contain more states, like the biases of the sensors of the inertial measurement unit. We assumed that those are constant during the run of the experiments and thus did not include them to the platform state. Although an additional state can contribute for a longer stability of the whole system, the number of particles grows usually exponential with the number of parameters. This trade-off has to be considered especially in the case of real-time systems.

5.2.2 Map State

The map consists of the states of several landmarks. As the states of the landmarks are estimated with an extended Kalman filter, the state of one landmark consists of a mean vector and a covariance matrix according to the parametrization. The parameterization of the landmarks depends on the chosen representation. We only use one global map per particle that combines all available landmarks in the lab frame.

Other approaches exist that represent the whole map as a combination of several sub-maps. The advantage of those approaches is that not the whole map state has to be present in memory during the execution of the system. Instead, only the part has to be available that is momentary in range of the sensors of the platform. Furthermore, errors between the several sub-maps can be modeled and corrected what will lead to a more robust system.

As the aim of this thesis is mainly to derive an implementation for a micro air vehicle operating in a limited area, we did not investigate into the sub-mapping approach. Instead we assumed that a global map is sufficient as the platform does not explore a wide area.

5.3 Processing Overview

During one iteration of the algorithm, either a new image or a measurement from the inertial measurement unit is processed. Depending on the type of available data, two different branches are followed.

5.3.1 Processing of Data from the IMU

The data from the IMU is used to update the proposal distribution. With the help of the state evolution model, the state of the platform hypotheses is modified according to the available data of the IMU and the expected error. In the end this will result in different hypotheses of the sensor noise that affected the current data. This means, that the whole set of particles will expand and occupy a larger area with time to model the uncertainty introduced.

5.3.2 Processing of Data from the Camera

The processing of data from the camera is more complex compared to the IMU processing. The camera is the only device that gives us information about the landmarks and thus has access to global references. Before an image is processed, a subset of the available landmarks is selected. This subset reflects the set of landmarks that are going to be tracked during a certain iteration. After that, a landmark is matched using its associated feature descriptor. If a matching was successful, the available data about the position of the feature on the image plane can be used to update the landmark state as well as to correct the estimation of the platform state. Finally, the image is searched for new landmarks that can be initialized if the number of available features drops below a certain threshold.

Calculation of Visible Landmarks

Once a new image is ready for processing, the first question to be answered is what landmarks should actually be tracked. Instead of trying to detect all landmarks of the database, the amount is limited in our approach. This is due to computational reasons and to avoid ambiguities during matching.

To calculate the visible features, the mean estimation of the position of a particular landmarks is projected to the image plane according to the state of each particle. If the projection lies within the area that the image sensor can capture, the landmark is added to the list of visible features. To limit the total amount of features, this process is stopped if a specific number of visible landmarks has been selected.

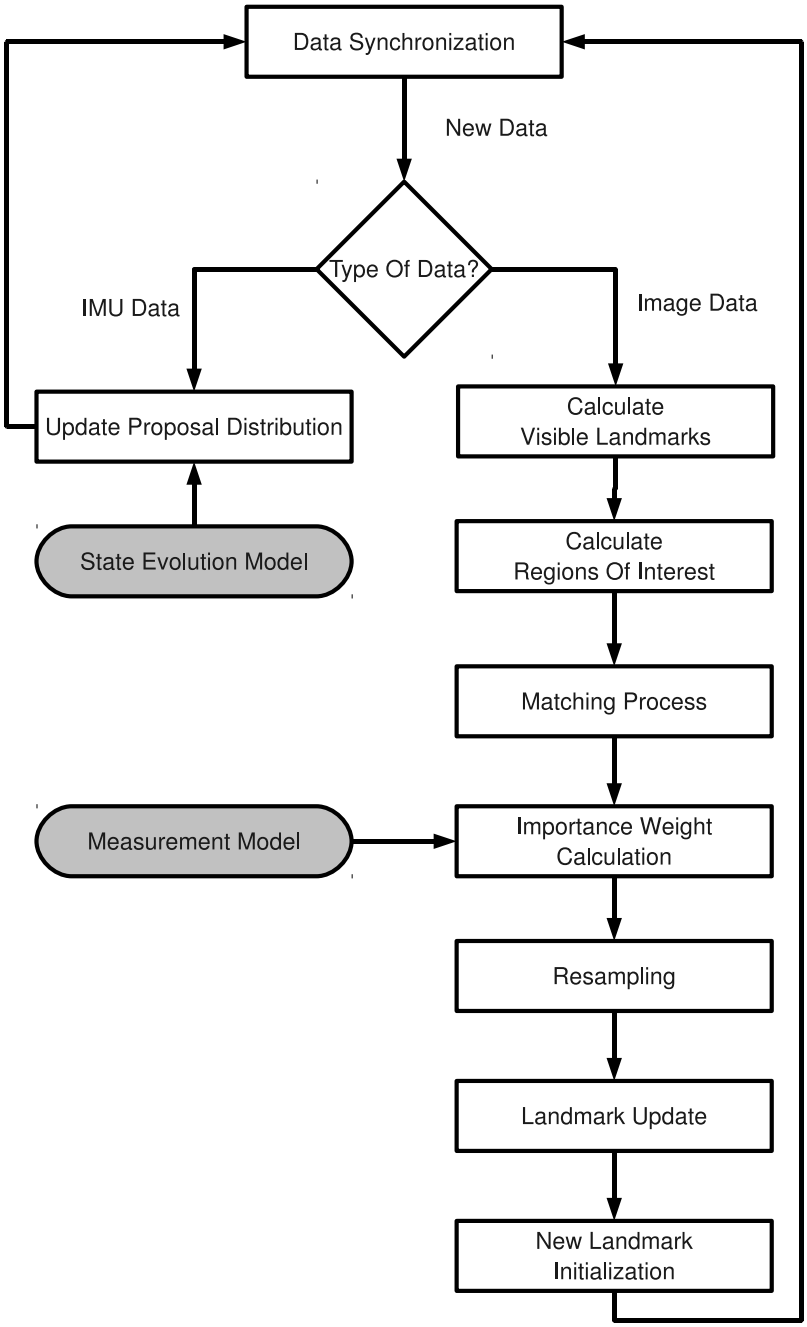


Figure 5.2: Illustration of the processing steps involved in this approach. Data from the inertial measurement unit is used to modify the proposal function with the help of the state evolution model. The images from the camera are processed to derive the position of features on the image plane that represent the projection of landmarks. These positions are used to calculate the importance weights of each particle with the help of the measurement model. Furthermore, the positions of the features are also used to update the estimations of the landmarks.

As only a subset of the possible features is considered, a strategy has to be used for limiting the amount. In this approach, landmarks are favored according to the time of initialization. As we iterate through the available map of landmarks for the selection of visible landmarks, those landmarks will be considered first, that were initialized earlier. This reduces the number of needed iterations and thus contributes for the real-time application.

When using sub-mapping the selection process can be further speeded up as the map is already limited due to geometric constraints. Also, the selection process could optimize the set of landmarks in the sense, that the expected contribution to the posterior is maximized. Both approaches were not followed within this work.

Calculation of Regions of Interest

To reduce the computational time for the matching process, regions of interest are calculated for each feature. The matching process will be limited to this area what also prevents ambiguous and false matches as the search space is limited.

This top-down approach was introduced by [CD09] where the uncertainty of the landmark was projected to the image plane to gain the regions of interest. As [CD09] used an EKF based SLAM approach it was possible to transfer the uncertainty by one calculation as it is directly accessible via the filter state.

When using a Rao-Blackwellized approach, the projection of the uncertainty involves more operations as it is a combination of the different platform states and the state of the individual EKF filters that estimate the parameters of a landmark. This leads to a high computational effort why we have chosen another strategy. We only project the mean estimation of the landmarks to the image plane. Each projection is the center of a rectangular area where the according feature is expected. The union of these rectangles define the complete region of interest in the end.

Matching Process

For the description of the features, patches are used in this approach. Before the matching occurs, the interpolated patches are rotated to match the MMSE estimation of the momentary platform rotation around the optical axis and afterwards a metric is calculated using the normalized cross correlation.

The calculation of the metric is done for every possible position within the region of interest. Although this process is optimized by using integral images, the remaining complexity is high. This is caused as for every position the multiplication between all the data of the patch and the according image data has to be taken out.

Instead of taking just the maximum of the metric within the region of interest, the result is grouped into correct, absence and ambiguous matches. An absence match is detected if the metric stays below a threshold for all considered points. To detect an ambiguous match, the maximum of the metric is compared to the second highest value. If the ratio is over a certain threshold, what means that the values do not differ much, the match is regarded as being ambiguous. If neither an absence nor an ambiguous match is detected, the result is regarded as being correct.

The concept of absence and ambiguous matches helps to prevent outliers. Points, that were retrieved by the feature detector but are actually not positioned on a highly discriminating area get detected in this way. Not correct matches are neither used for the update of the landmark state nor for the calculation of the importance weights.

Nevertheless, it is not possible to detect false matches with this concept. Although the possibility of such matches is reduced, they are still possible and accepted without further consideration in our application. We assume that an overall correct matching result will reduce the influence of false matches to a minimum.

Importance weight calculation

The importance weights reflect the ratio between the target and the proposal function. With our definition of the proposal function, one importance weight directly represents the measurement likelihood for a certain platform state. The exact calculation is discussed in section 5.5.1.

In figure 5.3 an example for the resulting importance weights for an artificial scene is shown. The first plot visualizes the weights on a plane parallel to the optical axis of the camera while the second plot visualizes the weights of an orthogonal plane to the optical axis. It is clearly visible that the discrimination along the viewing direction of the camera is small because of its inability to measure distances. As the plot was made while tracking several landmarks, a small discrimination was nevertheless possible.

Resampling step

In order to get a fixed execution time, we have chosen to trigger the resampling process in each iteration. To keep the variance of the system states, systematic resampling is used as described in section 2.4.4.

Landmark Update

After the calculation of the importance weights of the particles and the resampling step, the landmarks are updated. This means, that the associated Kalman filters for each landmark is given a new measurement according to the position of the matching result which is used to update the estimation of the landmark. For this, the standard equations of the extended Kalman filter are used.

This update is postponed after the resampling step in our approach. Instead of adjusting each hypothesis of the map and afterwards discarding a lot of hypotheses including updated landmarks, it is better to update the map after resampling. As many particles will share the same position and map, the update has only to be done once per shared landmark. This method becomes possible by using a lazy-copy strategy of the landmark states. This means, that the actual copying is postponed until it is definitely needed.

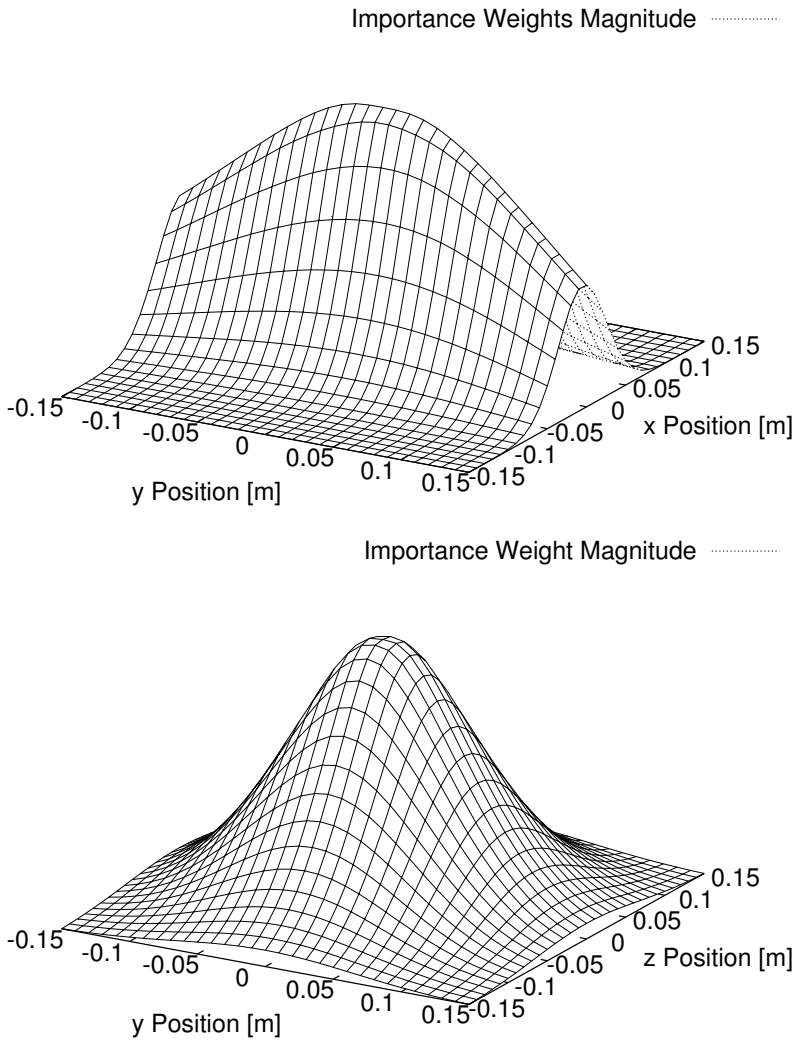


Figure 5.3: These plots illustrate the calculated importance weights depending on the position of the platform for an artificial scene consisting of several landmarks. The first plot shows the weights on a plane parallel to the optical axis while the second plot shows the weights on a plane orthogonal to the optical axis. The discrimination in the viewing direction was only possible because several tracked landmarks were used for the importance weight calculation. To derive the plots, the magnitudes of the importance weights were interpolated.

New Landmark Initialization

The last step of the processing of an image is the initialization of new landmarks. If it was chosen to initialize a new landmark, the image is searched with the help of a feature detector to obtain new points for the initialization process.

Instead of searching the whole image, the search region is greatly reduced. The basic idea is to spread the features as much as possible as more distant features on the image plane will contribute better to the localization estimation. On the other hand, the result of the feature detector has to be considered as well. It might be the case, that no features can be detected in regions that are far apart from the rest of the features. Thus, we have chosen to divide the image with a rectangular grid. Whenever a new feature should be initialized, it is checked if the cells of the grid are already occupied by a visible feature. Among the set of all cells that are not occupied, one cell is randomly chosen and the feature detector is executed for this area. If a feature was detected in the selected cell and the Euclidean distance to all other features is below a threshold, it is initialized and added to the map of each particle. Upon an unsuccessful attempt, the state of the system is not changed for the iteration.

5.3.3 Landmark Lifecycle Management

The landmark lifecycle management implements strategies regarding the time point for the initialization of a new landmark and its deletion. While it is obvious that new landmarks have to be initialized when exploring previously unknown areas in order to generate a map of the environment, the strategy for deletion has to consider at what point in time a landmark does not contribute to an enhancement of the estimation anymore.

Initialization Strategy

Our strategy for the initialization has the aim to keep the number of visible features constant. As the complexity of the algorithm increases linear with the number of visible features, a too high count of those will lead to a computational time that prevents a real-time application. If the number drops too much, the obtained information for the localization estimation is not sufficient. In particular, the initialization process is triggered whenever the number of visible features drops below a certain threshold. This threshold is chosen to be as high as possible while keeping the real-time abilities of the software.

Deletion Strategy

A landmark is in choice for deletion whenever it either does not contribute to the estimation or might even lead to a degradation of the estimation process.

If a landmark is not visible anymore because of occlusion or changes in the scenery, a positive match is not possible. This means that the according region of interest is still searched image by image leading to an unnecessary computational effort. The same

happens if the feature detector failed and a landmark was initialized at a position that is not highly discriminating from the proximity. This will cause many ambiguous matches that are not used as a measurement for the estimation process. In both cases, the landmark should be discarded.

To determine the point in time for the deletion, we use a binary Bayes filter [TBF05] for each landmark that is able to incorporate negative information. Negative information here directly means an ambiguous or absence matching. It uses a single binary variable that is incremented whenever a correct matching was detected and decremented if not. Whenever this variable drops below a threshold, the landmark is regarded as instable and discarded. Furthermore, we have chosen to limit the maximum this variable can reach to speed up the response if the tracking to a certain landmark is lost.

5.4 State Evolution Model

The state evolution models describes the change of the system state between two time steps incorporating the control vector. As we use the measurements of the IMU as a control input, the state evolution model mainly consists of a strapdown algorithm and the addition of noise to cover the uncertainties of the measurements of the IMU.

According to the results of section 3.4, the complete state evolution model is:

$$\begin{pmatrix} \mathbf{t}_{m,k}^{(L)} \\ \dot{\mathbf{t}}_{m,k}^{(L)} \\ \mathbf{r}_{LM,k} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{I}T & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{t}_{m,k-1}^{(L)} \\ \dot{\mathbf{t}}_{m,k-1}^{(L)} \\ \mathbf{r}_{LM,k-1} \end{pmatrix}}_{\text{state transition}} + \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ \mathbf{C}_{LM}^{T}(\mathbf{r}_{LM,k-1})T & 0 & \mathbf{I}T \\ 0 & \mathbf{E}_{LM}^{-1}(\mathbf{r}_{LM,k-1})T & 0 \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{a}}_{k}^{(M)} + \mathbf{n}_{Acc} \\ \tilde{\boldsymbol{\omega}}_{k}^{(M)} + \mathbf{n}_{Gyro} \\ \mathbf{g}^{(L)} \end{pmatrix}}_{\text{incorporation of available measurements}} \tag{5.7}$$

where $\mathbf{I}$ is the identity, T the sampling rate, $\mathbf{E}$ the turn rate transformation matrix, $\mathbf{C}$ the rotation matrix, $\tilde{\mathbf{a}}$ and $\tilde{\omega}$ are the available measurements from the acceleration respectively turn rate sensors, $\mathbf{g}$ is the gravity vector and $\mathbf{n}_{Acc}$ as well as $\mathbf{n}_{Gyro}$ refer to the measurement noise. The rest of the variables are related to the platform state that was described in section 5.2.1.

Each time, data from the IMU is available, this state evolution model is used to update the set of particles. For each particle, another sample $\mathbf{n}_{Acc}$ and $\mathbf{n}_{Gyro}$ for the measurement noise is used.

We did not use the correction term of equation 3.28 that describes the change of the axes of the acceleration sensors between the measurements because the contribution to the calculation is marginal.

5.5 Measurement Model

The measurement model is needed to incorporate the measurements into the estimate of the platform state and the map. The basic measurement model describes, how the parameterization of a landmark $\mathbf{l}$ is related to a point on the image plane $\mathbf{p}^{(I)}$ given the platform state $\mathbf{x}$:

$$\mathbf{p}_l^{(I)} = h(\mathbf{l}, \mathbf{x}) \tag{5.8}$$

The solution is a combination of two functions, one function h_c to represent the landmark within the camera coordinate system and one function h_p for the projection onto the image plane. The chaining of these functions gives the point on the image plane:

$$\mathbf{p}_l^{(I)} = h_p(h_c(\mathbf{l}, \mathbf{x})) \tag{5.9}$$

The function for the projection is defined as

$$\mathbf{p}_l^{(I)} = h_p(\mathbf{p}_l^{(C)}) = \begin{pmatrix} p_{x,l}^{(I)} \\ p_{y,l}^{(I)} \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} p_{x,l}^{(C)} \\ p_{y,l}^{(C)} \\ p_{z,l}^{(C)} \end{pmatrix} \frac{1}{p_{x,l}^{(C)}} \tag{5.10}$$

according to the equation 3.4 of the already known pinhole camera model. To really gain a point on the image plane, the actual projection function is already included.

The function $h_c(\mathbf{l}, \mathbf{x})$ to relate the representation of a certain landmark to a point in the camera frame depends on the parameterization and is discussed in a later section.

5.5.1 Importance Weight Calculation

With the measurement model we are now able to compute the importance weights for each particle. The importance weights reflect the ratio between the target and the proposal distribution. To be able to calculate them, the map gets marginalized:

$$\omega_k^{[i]} = \alpha p(\mathbf{z}_k | \mathbf{x}_k^{[i]}) \omega_{k-1}^{[i]} = \alpha \prod_{f=1..F} \int_{\mathbf{l}_f^{[i]}} p(\mathbf{z}_f | \mathbf{x}_k^{[i]}, \mathbf{l}_f^{[i]}) p(\mathbf{l}_f^{[i]}) d\mathbf{l}_f^{[i]} \omega_{k-1}^{[i]} \tag{5.11}$$

where F is the number of available features of a measurement, $\mathbf{l}_f^{[i]}$ is the landmark of particle $[i]$ that is associated with the feature f and $\mathbf{z}_f$ is the according measurement. This means that the weight of each particle is the product of all the likelihoods of the measurements considering the platform state that particle represents and the certainty of the individual landmarks.

To actually calculate ω, we need a closed form solution for the integral involved in this equation. This is done by a first order Taylor approximation of the measurement model h. This step is similar to the calculation of the measurement covariance within the extended Kalman filter. The measurement covariance can be calculated by

$$\mathbf{M}_{k,l} = \mathbf{H}_{k,l}^T \mathbf{P}_{k,l} \mathbf{H}_{k,l} + \mathbf{Q}_k \tag{5.12}$$

where $\mathbf{H}_{k,l}$ is the measurement Jacobian for one landmark at time k considering one particular particle, $\mathbf{Q}_k$ the measurement noise and $\mathbf{P}_{k,l}$ is the covariance of the according landmark. This equation helps to project the uncertainty of the landmark onto the image plane. With its help and the estimation of the measurement $\hat{\mathbf{z}}_{k,f}$ that is derived by using the measurement model, the likelihood of a single measurement is given as:

$$\omega_{k,l} = \alpha |2\pi \mathbf{M}_{k,l}|^{-\frac{1}{2}} \exp^{-\frac{1}{2}(\tilde{\mathbf{z}}_{k,f} - \hat{\mathbf{z}}_{k,f})^T \mathbf{M}_{k,l}^{-1} (\tilde{\mathbf{z}_{k,f}} - \hat{\mathbf{z}}_{k,f})} \tag{5.13}$$

where $\mathbf{M}_{k,l}$ is the measurement covariance for a landmark l at time k and $\tilde{\mathbf{z}}_{k,f} - \hat{\mathbf{z}}_{k,f}$ is the difference between the predicted and measured position of the feature f that is associated with the landmark l.

5.5.2 Measurement Jacobian

For the calculation of the importance weights as well as for the parameterization of the Kalman filter that updates the state of a landmark, the Jacobian $\mathbf{H}$ of the measurement function is needed. This Jacobian depends on the parameterization of the landmark, the pinhole camera model and the momentary estimated state of the platform.

This is necessary as the measurement functions are approximated with a first order Taylor expansion in order to be usable for the Kalman equations as well as for the importance weight calculation:

$$\mathbf{p}_k^{(I)} \sim h(\hat{\mathbf{l}}_k, \hat{\mathbf{x}}_k) + \left. \frac{\partial h(\mathbf{l}, \mathbf{x})}{\partial \mathbf{l}} \right|_{\mathbf{l}=\hat{\mathbf{l}}_k, \mathbf{x}=\hat{\mathbf{x}}_k} \tag{5.14}$$

As the measurement Jacobian is a combination of several functions, the complete Jacobian is calculated with the chain rule:

$$\frac{\partial h}{\partial \mathbf{l}} = \frac{\partial h}{\partial h_c} \frac{\partial h_c}{\partial \mathbf{l}} = \mathbf{H}_p \mathbf{H}_c \tag{5.15}$$

Projection Jacobian

The projection Jacobian relates the change of a point in the camera frame to a change of its projection on the image plane. As we only work with undistorted points, the undistortion function is not part of the projection Jacobian. Instead, the undistortion happens right after the matching process. Nevertheless, it is also possible to work with the distorted versions of the points on the image plane but then the projection Jacobian has to be modified accordingly.

The projection Jacobian is

$$\mathbf{H}_p = \frac{\partial h}{\partial h_c} = \mathbf{K} \begin{pmatrix} 0 & 0 & 0 \\ \frac{-p_{y,l}^{(C)}}{(p_{x,l}^{(C)})^2} & \frac{1}{p_{x,l}^{(C)}} & 0 \\ \frac{-p_{z,l}^{(C)}}{(p_{x,l}^{(C)})^2} & 0 & \frac{1}{p_{x,l}^{(C)}} \end{pmatrix} \tag{5.16}$$

It is obvious that this Jacobian is highly depending on the distance of the landmark in the camera frame.

5.6 Landmark Parameterization

A landmark can be represented in different ways. Usually, the variables for the parameterization of a landmark refer to distances and angles that are represented in the lab frame to gain a global reference. Depending on the chosen parameterization, the landmark is more sensitive to small changes in the position of the platform what can lead to an underestimation of the covariances.

Depending on the representation, the function h_c to calculate the estimated mean position in the camera frame as well as $\mathbf{H}_c$, the Jacobian of this function, will change. Another representation also involves another form of initialization that has to be considered.

Two kind of parameterizations are introduced here: The XYZ parameterization and the inverse depth parameterization.

5.6.1 XYZ Parameterization

In the XYZ parameterization, the parameters of a landmark are just the coordinates within the Euclidean coordinate system of the lab:

$$\mathbf{l} = \mathbf{p}_l^{(L)} = \begin{pmatrix} p_{x,l}^{(L)} \\ p_{y,l}^{(L)} \\ p_{z,l}^{(L)} \end{pmatrix} \tag{5.17}$$

The transformation to the camera frame can be calculated with the known equations for transforming points between coordinate systems:

$$\mathbf{p}_l^{(C)} = h_c(\mathbf{l}, \mathbf{x}) = \mathbf{C}_{BC}(\mathbf{C}_{LB}(\mathbf{p}_l^{(L)} - \mathbf{t}_b^{(L)}) - \mathbf{t}_c^{(B)}) \tag{5.18}$$

where $\mathbf{C}_{LB}, \mathbf{t}_b^{(L)}$ and $\mathbf{C}_{BC}, \mathbf{t}_c^{(B)}$ are the translation and rotation between lab and body respectively body and camera. While the relation between the body frame and the camera frame is fixed, the translation and rotation from the lab to the body can be calculated out of the current state of the platform.

The Jacobian of this function is simply the chained rotation between those frames:

$$\mathbf{H}_c = \frac{\partial h_c(\mathbf{l}, \mathbf{x})}{\partial \mathbf{l}} = \mathbf{C}_{BC}\mathbf{C}_{LB} = \mathbf{C}_{LC} \tag{5.19}$$

evaluated at the momentary estimation of the rotation between the lab and the body.

The drawback of the XYZ representation is the enormous sensitivity to small changes of the platform position. During initialization of a landmark, the distance to the pinhole is unknown as it is not observable and thus has to be initialized with a high uncertainty.

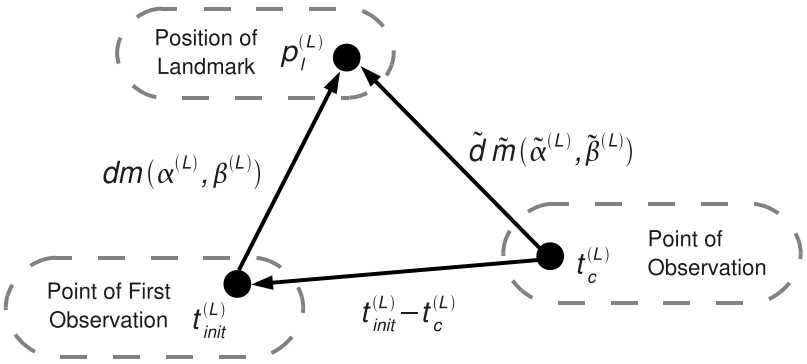


Figure 5.4: Illustration of the parameters involved in the inverse depth representation. The parameters of the landmark consist of the point of the camera at the first observation, two angles defining a unit vector to the landmark and the distance. Instead of directly estimating the distance its inverse is estimated.

This uncertainty will lead to a huge change of the position of the landmark during the update step when incorporating new measurements. This makes it very unlikely that the mean estimate of the position of the landmark will be in the correct place after some observations. Instead we could observe that the certainty in the distance component gets overestimated very fast, contributing to an overall instable system.

To solve this issue sometimes an approach called delayed feature initialization is used. This means, that before adding a landmark with the XYZ parameterization to the system state, it is first tracked for several frames. Once the distance can be estimated with certainty out of these measurements, the landmark gets actually initialized and is added to the system state. This can be done by tracking multiple hypothesis for the distance with a particle filter, by using a sum of Gaussians to track the distance or by triangulation [BGK06]. Whatever method is chosen, the drawback is that delayed initialization means that the landmark will not contribute to stabilize the system for several frames although the information of the landmarks position on the image plane is available. Furthermore, this means another additional computational challenge as with the particle filter or sum of Gaussian approach, several hypotheses for one landmark have to be evaluated in each iteration.

5.6.2 Inverse Depth Parameterization

The inverse depth parameterization developed by [MCD06] combines two aspects to gain better results for the final landmark estimate and to be able to immediately initialize a landmark. Instead of the distance its inverse is estimated and the position of the landmark is represented in relation to the first point of observation.

For the derivation in this section, the position of the camera is represented in the lab frame. The according translation can be calculated with the known equations for point transformation between coordinate systems:

$$\mathbf{t}_c^{(L)} = \mathbf{C}_{LB}^T \mathbf{t}_c^{(B)} + \mathbf{t}_b^{(L)} \tag{5.20}$$

where $\mathbf{C}_{LB}$ is the Rotation between the lab frame and the body frame and $\mathbf{t}_c^{(B)}$ is the translation of the camera within the body frame while $\mathbf{t}_b^{(L)}$ is the position of the body expressed in the lab frame.

In the inverse depth parameterization, a landmark is represented by a set of six parameters:

$$\mathbf{l} = \begin{pmatrix} t_{x,init}^{(L)} \\ t_{y,init}^{(L)} \\ t_{z,init}^{(L)} \\ \alpha^{(L)} \\ \beta^{(L)} \\ d \end{pmatrix} \tag{5.21}$$

where $\mathbf{t}_{init}^{(L)}$ is the translation of the camera, α and β are two angles describing the direction to the landmark and d is the distance from the projection center. All angles and coordinates refer to the point in time of the first observation and are given in the lab frame. Otherwise, a relation to later observations is impossible as no global reference is available. Even if the state vector for a landmark is composed of six parameters, only three are considered to be uncertain. As each particle follows a hypothesis of the platform state, the position of the camera during the first observation is assumed to be perfectly known. Thus, this parameter is not further estimated by a Kalman filter in our approach.

To actually calculate the position of a landmark in the lab frame, the following equation is used:

$$\mathbf{p}_l^{(L)} = \begin{pmatrix} t_{x,init}^{(L)} \\ t_{y,init}^{(L)} \\ t_{z,init}^{(L)} \end{pmatrix} + d \begin{pmatrix} \cos(\alpha^{(L)})\cos(\beta^{(L)}) \\ \sin(\beta^{(L)}) \\ \sin(\alpha^{(L)})\cos(\beta^{(L)}) \end{pmatrix} = \mathbf{t}_{init}^{(L)} + d\mathbf{m}(\alpha^{(\mathbf{L})}, \beta^{(\mathbf{L})}) \tag{5.22}$$

what defines a point starting from the first point of observation along a ray defined by α and β to a point with distance d.

Unfortunately, the distance is still not known and has to be estimated. As the distance might be between zero and infinity, a huge space has to be covered by this estimation. To overcome this issue, [MCD06] proposed to use the inverse of the distance as a parameter instead of the distance itself $\rho = \frac{1}{d}$. With this, the distribution of this variable is closer to a Gaussian distribution as if using the distance directly. Notice, that it is not possible to use this concept with the XYZ representation as the distance is not explicitly modeled.

As the position of the platform changes, an equation has to be developed to relate another position of the platform to the initial parameterization. Given the first point of observation $\mathbf{t}_{init}^{(L)}$, an initial unit vector $\mathbf{m}(\alpha, \beta)$ and the distance d, the following relation to another point for observation $\mathbf{t}_c^{(L)}$, a vector $\mathbf{m}(\tilde{\alpha}, \tilde{\beta})$ and the distance $\tilde{d}$ is fulfilled, see figure 5.4:

$$\mathbf{t}_{init}^{(L)} + d\mathbf{m}(\alpha^{(L)}, \beta^{(L)}) = \mathbf{t}_c^{(L)} + \tilde{d}\mathbf{m}(\tilde{\alpha}, \tilde{\beta}) \tag{5.23}$$

This can be written as

$$u\mathbf{m}(\tilde{\alpha},\tilde{\beta}) = \frac{1}{d}(\mathbf{t}_{init}^{(L)} - \mathbf{t}_{c}^{(L)}) + \mathbf{m}(\alpha^{(L)},\beta^{(L)}) \tag{5.24}$$

where u refers to an unknown scalar. As the camera cannot observe any distances, this scalar will not alter the measurement in any way and cancel out when using the measurement model to get the point on the image plane. It is obvious that the distance expressed with this function has no influence if the difference between the initial point of observation and another one tends to zero.

As the ray to the landmark is defined within the coordinate system of the lab frame with this equation, the actual measurement function within the camera frame includes the rotation between both frames. With the inverse depth for the distance, the complete measurement function to relate a landmark with inverse depth parametrization to a point $\tilde{\mathbf{p}}_{l}^{(C)}$ on a ray starting at the optical center is:

$$\tilde{\mathbf{p}}_{l}^{(C)} = h_c(\mathbf{l},\mathbf{x}) = \mathbf{C}_{LC}\left(\rho(\mathbf{t}_{init}^{(L)} - \mathbf{t}_{c}^{(L)}) + \mathbf{m}(\alpha^{(L)},\beta^{(L)})\right) \tag{5.25}$$

As three out of the six parameters for the landmark representation are considered to be perfectly known and thus are not further estimated, the according extended Kalman filter is only initialized with the parameter for the inverse distance and the two angles defining the ray at the first point of observation:

$$\mathbf{l}_{EKF} = \begin{pmatrix} \alpha^{(L)} \\ \beta^{(L)} \\ \rho \end{pmatrix} \tag{5.26}$$

This results in the following Jacobian for the measurement function if using inverse depth parametrization:

$$\mathbf{H}_c = \frac{\partial h_c}{\partial \mathbf{l}_{EKF}} = \mathbf{C}_{LC}\begin{pmatrix} -\sin(\alpha^{(L)})\cos(\beta^{(L)}) & -\cos(\alpha^{(L)})\sin(\beta^{(L)}) & t_{x,init}^{(L)} - t_{x,c}^{(L)} \\ 0 & \cos(\beta^{(L)}) & t_{y,init}^{(L)} - t_{y,c}^{(L)} \\ \cos(\alpha^{(L)})\cos(\beta^{(L)}) & -\sin(\alpha^{(L)})\sin(\beta^{(L)}) & t_{z,init}^{(L)} - t_{z,c}^{(L)} \end{pmatrix} \tag{5.27}$$

For the initialization, the initial position of observation has to be known, a ray towards the landmark has to be calculated and an initial estimation of the inverse depth is needed.

As each particle knows the platform state without uncertainty, the point of initial observation is just the camera position calculated with the current platform state hypothesis of the particle:

$$\mathbf{t}_{init}^{(L)} = \mathbf{t}_{c}^{(L)} \tag{5.28}$$

The initial angles are

$$\begin{aligned} \alpha &= \arctan2(\tilde{p}_{z,l}^{(L)}, \tilde{p}_{x,l}^{(L)}) \\ \beta &= \arctan2(\tilde{p}_{y,l}^{(L)}, \sqrt{(\tilde{p}_{x,l}^{(L)})^2 + (\tilde{p}_{z,l}^{(L)})^2}) \end{aligned} \tag{5.29}$$

where $\tilde{\mathbf{p}}_l^{(L)}$ is a point on the ray towards the landmark in the lab frame and can be calculated by using the inverse projection for the pinhole camera model with an additional rotation to the coordinate system of the lab frame. As the length of the ray is unimportant, an arbitrary value for the distance can be chosen when using the inverse projection. The complete transformation with an assumed distance of $x_c = 1$ is:

$$\tilde{\mathbf{p}}_l^{(L)} = \begin{pmatrix} \tilde{p}_{x,l}^{(L)} \\ \tilde{p}_{y,l}^{(L)} \\ \tilde{p}_{z,l}^{(L)} \end{pmatrix} = \mathbf{C}_{LC}^T \begin{pmatrix} 1 \\ y_c \\ z_c \end{pmatrix} = \mathbf{C}_{LC}^T \begin{pmatrix} 0 & 0 & 1 \\ \frac{1}{f_{x,px}} & \frac{-s}{f_{x,px}f_{y,px}} & \frac{s f_{oy,px}}{f_{x,px}f_{y,px}} - \frac{f_{ox,px}}{f_{x,px}} \\ 0 & \frac{1}{f_{y,px}} & -\frac{f_{oy,px}}{f_{y,px}} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad (5.30)$$

where the parameters for the inverse projection are explained in section 3.2.1. Before the inverse projection occurs, the coordinates on the image plane are corrected according to the distortion model.

For the initialization of the inverse distance, we set the initial estimation according to the proposed values of [MCD06] to

$$\rho = 0.5 \left[\tfrac{1}{m}\right] \qquad \sigma_\rho = 0.25 \left[\tfrac{1}{m}\right] \quad (5.31)$$

Notice that this choice for the initialization includes infinity within the 2σ bound and the shortest distance that can be expressed within this bound is $1m$:

$$\left(\frac{1}{\rho - 2 * \sigma_\rho}, \frac{1}{\rho + 2 * \sigma_\rho}\right) = (\infty, 1)\,[m] \quad (5.32)$$

5.7 Complexity of Visual SLAM

The complexity of the visual SLAM approach depends on the number of particles N, the maximum number of tracked features F within one image, the number of Landmarks L in the map, the size of one patch P and the region of interest R that is searched for an according match.

The critical step in this approach is the calculation of visible landmarks. As the whole map of each particle is searched to derive the features that are actually tracked, the complexity of this step is $O(NL)$. This disagrees with Thrun's claim [TBF05] that the complexity of the factored solution of the SLAM problem is $O(N\log(L))$. This is caused by two constraints of him: On the one hand he only considers one measurement at a time and on the other hand he assumes that the according landmark for the single measurement can be found within logarithmic time. With another representation of the map, the complexity of this step can be reduced in our approach as well.

To calculate the regions of interest, the uncertainty of each feature on the image plane has to be examined what leads to the complexity of $O(NF)$. The matching process depends on the number of features, the calculated search area and the size of the patches. This complexity is $O(R^2P^2F)$ assuming that the search area as well as the patch is square in the dimensions and that R respectively P represent the length in one dimension. These steps are not dependent on the number of particles.

To calculate the importance weights, the likelihood of the matching result for all features has to be evaluated per particle. The resulting complexity is $O(NF)$. During the resampling step, only the references of the particles are copied. With the systematic resampling approach, the complexity of this step is $O(N)$. Although the complexity in the step for updating the landmarks is reduced as it is postponed after the resampling, the upper bound is $O(NF)$. This is caused by the fact that for each tracked feature that is successfully matched, the according landmark in the maps of all particles has to be updated. The complexity of the landmark initialization depends on the size of the regions that are used for the detection of new features and the area used to build the structure tensor. The resulting complexity is $O(T^2A^2)$ where T describes the length within one dimension of the square structure tensor and A refers to the length of one dimension of the square search area. Even if this complexity seems high it has to be considered that this step is not necessary for each iteration. Only if the number of features drops below a certain threshold, the detection of new features is triggered.

The state evolution model results in a complexity of $O(N)$ as the platform state within each particle has to be updated according to the measurements of the inertial measurement unit and the sampled noise hypothesis.

6 Implementation

This chapter is devoted to describe the particular hardware and software used for the experimental part. Certain aspects considered during the implementation of the software are highlighted. Furthermore, the approach to map the measurements of the tracking system to the estimated values of the visual SLAM algorithm is shown.

6.1 Hardware Aspects

6.1.1 Inertial Measurement Unit

We used an XSens MTx sensor as inertial measurement unit. This sensor features three acceleration sensors, three turn rate sensors and three magnetometers. The unit is able to provide the data with a sampling rate up to 100Hz and offers on-board processing. The sensor itself is connected via a serial to USB converter to a computer. The manufacture offers drivers and examples for Linux and Windows.

For the experiments, we used the calibrated data from the turn rate sensors and acceleration sensors with a sampling frequency of 60Hz. The calibration process includes a rough compensation of bias evolution due to temperature changes and a correction of misaligned sensor axes. This calibration was done by the manufacture and the parameters are stored into non-volatile memory of the device. The calibrated data is not to be confused with the processed data, that offers an attitude and position estimation from a fusion of the magnetometer, accelerometer and turn rate data. The magnetometer readings were not used for our experiments at all.

6.1.2 Camera

As image device, we used a monochrome Chameleon camera from Point Grey Research. This device offers an image sensor with a high resolution of 1296x964 pixels and a compact and lightweight form factor. It is connected to a computer via a USB 2.0 port. The manufacture offers with its driver framework a functionality equal to an device connected via IEEE 1394. This means that the camera parameters can be controlled easily within a software. Drivers and examples are offered for Windows as well as for Linux.

For the experiments, we captured images with a rate of 30 frames per second in a resolution of 640x480 pixels. The data format was set to a resolution of one byte per sensor pixel. To avoid effects of the automatic adjustment mechanism of the camera, we fixed all parameters except the gain value.

Furthermore, we used a lens with a fixed focal length of 6mm. The combination with the camera offered a field of view of approximately 44 degrees in the horizontal direction and 33 degrees in the vertical direction.

6.1.3 Tracking System

We used a Vicon Bonita tracking system to gain ground truth reference data. This tracking system consists of several cameras that can detect special infrared markers. The provided software of the manufacture can combine the measurements from the cameras to estimate the attitude and position of any objects with markers attached to them. The system offers a capture speed up to 240 frames per second and an easy deploy and calibration process.

For the experiments we used the system with a capture rate of 60Hz to be able to compare the results of the implementation of the visual SLAM algorithm with a reference.

6.2 Software Aspects

The core components of the software are written in the programming language Java. It was chosen due to its cross-platform compatibility. Java software is compiled to an intermediate language called Java byte code and a Java virtual machine is used to actually execute the program. Every platform offering an implementation of a Java virtual machine is able to execute Java byte code. The concepts exceed the behavior of an interpreted language by far by the usage of an just in time compiler that optimizes chunks of Java byte code to a certain platform and by the usage of platform dependent implementations of the virtual machine with optimized libraries. These concepts together offer a speed comparable to other approaches producing machine dependent code.

The aim was to design a high flexible software that is able to process measurements in real-time and in a post-processing mode. To achieve this requirement, the software offers a modular design to be able to evaluate different approaches. The implementations to process the data as well as the channels to write or read the data are changeable during runtime.

6.2.1 Data Acquisition

The low-level communication to the different sensors is implemented in the programming language C++ because the manufacturers for the camera, the inertial measurement unit and the tracking system do not offer support for Java. After the data has been read from the sensors, it is provided via a TCP/IP server to be used by the Java program. The Java implementation establishes a connection to the TCP/IP servers and offers the data in an event driven way. This means that classes can register to be called whenever data from the sensors is available.

This client-server concept offers the possibility to separate the locations for data acquisition and processing. Due to weight and energy restrictions of micro air vehicles, the

abilities of on-board processing are very limited. A client-server concept offers the possibility of processing the data at another location, where less restricted requirements for the hardware apply.

Furthermore, classes were implemented to write the data to a file. The stored data in these files can be used to produce input to the Java implementation. A synchronization mechanism will trigger the certain events for the arrival of data in the same order as if they were triggered with the sensors connected.

Synchronization

When reading from files, the timestamps of the available data packets are compared to each other and the data is provided in the correct order. At every step in time, only one packet for each individual sensor is hold in memory. After processing a certain packet, it is marked to be renewed in the next iteration.

For the real-time processing of data, the behavior of the synchronization is different. The data of the sensors is provided in the order of arrival. Due to the unpredictable execution time of a Java implementation, the system has to be able to detect and handle situations where more data is available than can be processed. As the processing of images takes longer than the incorporation of data from the inertial measurement unit, packets from the camera get discarded when the system runs out of resources. Another aspect of discarding data from the camera instead of the inertial measurement unit is provided by the fact, that the state evolution model is driven by the packets from the IMU and thus will keep track of the uncertainty. If packets from the inertial measurement unit are discarded, the uncertainty of the system is not modeled in a correct way what finally will lead to instabilities.

6.2.2 Modular Engine

The engine of a system defines the core components that manipulate the data. The engine of our application was designed to be as flexible as possible. To fulfill this requirement, a modular implementation concept was chosen. The engine itself is a very lightweight software component that only passes the data between different modules. This gives us the possibility to define the behavior at the start of the program or even to change certain implementations while the program is running without losing any data.

6.2.3 Real-Time Optimization

For real-time uses it is very important that the system does not get instable at any point in time and that the localization and mapping data is available with a constant rate. Instability in this context means that the system is not able to process all the incoming data anymore. While we already considered this problem with the synchronization strategy, we also have to make sure that the output of our own system is available in time to keep control of the flying platform. To reach this aim, we implemented techniques to reduce the latency of the processing.

Smart Object Store

Using Java, the most difficult factor one has to consider is the garbage collector when designing a program with real-time scope. Although the user is able to allocate objects, no explicit possibility for deallocation exists. Instead, a special program called garbage collector keeps track of the references to objects and will free the occupied memory at an unspecific point in time if all references to a certain object were dropped. The garbage collector is part of the Java virtual machine and no possibility exists to switch the behavior to an explicit allocation/deallocation concept. Although the garbage collector works very reliable, the point of execution is undefined. Thus, unpredictable latency will be introduced while the program is running.

To overcome this behavior, we implemented a smart object store to provide an own memory management with defined latency and a lazy copy implementation. The store is initialized with a certain amount of objects that are all allocated at the start of the system with the provided Java functions. During the runtime of a program, the object store serves allocate and release requests. For the allocation of an object, one object of the store is taken and reseted to a pre-defined state. When the program finished using a certain object, it can release the object what means that it is put back into the object store where it is available for further usage. This concept preserves a triggering of the garbage collector, as the objects are referenced all the time. They are either available in the object store for re-usage or in use by the program.

Lazy copying is another concept implemented by the object store that helps to reduce the amount of unnecessary data copy. For vSLAM, it is often needed to duplicate huge amounts of data e.g. at the resampling step, where a whole particle including its map with many landmarks has to be copied. Instead of directly copying the whole data, we use lazy copying. With this concept, the state of an object only gets copied if several references to it exist and the state of the object is going to be changed. To implement lazy copying, we use handles to objects that offer methods to duplicate or to start modifying an object. The represented object is stored into a holder that keeps tracks of the number of handles pointing to it. If more than one handle is pointing to an object that is signaled to be modified, then a deep copy of the object's data is started, what means, that a similar object from the object store is allocated and the data is copied to this new object. This behavior is implemented in the handles, the smart object store and the holder of objects. As the access to objects always happens via a handle, the object store needs no further user interaction.

Because of the resampling and the involved copying of the map, many particles will have the same state of certain landmarks, especially if the landmarks have been out of sight for a longer time. With the concept of lazy copying, only one object to represent those landmarks is needed that can be shared among all particles. This means, that the memory need is reduced greatly.

Data Passing

We also tried to optimize the structure of the data that is passed between the individual modules. The aim was to reduce the overhead that is needed to manage the data. The

result is an extensive use of doubly linked lists as they offer constant time for adding and removing an element at an arbitrary place in the list and constant time for accessing an element at the start of the list.

For visual SLAM, the order of processing the individual landmarks or features usually does not matter. For this reason, we are not limited by the fact, that a list has a linear complexity for a random access. The basic representation is a list of features with a list of all the landmarks attached to a single feature, that were initialized because of it. Once this list was created during the step of determining the visible features, it can be used in all further steps without any modification.

Coordinate Frames

Each coordinate frame is represented by an object that can be chained to another one. This reference reflects the parent-child relationship of frames that are defined within each other. In the end, the frames build up a tree with the laboratory frame as the root. For the easy transformation of points between those frames, a parameter can be passed to the methods of the objects, that defines how many levels downwards, if transforming from child to parent, or upwards, if transforming from parent to child, a certain point should be transformed. This concept allows easy usage and extension of the different coordinate frames without the need to deal with the individual implementation of the functionality of a frame. Furthermore, a caching mechanism is used, that allows to fix a rotation among several frames. Usually, a point is passed between all the frames and transformed until the desired level of transformation is reached. With the caching mechanism, the computational time is reduced as the relation between the frames is only evaluated once and afterwards re-used.

6.3 Tracking System

The tracking system gives us pose information of an object defined by a set of markers. These information include the position as well as the attitude of an object. The measurements give the relation between the frame zero, defined during the initialization of the tracking system, and the frame tracking, that is attached to the particular platform. During the initialization process of the tracking system, a reference object is used to define the attitude as well as the origin of the coordinate frame zero, where the measurements take place. Thus, to be able to incorporate the information of the tracking system as the ground truth for the experimental part, the measurements have to be transformed to match the estimation of the vSLAM system. Notice, that we have now two coordinate frames that are attached to the body: One from the tracking system called tracking frame and one from the vSLAM estimation process called body frame.

6.3.1 Tracking System Coordinate Frames

For the tracking system, two more frames are introduced:

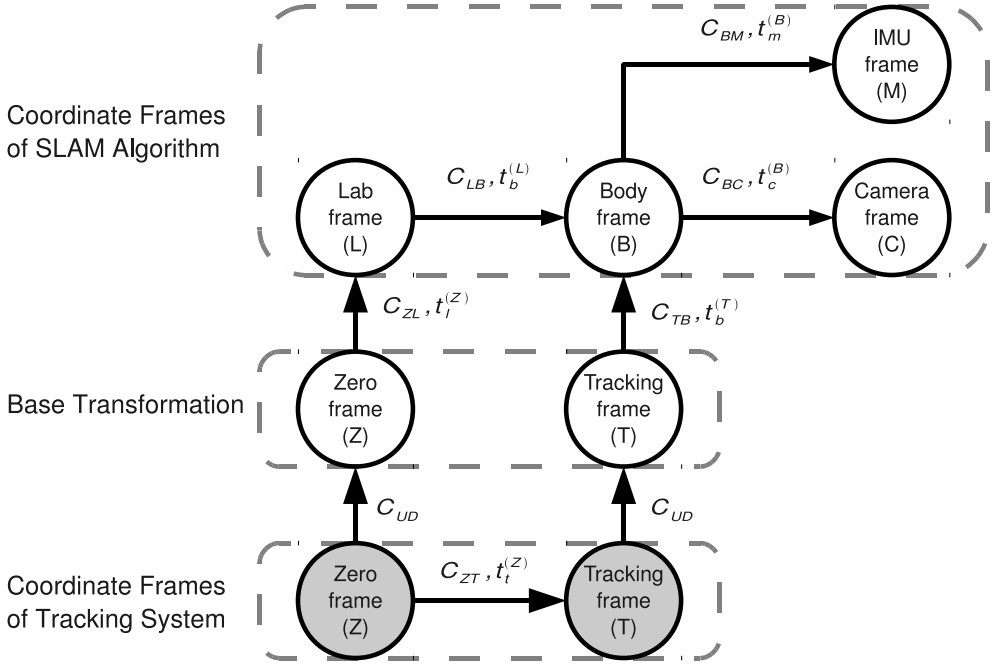


Figure 6.1: Illustration of the coordinate systems used in the visual SLAM algorithm, the coordinate systems of the tracking system and the relations between them. The zero frame defines the origin of the coordinate system of the tracking system while the origin of the tracking frame is within the volume of markers that are attached to an object. Because the tracking system uses another definition of the coordinate systems than the SLAM algorithm, a base transformation between both systems is necessary.

- *Zero frame* $^{(Z)}$
  The zero frame is not part of the SLAM algorithm but used to describe measurements of the tracking system. The frame zero defines the origin and initial attitude of the coordinate system, that the tracking system uses to gain measurements of a certain object that should be tracked. If the tracking system is used, the lab frame becomes a child of the zero frame. This rotation and translation is referred to as $\mathbf{C}_{ZL}$ and $\mathbf{t}_l^{(Z)}$
- *Tracking frame* $^{(T)}$
  To use the tracking system a device has to be equipped with markers. The origin of the tracking frame is in the center of these markers. The attitude is aligned to be as equal to the axis of the body frame as possible although a complete match is not possible and has to be considered when interpreting the measurements. The tracking system yields measurements of the tracking frame in reference to the zero frame. Thus, the tracking frame is child of the zero frame and the translation and rotation between those is written as $\mathbf{C}_{ZT}$ and $\mathbf{t}_t^{(Z)}$

As the axes of the tracking system are defined different in comparison to the used coordinate systems in the SLAM implementation, the axes have to be transformed to a

similar system before those calculations are possible. This is modeled with the rotation matrix $\mathbf{C}_{UD}$.

6.3.2 Using the Attitude Information

To use the attitude information, we define the lab frame within the frame zero of the tracking system. The frame tracking is also defined within the frame zero. Furthermore, we have to incorporate the fact, that the tracking system is using a coordinate system with an upwards directed z-axis, while the vSLAM system uses a coordinate system with a downwards directed z-axis. Downwards and upwards is defined by the direction of the gravity vector.

The relationship between the frames in the tracking system and the frames of the vSLAM system can be expressed as:

$$\mathbf{C}_{LB}\mathbf{C}_{ZL}\mathbf{C}_{UD} = \mathbf{C}_{TB}\mathbf{C}_{UD}\mathbf{C}_{ZT} \tag{6.1}$$

where $\mathbf{C}_{UD}$ is the matrix of the base transformation, $\mathbf{C}_{ZL}$ defines the orientation between the zero frame and the lab frame, $\mathbf{C}_{LB}$ is the estimated orientation of the body in the lab frame, $\mathbf{C}_{ZT}$ is the orientation of the body measured by the tracking system in reference to the zero frame and $\mathbf{C}_{TB}$ models the misalignment of the axes between the tracking frame and the body frame.

The very last orientation matrix, $\mathbf{C}_{TB}$, is needed because the axes of the two coordinate systems attached to the platform, the tracking frame and body frame, are not aligned equally. The orientation matrix is also called misalignment matrix. This misalignment can not be calculated out of a single snapshot of the system. Instead, the attitude of the platform has to change to be able to estimate correct values. In our case, we estimated this misalignment matrix in a post-processing step.

To use the measurements to compare it to the estimation of the tracking system, the equation can be reordered as follows

$$\mathbf{C}_{LB} = \mathbf{C}_{TB}\mathbf{C}_{UD}\mathbf{C}_{ZT}\mathbf{C}_{UD}^{T}\mathbf{C}_{ZL}^{T} \tag{6.2}$$

what can be directly compared to the estimation of the visual SLAM system. An unknown factor is still the orientation of the lab frame that is now defined in the zero frame. To calculate this orientation, we use that fact, that we know the relation between the lab frame and the body frame after initialization of the visual SLAM system. Pitch and roll of the system are initialized according to the measured gravity vector, while the yaw angle is initialized to zero. Thus, the orientation of the lab frame can be calculated as:

$$\mathbf{C}_{ZL} = \mathbf{C}_{LB,Init}^{T}\mathbf{C}_{TB}\mathbf{C}_{UD}\mathbf{C}_{ZT,Init}\mathbf{C}_{UD}^{T} \tag{6.3}$$

where $\mathbf{C}_{ZT,Init}$ donates the first measurement of the tracking system and $\mathbf{C}_{LB,Init}$ is the initial attitude estimation of the visual SLAM system.

Combining equations 6.2 and 6.3 yields the formula that we used to transform the measurements from the tracking system to the visual SLAM algorithm:

$$\mathbf{C}_{LB} = \mathbf{C}_{TB}\mathbf{C}_{UD}\mathbf{C}_{ZT}\mathbf{C}_{ZT,Init}^{T}\mathbf{C}_{UD}^{T}\mathbf{C}_{TB}^{T}\mathbf{C}_{LB,Init} \tag{6.4}$$

6.3.3 Using the Translation Information

Similar to the incorporation of the attitude information, the relation across the different coordinate frames of a point $\mathbf{p}^{(B)}$ that is defined in the body frame is given as:

$$\mathbf{C}_{UD}^{T}(\mathbf{C}_{ZL}^{T}(\mathbf{C}_{LB}^{T}\mathbf{p}^{(B)} + \mathbf{t}_{b}^{(L)}) + \mathbf{t}_{l}^{(Z)}) = \mathbf{C}_{ZT}^{T}(\mathbf{C}_{UD}^{T}(\mathbf{C}_{TB}^{T}\mathbf{p}^{(B)} + \mathbf{t}_{b}^{(T)})) + \mathbf{t}_{t}^{(Z)} \tag{6.5}$$

where $\mathbf{t}_{b}^{(L)}$ is the translation between the lab frame and the body frame, $\mathbf{t}_{l}^{(Z)}$ between the zero frame and the lab frame and $\mathbf{t}_{t}^{(Z)}$ is the actual measurement of the tracking system, the translation between the zero frame and the tracking frame. The translation $\mathbf{t}_{b}^{(T)}$ is used to model the misalignment of the translation between the tracking system and the visual SLAM estimation.

As we want to measure the point defined by the origin of the body frame, $\mathbf{p}^{(B)}$ becomes zero. With the knowledge, that the origin of the lab frame equals the origin of the body frame when the system is initialized, we can express the translation between the lab and the body frame with the measurements from the tracking system as:

$$\mathbf{t}_{b}^{(L)} = \mathbf{C}_{ZL}\mathbf{C}_{UD}(\mathbf{C}_{ZT}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)} - \mathbf{C}_{ZT,init}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)} + \mathbf{t}_{t}^{(Z)} - \mathbf{t}_{t,init}^{(Z)}) \tag{6.6}$$

where $\mathbf{t}_{t,init}^{(Z)}$ is the measured translation of the tracking system when the system gets initialized. The detailed derivation is described in appendix A.2.

7 Experimental Results

For the experiments, we built a simple platform that should simulate the behavior of a flying quadrotor. This platform was equipped with the camera, the inertial measurement unit and markers to be able to follow the trajectory with the tracking system. The translation between the individual devices was chosen to reflect the positioning on a real quadrotor. To obtain measurements, we used an artificial scene consisting of several boxes where we attached features to.

This chapter describes the results of these experiments and the derived conclusions.

7.1 Experiment Description

For the experiment, we put four boxes together in an “U”-shape like visible in figure 7.1. The sides facing inwards the “U” were prepared with papers showing different images and graphics in order to be usable as features. Other papers were attached to the floor as well. The system was started outside the formation of the boxes facing into the center. It was moved for 120 seconds while keeping the field of view inside the formation of boxes. We captured the data and evaluated different parameters in a post-processing step. The following discussion is based on an evaluation with 500 particles and up to 24 landmarks that were tracked simultaneously.

As the attitude estimation differs from the true values up to two degrees, the assumed noise for the acceleration sensors was set to a relative high value in order to obtain a stable system. This leads to a loss of accuracy in the position estimation and map building.

7.1.1 Map

Figure 7.2 shows a top and a side view of the point estimation of the landmarks at the end of the experiment with the boxes used for the experiment as reference. The position of the landmarks was directly taken from the estimation without any further scaling, rotation or translation. As each particle tracks an own hypothesis of the map, one has to be chosen to be drawn. This figure represents the map of the particle with the highest importance weight at the last time step. It is not true in general that this particle has the most accurate hypothesis of the map because the history is not considered in this selection. Nevertheless, it offers a clue for the momentary quality of the map.

It can be seen in the first illustration that the whole map is rotated around the z-axis. As the system has no reference for the initialization of this axis, the rotation depends only

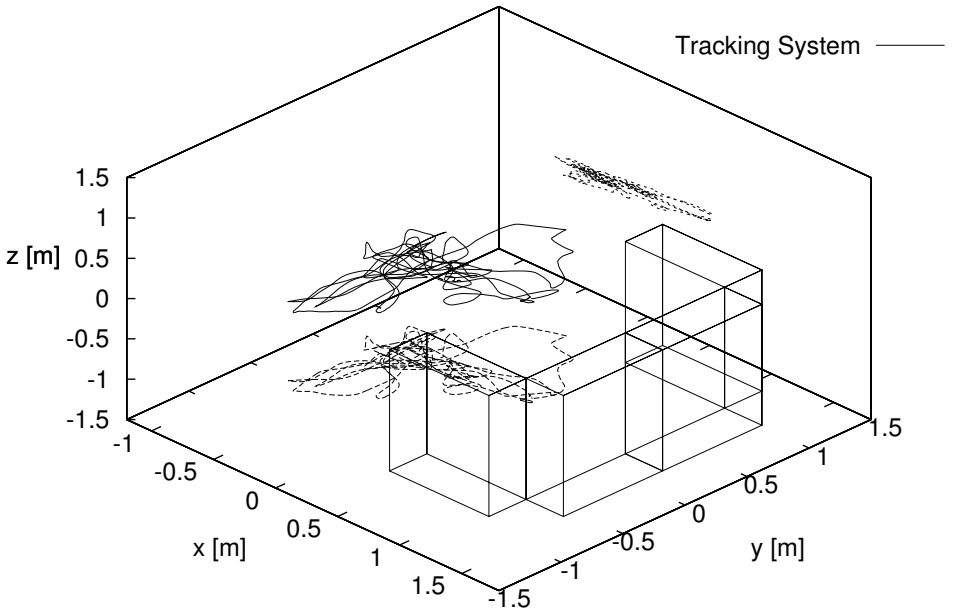


Figure 7.1: Illustration of the setup of the experiment. We used four boxes that formed an “U” and attached papers with features onto them. These are illustrated by the four wire-frame boxes. The trajectory reconstructed from the data of the tracking system is visible where the time for the whole movement was 120 seconds. The visible field of the camera was kept inside the formation of the boxes. The position, size and arrangement of the boxes were measured in order to be able to include them in the illustration.

on the direction of the optical axis when the system starts. Furthermore, when rotating the map to match the drawing of the boxes, it is visible that the scaling estimation of the map seems to be wrong as they are within the boxes. As these are solid objects, no landmark should be seen within them. As the distance is not observable and the acceleration sensor was modeled with a very high noise level, it is clear that the distance cannot be estimated with high accuracy. Instead, the distances will converge to a value influenced by the prior during the initialization of the individual landmarks. The landmarks used to derive this plot were landmarks with inverse depth initialization that were initialized according to equation 5.31. Thus it is noticeable that the landmarks converged into the correct direction but did not reach the true distance. Once a landmark has a wrong distance for any reason, it will lead to a scaling factor of the whole map as long as the measurements from the acceleration sensors are not accurate enough to prevent this.

The second plot shows a view from the side into the scene. The boxes were standing on the floor that is not visualized in this plot. It is noticeable that the most bottom landmarks are in one line as well as the right most landmarks. This is a clue that the system was able to estimate the roll and pitch angles very accurate as otherwise a tilt of these

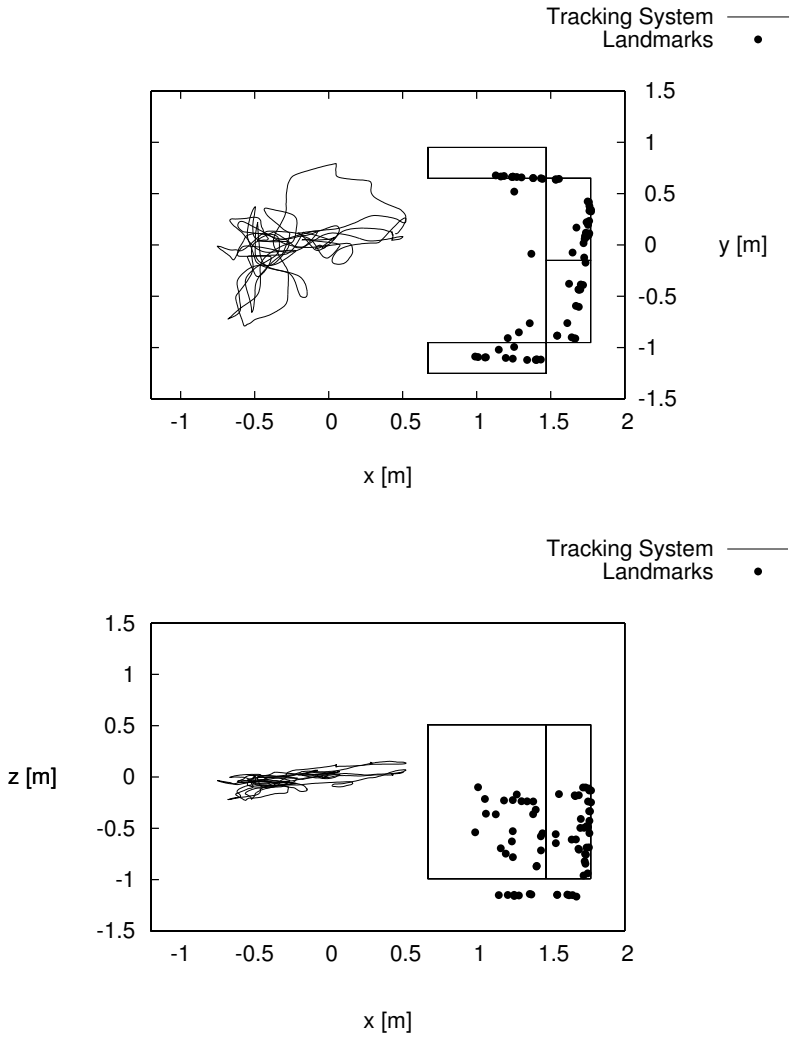


Figure 7.2: A top and side view of the experiment setup already presented in figure 7.1. The landmarks derived from the estimation process were added to this illustration.

lines would have been the result. This is caused by the influence of the gravity vector on the acceleration measurements of the inertial measurement unit. If the attitude is wrong, the hypotheses get discarded within several frames because they accelerate in a direction where they will get a very low importance weight as their position and attitude does not match the measurements of the feature detector anymore. This means that only those particles will survive a longer time and are able to build a good map, that have an accurate estimation history of the roll and pitch angles. On the other hand, a translation of the whole map in the direction of the negative z-axis is visible. As the floor can be visualized by extending the bottom lines of the boxes, the bottom most landmarks seem to be under the floor. First it has to be distinguished if it is a true translation or a scaling of the whole map. In this case, it is a scaling that was already

seen in the top view. When correcting all coordinates by a specific factor, the landmarks match almost exactly the border of the boxes.

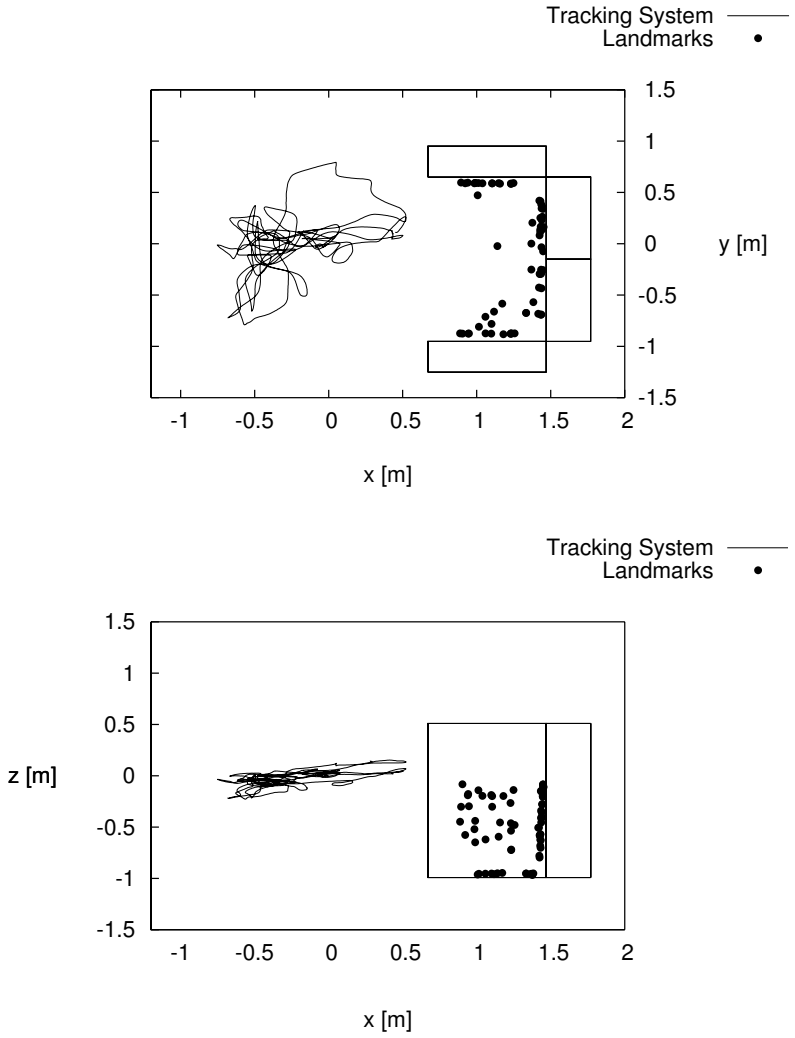


Figure 7.3: Top and side view of the experiment with the landmark similar to figure 7.2. The estimated positions of the landmarks are corrected for scaling and rotation.

Figure 7.3 shows the estimation of the map that is corrected by a constant scaling factor and the rotation around the z-axis. It can be seen that the position of the landmarks match the shape of the boxes very well. To calculate the factor for the scaling, the trajectory measured by the tracking system was compared to the estimation of the SLAM algorithm. The distance of the vectors between the start position and the estimation of the platform position respectively the measured position for one time step was summed up and divided in the end to get a scaling factor. This is just an approximation but has the advantage that it is not sensitive to errors in the rotation between the both trajectories of the SLAM and the tracking system. The same scaling factor was used for

the comparison of the estimated and measured positions of the platform in the later sections.

The points in the middle of the “U”-shape of the boxes belong to landmarks represented by papers that were arranged on the floor. Thus, these are not erroneous estimations of landmark positions.

7.1.2 Position

The plots 7.4 show the position of the platform over the time. The estimated position with the help of the SLAM algorithm as well as the reference position of the tracking system are visible. The plots start from the position zero because no information about the initial position is available for the estimation process and this point was arbitrary chosen to be the start point. The tracking system measures positions relative to a predefined point within the volume that it can observe. The coordinates of the tracking system were transformed to match the representation of the SLAM algorithm like described in section 6.3. The scaling was adjusted according to the same scaling factor already discussed in section 7.1.1. No further adjustments to the position or rotation were made.

When putting the focus onto the plots showing the differences in the position, it is visible that the highest difference is present in the z-coordinate followed by the y- and x-coordinate. This is counter-intuitive as the z-axis should be the most stable one when incorporating measurements from the inertial measurement unit. This is based on the different effects on the axes of the gravity vector caused by an erroneous attitude. The effect on the measurements for the axis parallel to the gravity vector is considered to be least influent, see section 3.4.3. We assume that this error is caused by the limited movement of the platform in the z-axis. As the movement is small, the measurements do not lead to a high discrimination in the z-coordinate for the localization process. This is also visible when comparing the absolute estimation of the SLAM algorithm with the absolute reference data for the z-axis. The estimation does hardly follow the movements measured by the tracking system. This is also an evidence for missing observability of the translation along this axis.

Also counter-intuitive is the comparison of the difference in the x-axis with the y-axis. This is the axis almost parallel aligned to the optical direction of the camera when the system started. As the camera is a projective device and can not observe distances, translations along this axis are not discriminated very well by the movement of the features on the image plane. Although the attitude of the platform was changed, the viewing direction was kept inside the boxes during the whole experiment. We think this is caused by the relative small distance of the camera to the landmarks. The smaller this distance, the more the features on the image plane will move and contribute to the localization process. Points at infinity do not change their angle when moving towards them. We think, that an experiment with a higher distance to the landmarks will better proof the effect of the unobservable distance.

If the result of the plots 7.4 are compared to a solution that uses only a strapdown algorithm, the improvement is enormous. By just incorporating the measurements according to a standard strapdown algorithm, see section 3.4, the error in the position

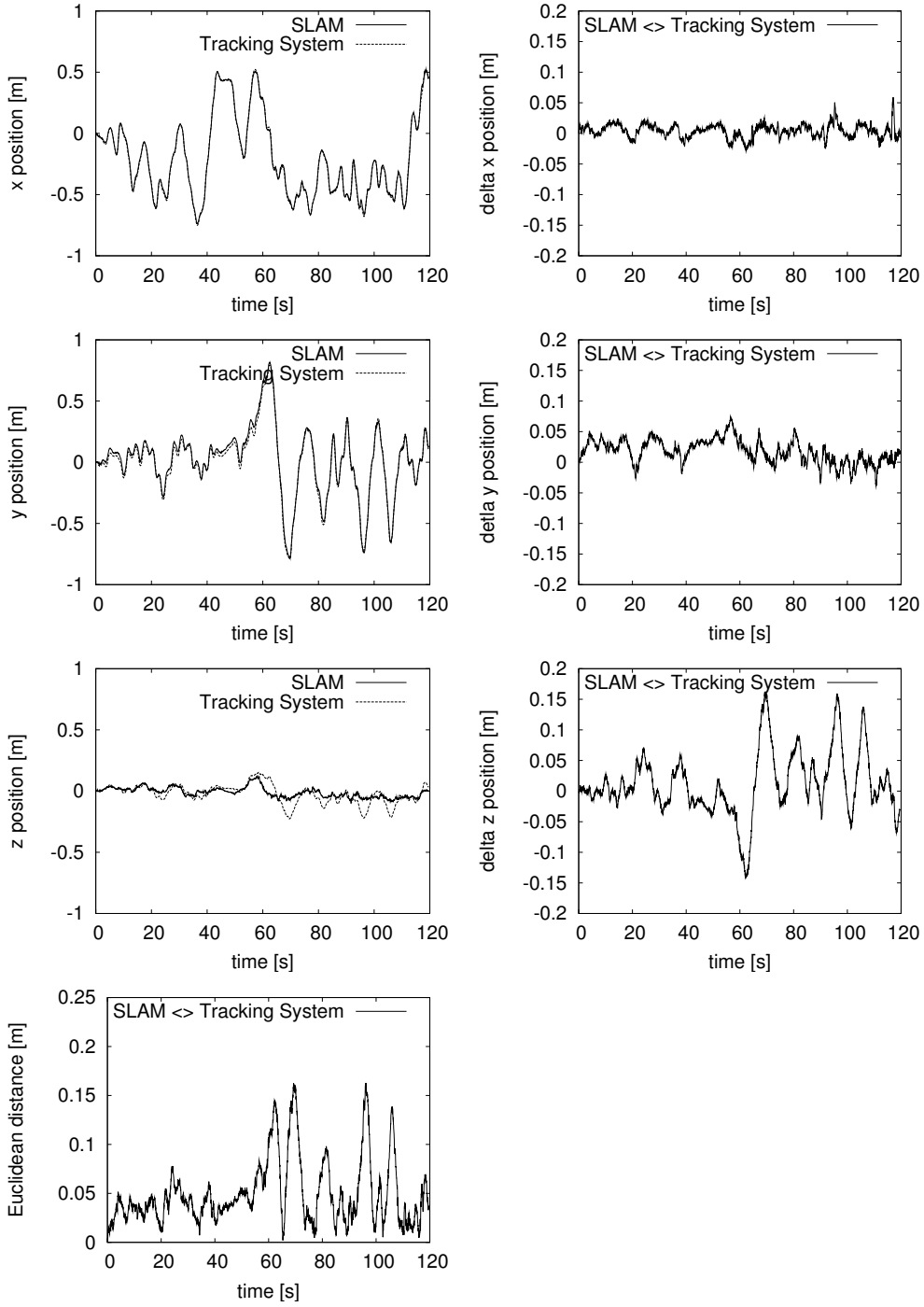


Figure 7.4: These plots show the position of the platform over the time. The left column shows the estimated attitude and the reference attitude from the tracking system in a combined plot. The difference of the individual coordinates is visible in the right column. The last row shows the Euclidean distance between the estimated position from the SLAM implementation and the reference position.

calculation exceed 100 meter within the two minutes the experiment was run. Thus, it is clearly shown that the approach first permits the position calculation over a long time.

7.1.3 Attitude

If comparing the plots for the estimated angles and the reference angles in figure 7.5, it is noticeable that the plots for the roll and pitch angle do not start at zero. This is caused by the initialization strategy of the system. The measurable projection of the gravity vector to the axes of the acceleration sensors is used to determine an initial attitude of the system. The yaw angle starts at zero because it is not observable by the projection of the gravity vector and was arbitrarily chosen to start at zero. Any other value might be used as well what leads in the end to a rotation of the estimated map.

The error characteristics of the individual axes do not differ much from each other. Although the maximum error is below two degrees, this has to be considered critical as the error propagation in the strapdown is most sensitive to an error in the attitude. To get a stable system even with this error in the attitude, the assumed noise for the acceleration sensors in the sampling step was increased. This procedure allows the estimation process to follow the real movement even with a wrong projected gravity vector. The drawback of this approach is the loss of accuracy in the positioning estimation. Another group suffered from this effect as well and proposed to use two different state evolution models, one when the system is exposed to small accelerations that does not consider the real measurements and one when the system exceeds a certain level of acceleration [BS08]. We could not observe a better result when trying this approach. Instead the estimation got worse because no information of the acceleration sensors is used if the measurements are under a certain threshold. Even a usage of the sensors assuming a very high noise level helps the estimation process to gain a better result.

7.2 Influence of Particle Number

As the errors of the inertial measurement unit are modeled with the distribution of the particles, enough particles have to be available in order to provide a good estimation. If no particle occupies a state close to the real one, the system is not able to interpret the measurements correctly.

This is clearly visible in figure 7.6. The base for this graph is the same captured data like used in the previous section. It was evaluated how a different particle number influences the result of the system. The first graph shows an abstraction of the result when doubling the particle number from 50 to 6400 between each evaluation. Altogether 5 runs of the algorithm with different seeds for the random number generators were made per fixed number of particles and the resulting scaling factor of the estimated trajectory was used to judge if it was a stable run or not. If this scaling factor exceeds specific bounds, this means that the tracking of the landmarks failed and the estimation of the position was completely driven by the inertial measurement unit. As the introduced errors are unbounded, the scaling will exceed specific limits in that case.

It is visible that no stable run was derived for 25, 50 or 100 particles. The cause is the lack of available hypotheses for the map as well as the position. As no particle

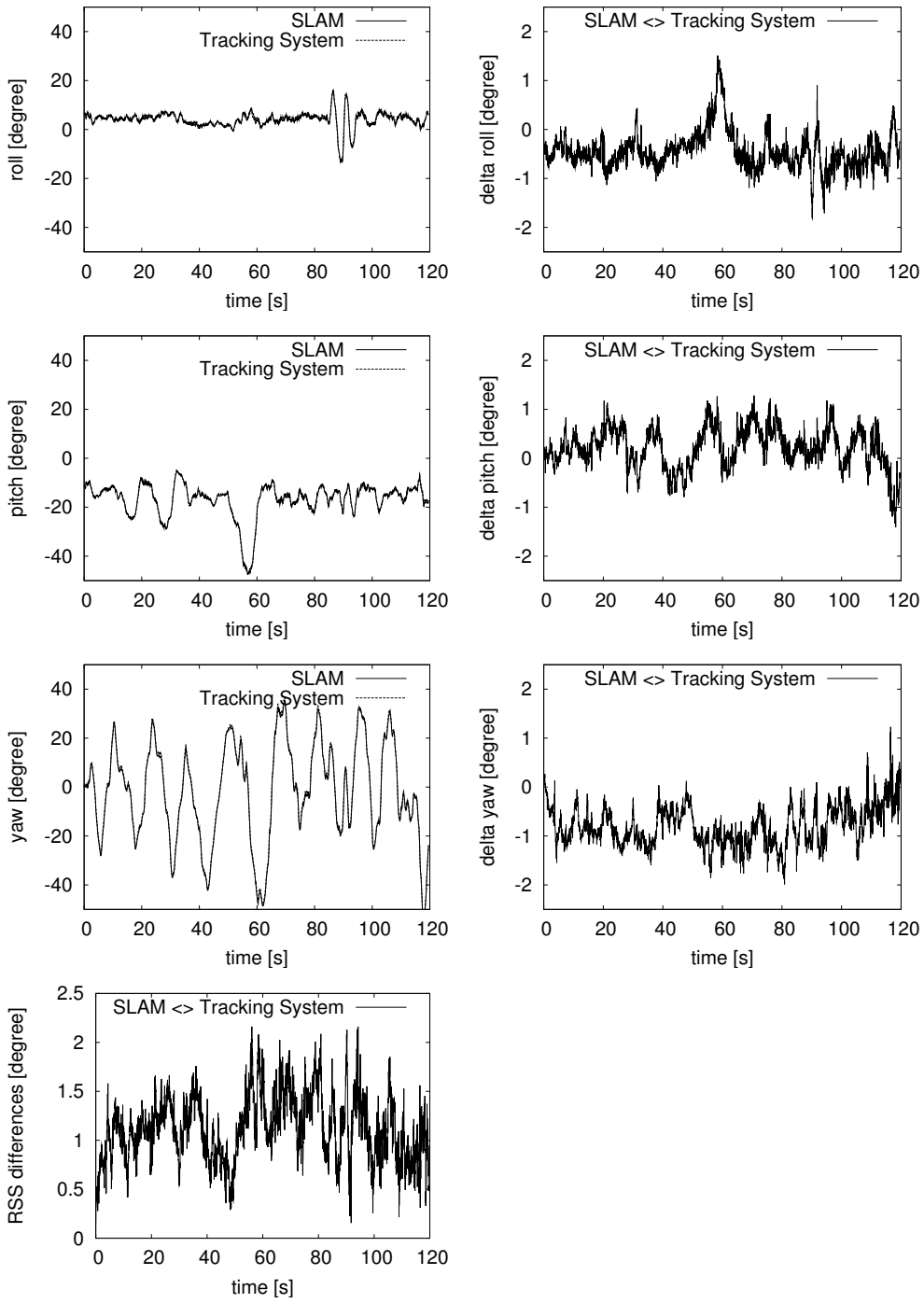


Figure 7.5: These plots show the estimated Euler angles of the SLAM algorithm and the reference angles calculated from the data of the tracking system. The left column shows the estimated attitude and the reference attitude in a combined plot. The difference between the individual values is visible in the right column. The very last row shows the square-root of the sum of the squared (root sum square, RSS) differences of the individual Euler angles.

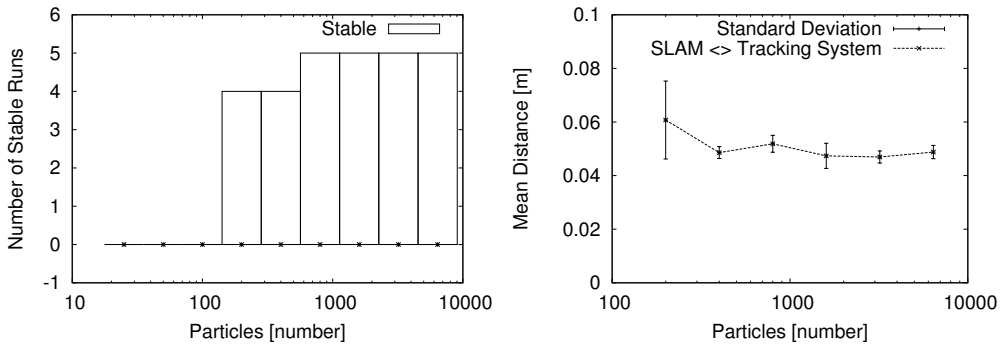


Figure 7.6: These plots show the influence of the particle number to the stability and to the error of the estimation. The left plot shows how many runs of the algorithm with different seeds for the random number generator were successful. The right plot shows the averaged mean distance between the trajectory of the tracking system and the estimated trajectory of the SLAM algorithm for all stable runs.

matches the measurements, the weighting step degenerates and does not reflect the true distribution. For 200 and 400 particles, four out of five runs were successful. This is a clue for the underlying probabilistic process used to solve the SLAM problem. As the state evolution model is driven by randomly drawing values from a certain distribution, the possibility always exists that no particle reflects the true state of the system if no sample was available that altered the state of one particle accordingly. For 800, 1600, 3200 and 6400 particles, all runs were successful. This means that always enough hypotheses were available to represent the true state. Nevertheless, this graph only reflects one particular experiment. If the number of landmarks increases and some are reobservated after a long time, a hypothesis has to be available that represents the true position of the landmark. As the resampling introduces a depletion of the map, the number of particles has to be adjusted accordingly. It is assumed that a larger map also needs a higher count of particles if no sub-mapping strategy is used.

The second plot shows the mean of the distances of the estimated position of the platform and the measurements from the tracking system, averaged for all runs including the standard deviation. Only the stable runs were considered, thus, the results for 25, 50 and particles are not visualized within this plot. It is noticeable that the standard deviation for 200 particles is high compared to the other number of particles. This reflects directly the border in the count of particles that are necessary to gain a stable and accurate system.

7.3 Influence of Feature Number

In order to get a stable system, the measurements from the camera are necessary as they are the only reference of the system that helps to bound the errors of the position and attitude estimation. The plots in figure 7.7 show the influence of the number of

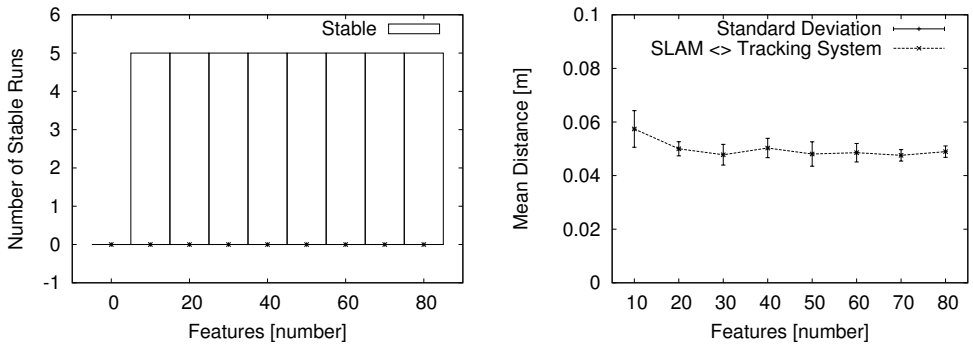


Figure 7.7: These plots show the influence of the number of tracked features to the stability and to the error of the estimation similar to figure 7.6.

features that are tracked within one frame. As the computing time is limited, the number of features that are matched and tracked has to be adjustable. To derive these plots, the same set of captured data was used like in the previous section. The number of particles was fixed to 1000 particles per run.

The first graph shows the number of stable runs for different number of features. Like in the previous section, a stable run was detected by calculating and interpreting the scaling factor between the trajectory of the tracking system and the estimated one. As expected, the system has zero stable runs if no feature is tracked. This means, that the position and attitude estimation only depends on the inertial measurement unit. The unbounded error leads to the instability of the system. Between 10 and 80 features, no difference is seen. All runs were judged as stable.

The second graph shows how the number of features influences the mean error of the distance between the tracking system and the estimated position. The values were derived by averaging the results of five different runs. The standard deviation is also plotted. Except for 10 features, no obvious difference is visible. This might be caused by the initialization strategy of the individual landmarks and their close distance to the camera. The closer the distance, the better the information of the landmarks is usable as the magnitude of the change of the angles is high. As we initialized the features with the aim to get a uniform distribution of their positions on the image plane, the measurements of as few as ten features for this experiment is enough to discriminate the position and attitude with high accuracy.

8 Conclusion

An approach to the problem of simultaneous localization and mapping was analyzed in this thesis. Starting from a general description of Bayesian filters, a factored solution based on particle filters and extended Kalman filters to solve the combined estimation problem was presented. As sensors for the platform, a camera and an inertial measurement unit was chosen. The according characteristics of these sensors were discussed and the mathematical models were derived in order to merge the measurements to get a stable system. The experimental results were discussed and proofed the accuracy of the system.

The complete processing chain for the abstraction of the information obtained from the camera was discussed. The system is able to identify natural landmarks within the images and track those in order to build a map and localize the platform. For the description of the projected landmarks, patches were used as feature descriptors. These patches enable the system with the usage of the normalized cross correlation to repeatedly identify a certain landmark among several frames. The strategies for the management, deletion and initialization of the landmarks were discussed. Furthermore, two parametrization for the landmark representation were shown and their employment within an extended Kalman filter derived.

The data from the inertial measurement unit was used for the propagation of the state of the platform. It was shown that the unbounded errors introduced by the inertial measurement unit could be successfully limited by merging the information obtained from the camera. This combination yields a stable system that is able to follow high dynamic movements.

The experimental results showed that the system is able to localize itself within a few centimeters. Although the estimated map was subject to scaling and rotation, the arrangement of the landmarks reflected the true world with high accuracy. In order to yield these results, the necessary software was implemented from scratch and enhanced by specific technologies to enable the real-time usage. To be able to incorporate the measurements from the tracking system, the according mathematical model was derived.

Concluding, the results show that the simultaneous localization and mapping problem can be successfully solved with an inertial measurement unit and a single camera. With the help of further research it is expected that this will lead to fully autonomous systems that can be used in several kind of applications.

8.1 Future Work

Although a running system was developed that proofed a high accuracy, not all problems could be addressed in the scope of this thesis. The system is constrained to small

environments and lacks stability under certain circumstances. This section is devoted to give an idea how the system could be improved and extended in the future.

- Proposal distribution
  An improvement of the proposal distribution will lead to a higher accuracy in the estimation process. In our approach, the sampling of the proposal distribution was based on the measurements and uncertainties introduced by the inertial measurement unit. As the images from the camera provide information about the accelerations and turn rates as well, the sampling of the proposal distribution should be based on the information of both sensors.
- Map representation
  The representation of all landmarks within a common coordinate system leads to problems. Every landmark that is newly initialized has a higher uncertainty in the position estimation. This is contributed to the increasing uncertainty in the localization what should be reflected in the distribution of the individual landmarks. As hypotheses get discarded during the resampling, the uncertainty of the position of the landmarks gets discarded as well. This leads especially to problems if regions are in the field of view again after a longer period of absence. The parameters of the landmark are overestimated and thus the measurements are misinterpreted and lead to an instability of the system. This problem can be focused with a strategy to divide the global map into sub-maps. Whenever all landmarks of a sub-map leave the field of view, they should be excluded from the map of the particles to avoid the change of the distribution due to resampling.
- IMU-camera calibration
  The calibration between the body frame and the camera respectively IMU frame was done manually for our experiments. In order to increase the accuracy, this process should be automated and rely on the measurements from the individual sensors.
- Initial distance estimation
  Due to inaccuracies in the attitude estimation, the acceleration sensors were modeled with a high noise level to avoid effects caused by a wrong projection of the gravity vector to the global axes. This also means that the estimation of the position of the platform mainly depends on the images from the camera. As the camera is not able to measure distances, the scaling of the derived map depends on specific priors of the system. This problem can be solved by adjusting the initial distance of the system by observing a known pattern. Of course, this problem becomes obsolete if the measurements from the inertial measurement unit can be incorporated with less uncertainty.
- Patch transformation
  Although the patches were corrected for rotation before matching, other transformations were not considered. To improve the matching result the patches can be projected to the image plane according to the momentary state estimation of the platform. This involves further computational complexity and the trade-off between gain in accuracy and computing time has to be considered.
- Outlier rejection
  The strategy used in our approach to prevent outliers only considers the metric

obtained from the normalized cross correlation in order to judge the quality of the matching. A consideration of the expected position on the image plane and the agreement of an individual position of a feature to the whole set of features can be exploited to reject outliers and improve the estimation process.

- Synchronization
  The synchronization strategy used does not account for the offset between the arrival of data from the inertial measurement unit and the camera. To gain a higher accuracy, the position estimation could be interpolated to better match the true state of the platform.

Bibliography

[BB03] D.F. Bizup and D.E. Brown. The over-extended kalman filter - don't use it! In *Information Fusion, 2003. Proceedings of the Sixth International Conference of*, volume 1, pages 40 – 46, 2003.

[BGK06] Kostas E. Bekris, Max Glick, and Lydia E. Kavraki. Evaluation of algorithms for bearing-only slam. In *in Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 1937–1943. IEEE, 2006.

[BS08] G. Bleser and D. Strickery. Using the marginalised particle filter for real-time visual-inertial sensor fusion. In *Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on*, pages 3 –12, sept. 2008.

[BTGL06] Herbert Bay, Tuytelaars Tinne, Van Gool, and L. Surf: Speeded up robust features. In *9th European Conference on Computer Vision*, Graz Austria, May 2006.

[CD09] Margarita Chli and Andrew J. Davison. Active matching for visual tracking. *Robot. Auton. Syst.*, 57(12):1173–1187, 2009.

[CDM08] J. Civera, A.J. Davison, and J. Montiel. Inverse depth parametrization for monocular slam. *Robotics, IEEE Transactions on*, 24(5):932 –945, oct. 2008.

[Cro84] Franklin C. Crow. Summed-area tables for texture mapping. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212, New York, NY, USA, 1984. ACM.

[DC05] R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 64–69, 2005.

[DRMS07] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):1052–1067, 2007.

[HS88] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[JU01] S.J. Julier and J.K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238 – 4243 vol.4, 2001.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[MCD06] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular slam. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[Mor77] Hans Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, August 1977.

[MTKW02] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598. AAAI, 2002.

[MW04] J. Mallon and P.F. Whelan. Precise radial un-distortion of images. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 18 – 21 Vol.1, aug. 2004.

[SC86] R.C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

[SEG05] Robert Sim, Pantelis Elinas, and Matt Griffin. Vision-based slam using the rao-blackwellised particle filter. In *In IJCAI Workshop on Reasoning with Uncertainty in Robotics*, 2005.

[SGN05] Thomas Schön, Fredrik Gustafsson, and Per-Johan Nordlund. Marginalized particle filters for mixed linear nonlinear state-space models. *IEEE Trans. on Signal Processing*, 53:2279–2289, 2005.

[SGSB08] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard. Visual slam for flying vehicles. *Robotics, IEEE Transactions on*, 24(5):1088 –1093, oct. 2008.

[SLL01] Stephen Se, David Lowe, and Jim Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, pages 2051–2058, 2001.

[SLP07] N. Sunderhauf, S. Lange, and P. Protzel. Using the unscented kalman filter in mono-slam with inverse depth parametrization for autonomous airship control. In *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, pages 1 –6, sept. 2007.

[SSC90] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*, volume 8, pages 167–193. 1990.

[ST94] Jianbo Shi and Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[Thr02] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*. Morgan Kaufmann, 2002.

[Tit05] David H. Titterton. *Strapdown Inertial Navigation Technology (IEE Radar Series)*. Institution of Engineering and Technology, 2nd revised edition edition, 6 2005.

[TSKG09] David Törnqvist, Thomas B. Schön, Rickard Karlsson, and Fredrik Gustafsson. Particle filter slam with high dimensional vehicle model. *Journal of Intelligent and Robotic Systems*, 55(4):249–266, 2009.

[WCH92] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(10):965–980, 1992.

[Wen07] Jan Wendel. *Integrierte Navigationssysteme*. Oldenbourg Wissensch.Vlg, 3 2007.

[Woo07] Oliver J. Woodman. An introduction to inertial navigation. Technical Report UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, August 2007.

List of Figures

A Detailed Equations

A.1 Optimized Normalized Cross Correlation

The normalized cross correlation of an image Patch $\mathbf{P}(x,y)$ of size n with an image $\mathbf{I}(x,y)$ at position $(\delta x, \delta y)$ in the image is defined as:

$$NCC(\delta x, \delta y) = \frac{1}{n} \sum_{(x,y)\in\mathbf{P}} \frac{(\mathbf{P}(x,y) - \bar{\mathbf{P}})(\mathbf{I}(x+\delta x, y+\delta y) - \bar{\mathbf{I}})}{\sigma_P \sigma_I} \tag{A.1}$$

where

$$\begin{aligned} \bar{\mathbf{P}} &= \tfrac{1}{n} \sum_{(x,y)\in P} \mathbf{P}(x,y) & \sigma_{\mathbf{P}} &= \sqrt{\tfrac{1}{n} \sum_{(x,y)\in P} (\mathbf{P}(x,y) - \bar{\mathbf{P}})^2} \\ \bar{\mathbf{I}} &= \tfrac{1}{n} \sum_{(x,y)\in P} \mathbf{I}(x+\delta x, y+\delta y) & \sigma_{\mathbf{I}} &= \sqrt{\tfrac{1}{n} \sum_{(x,y)\in P} (\mathbf{I}(x+\delta x, y+\delta y) - \bar{\mathbf{I}})^2} \end{aligned} \tag{A.2}$$

To derive the optimized version, we will start with the numerator. As the numerator is the only part dependent on the sum, we may write:

$$\begin{aligned} & \sum_{(x,y)\in\mathbf{P}} (\mathbf{P}(x,y) - \bar{\mathbf{P}})(\mathbf{I}(x+\delta x, y+\delta y) - \bar{\mathbf{I}}) \\ = & \sum_{(x,y)\in\mathbf{P}} (\mathbf{P}(x,y)\mathbf{I}(x+\delta x, y+\delta y) - \mathbf{P}(x,y)\bar{\mathbf{I}} - \bar{\mathbf{P}}\mathbf{I}(x+\delta x, y+\delta y) + \bar{\mathbf{P}}\bar{\mathbf{I}}) \end{aligned} \tag{A.3}$$

with the help of the definition of the sum and squared sum of the image areas

$$\begin{aligned} \mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) &= \sum_{(x,y)\in\mathbf{P}} \mathbf{I}(x+\delta x, y+\delta y) \\ \mathrm{Sum}^2_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) &= \sum_{(x,y)\in\mathbf{P}} \mathbf{I}^2(x+\delta x, y+\delta y) \end{aligned} \tag{A.4}$$

we may write

$$\begin{aligned} \sum_{(x,y)\in\mathbf{P}} \mathbf{P}(x,y)\bar{\mathbf{I}} &= n\bar{\mathbf{P}}\bar{\mathbf{I}} \\ &= n\bar{\mathbf{P}}\tfrac{1}{n}\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) \\ &= \bar{\mathbf{P}}\,\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) \\ \sum_{(x,y)\in\mathbf{P}} \bar{\mathbf{P}}\mathbf{I}(x+\delta x, y+\delta y) &= \bar{\mathbf{P}}\,\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) \\ \sum_{(x,y)\in\mathbf{P}} \bar{\mathbf{P}}\bar{\mathbf{I}} &= \bar{\mathbf{P}}\,\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) \end{aligned} \tag{A.5}$$

Substituted into equation A.3 gives

$$\begin{aligned} & \sum_{(x,y)\in\mathbf{P}} (\mathbf{P}(x,y)-\bar{\mathbf{P}})(\mathbf{I}(x+\delta x, y+\delta y)-\bar{\mathbf{I}}) \\ = & \sum_{(x,y)\in\mathbf{P}} (\mathbf{P}(x,y)\mathbf{I}(x+\delta x, y+\delta y)) - \bar{\mathbf{P}}\,\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}) \end{aligned} \tag{A.6}$$

For the denominator, we start with the variance of the image under the area of the patch and transform it:

$$\begin{aligned} \sigma_{\mathbf{I}}^2 &= \tfrac{1}{n} \sum_{(x,y)\in P} (\mathbf{I}(x+\delta x, y+\delta y)-\bar{\mathbf{I}})^2 \\ &= \tfrac{1}{n} \sum_{(x,y)\in P} (\mathbf{I}^2(x+\delta x, y+\delta y) - 2\bar{\mathbf{I}}\mathbf{I}(x+\delta x, y+\delta y) + \bar{\mathbf{I}}^2) \end{aligned} \tag{A.7}$$

As the equations

$$\begin{aligned} \sum_{(x,y)\in P} (\bar{\mathbf{I}}\mathbf{I}(x+\delta x, y+\delta y)) &= \tfrac{1}{n}(\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}))^2 \\ \sum_{(x,y)\in P} (\bar{\mathbf{I}}^2) &= \tfrac{1}{n}(\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}))^2 \end{aligned} \tag{A.8}$$

are true, the variance can be expressed as

$$\begin{aligned} \sigma_{\mathbf{I}}^2 &= \tfrac{1}{n}\,\mathrm{Sum}_{\mathbf{I}}^2(\delta \mathrm{x}, \delta \mathrm{y}) - \tfrac{1}{n^2}(\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}))^2 \\ &= \tfrac{1}{n^2}(n\,\mathrm{Sum}_{\mathbf{I}}^2(\delta \mathrm{x}, \delta \mathrm{y}) - (\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}))^2) \end{aligned} \tag{A.9}$$

Thus, the denominator can be written as

$$\sigma_{\mathbf{P}}\sigma_{\mathbf{I}} = \tfrac{1}{n}\sigma_{\mathbf{P}}\sqrt{(n\,\mathrm{Sum}_{\mathbf{I}}^2(\delta \mathrm{x}, \delta \mathrm{y}) - (\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}))^2)} \tag{A.10}$$

Substituting equations A.10 and A.6 into the first equation 4.9 gives the optimized version presented in section 4.4.1:

$$NCC(\delta x, \delta y) = \frac{\sum_{(x,y)\in\mathbf{P}} (\mathbf{P}(x,y)\mathbf{I}(x+\delta x, y+\delta y)) - \bar{\mathbf{P}}\,\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y})}{\sigma_{\mathbf{P}}\sqrt{(n\,\mathrm{Sum}_{\mathbf{I}}^2(\delta \mathrm{x}, \delta \mathrm{y}) - (\mathrm{Sum}_{\mathbf{I}}(\delta \mathrm{x}, \delta \mathrm{y}))^2)}} \tag{A.11}$$

A.2 Using the Translation Information of the Tracking System

To derive the equation for using the translation information of the tracking system, we start with the dependencies of a point $\mathbf{p}^{(B)}$ in the body frame that is described in the tracking system and in the SLAM algorithm:

$$\mathbf{C}_{UD}^T(\mathbf{C}_{ZL}^T(\mathbf{C}_{LB}^T\mathbf{p}^{(B)} + \mathbf{t}_b^{(L)}) + \mathbf{t}_l^{(Z)}) = \mathbf{C}_{ZT}^T(\mathbf{C}_{UD}^T(\mathbf{C}_{TB}^T\mathbf{p}^{(B)} + \mathbf{t}_b^{(T)})) + \mathbf{t}_l^{(Z)} \tag{A.12}$$

where $\mathbf{C}_{UD}^T$ is the base transformation between the coordinate systems of the visual SLAM algorithm and the tracking system, $\mathbf{C}_{ZL}^T$ and $\mathbf{t}_l^{(Z)}$ are the rotation respectively

translation between the zero and the lab frame, $\mathbf{C}_{LB}^{T}$ and $\mathbf{t}_{b}^{(L)}$ are the rotation and translation between the lab and the body frame, $\mathbf{C}_{ZT}^{T}$ and $\mathbf{t}_{t}^{(Z)}$ are the measured rotation and translation of the tracking system between the zero frame and the tracking frame and $\mathbf{C}_{TB}^{T}$ as well as $\mathbf{t}_{b}^{(T)}$ describe the misalignment between the tracked coordinate system and the one used in the visual SLAM algorithm.

As we want to track the origin of the body frame, the point $\mathbf{p}^{(B)}$ is set to zero.

$$\mathbf{C}_{UD}^{T}(\mathbf{C}_{ZL}^{T}\mathbf{t}_{b}^{(L)}+\mathbf{t}_{l}^{(Z)})=\mathbf{C}_{ZT}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}+\mathbf{t}_{t}^{(Z)} \tag{A.13}$$

The translation between the zero and the lab frame is unknown and calculated from the data of the first measurement:

$$\mathbf{t}_{l}^{(Z)}=\mathbf{C}_{UD}(\mathbf{C}_{ZT,Init}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}+\mathbf{t}_{T,Init}^{(Z)})-\mathbf{C}_{ZL,Init}^{T}\mathbf{t}_{B,Init}^{(L)} \tag{A.14}$$

As the lab frame and the body frame are identical when the system is started, this simplifies to

$$\mathbf{t}_{l}^{(Z)}=\mathbf{C}_{UD}(\mathbf{C}_{ZT,Init}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}+\mathbf{t}_{T,Init}^{(Z)}) \tag{A.15}$$

The next step is to express the translation between the lab and the body frame starting from equation A.13:

$$\mathbf{t}_{b}^{(L)}=\mathbf{C}_{ZL}(\mathbf{C}_{UD}(\mathbf{C}_{ZT}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}+\mathbf{t}_{t}^{(Z)})-\mathbf{t}_{l}^{(Z)}) \tag{A.16}$$

Substituting equation A.15

$$\begin{aligned}
\mathbf{t}_{b}^{(L)} &= \mathbf{C}_{ZL}(\mathbf{C}_{UD}(\mathbf{C}_{ZT}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}+\mathbf{t}_{t}^{(Z)})-\mathbf{C}_{UD}(\mathbf{C}_{ZT,Init}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}+\mathbf{t}_{T,Init}^{(Z)})) \\
\mathbf{t}_{b}^{(L)} &= \mathbf{C}_{ZL}\mathbf{C}_{UD}(\mathbf{C}_{ZT}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}+\mathbf{t}_{t}^{(Z)}-\mathbf{C}_{ZT,Init}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}-\mathbf{t}_{T,Init}^{(Z)}) \\
\mathbf{t}_{b}^{(L)} &= \mathbf{C}_{ZL}\mathbf{C}_{UD}(\mathbf{C}_{ZT}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}-\mathbf{C}_{ZT,Init}^{T}\mathbf{C}_{UD}^{T}\mathbf{t}_{b}^{(T)}+\mathbf{t}_{t}^{(Z)}-\mathbf{t}_{T,Init}^{(Z)})
\end{aligned} \tag{A.17}$$

results in the equation 6.6 that was used to incorporate the translation information of the tracking system.