FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Mobile Robotik

EKF-SLAM mit optisch erkannten Markierungen zur Bestimmung einer Fahrzeugposition

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Markus Macsek, BSc
Matrikelnummer 1125676

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner
Mitwirkung: Univ.Ass. Dipl.-Ing. Dr.techn. Markus Bader

Wien, 28. September 2016

Markus Macsek	Wolfgang Kastner

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Mobile Robotics

EKF-SLAM using Visual Markers for Vehicle Pose Estimation

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computer Engineering

by

Markus Macsek, BSc
Registration Number 1125676

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner
Assistance: Univ.Ass. Dipl.-Ing. Dr.techn. Markus Bader

Vienna, 28th September, 2016

Markus Macsek	Wolfgang Kastner

Erklärung zur Verfassung der Arbeit

Markus Macsek, BSc
Weidengasse 7, 2203 Putzing am See

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 28. September 2016

Markus Macsek

Danksagung

An erster Stelle möchte ich mich bei meinen Eltern Iris und Franz Macsek bedanken, welche mir den Weg zu meiner Ausbildung ebneten. Sie haben es mir nicht nur ermöglicht, mich voll auf mein Studium zu fokussieren, sondern sie haben mich auch die ganze Zeit bis jetzt unterstützt. Neben meinen Eltern, wäre mein Erfolg im Studium nicht ohne meine Studienkollegen möglich gewesen. Sie waren für mich ein inspirierendes und motivierendes Umfeld, seit wir uns im ersten Semester kennen gelernt haben. Obwohl die Technische Informatik ein kleiner Studienzweig im Informatikstudium der Technischen Universität Wien ist, sind es so viele, dass ich sie nicht alle einzeln nennen kann. Weiters möchte ich Lukas Pfeifhofer und meinem mitwirkenden Betreuer Markus Bader für das Bereitstellen einer optischen Markenerkennung sowie deren Visualisierung danken. Aber nicht nur für das möchte ich mich bei Markus Bader bedanken, sondern auch für die viele Zeit für fruchtbare Diskussionen und Erklärungen, die er sich für mich genommen hat. Abschließend möchte ich noch meinen Betreuer Wolfgang Kastner hervorheben. Dieser hat diese Arbeit erst möglich gemacht und ermöglicht mir somit mein Studium abzuschließen.

Acknowledgements

First of all, I want to appreciate my parents Iris and Franz Macsek paving the way for my education. They enabled me not only to focus on my study, but they also support me all the time until now. Beside them, my success in study would not be possible without my student colleagues. They have always been an inspiring and motivating environment to me, since we met in the first semester. Although Computer Engineering is a small branch of the Informatics study at the Vienna University of Technology, they are so many I cannot name them all explicitly. Furthermore, I want to honor Lukas Pfeifhofer and my assistance advisor Markus Bader for providing me a visual marker detection and marker visualizations. But not only for this I want to thank Markus Bader, but also for spending all the time for fruitful discussions and explanations. Finally, I want to acknowledge my advisor Wolfgang Kastner for enabling me to write this thesis and conclude my study.

Kurzfassung

Das Bestimmen der Position und der Orientierung, d.h. die Pose eines Fahrzeuges in einer unbekannten Umgebung ist auch mit vorhandenen Methoden eine Herausforderung. Für diese Aufgabe muss zuerst eine Karte der Umgebung aufgenommen werden. Das *Simultaneous Localization and Mapping* (SLAM) Problem behandelt beide Aufgaben: das Bestimmen der Pose des Fahrzeugs sowie das gleichzeitige Erstellen einer Karte. Die Theorie von SLAM ist gut erforscht und es existieren bereits verschiedene Ansätze um das Problem zu lösen. Ein bekannter Ansatz ist EKF-SLAM, welcher die Erweiterung von Kalman-Filtern (EKF) nutzt. Ein weiteres Problem im Zusammenhang mit SLAM ist die Daten-Assoziierung von Sensor-Messungen mit der Karte. Zu diesem Zweck wird oft der *Nearest Neighbor Standard Filter* (NNSF) Ansatz benutzt, welcher nur wahrscheinliche Assoziierungen betrachtet und von diesen die Beste auswählt. Die in dieser Arbeit behandelten Karten konzentrieren sich auf charakteristische Merkmale in der Umgebung. Im konkreten Fall werden visuelle Markierungen sowohl zur Lokalisierung als auch zur Kartographierung verwendet. Diese ähneln QR-Codes von Konsumgütern. Das Ziel der Arbeit ist bekannte Algorithmen für EKF-SLAM zu implementieren. Dabei sollen einerseits die IDs in den Markierungen zur Daten-Assoziierung dienen. Andererseits sollen Markierungen mittels NNSF in der Karte zugeordnet werden, deren IDs nicht erkannt wurden. Dadurch erhält man einen hybriden Ansatz für die Daten-Assoziierung, welcher das SLAM-Ergebnis verbessern soll. Der wissenschaftliche Beitrag dieser Arbeit ist ein Rauschmodell, welches den Messfehler der genutzten Markierungs-Erkennung abbildet. Aus diesem Grund wurden Messungen von visuellen Markierungen aufgenommen und statistisch ausgewertet, um die Varianz der Messungen mittels Funktionen zu approximieren. Schlussendlich wurde dieses Rauschmodell samt der behandelten Algorithmen in einem Modul für das *Robot Operating System* (ROS) umgesetzt. Dabei wurde gezeigt, dass die Implementation in der Simulation einsetzbar ist.

Abstract

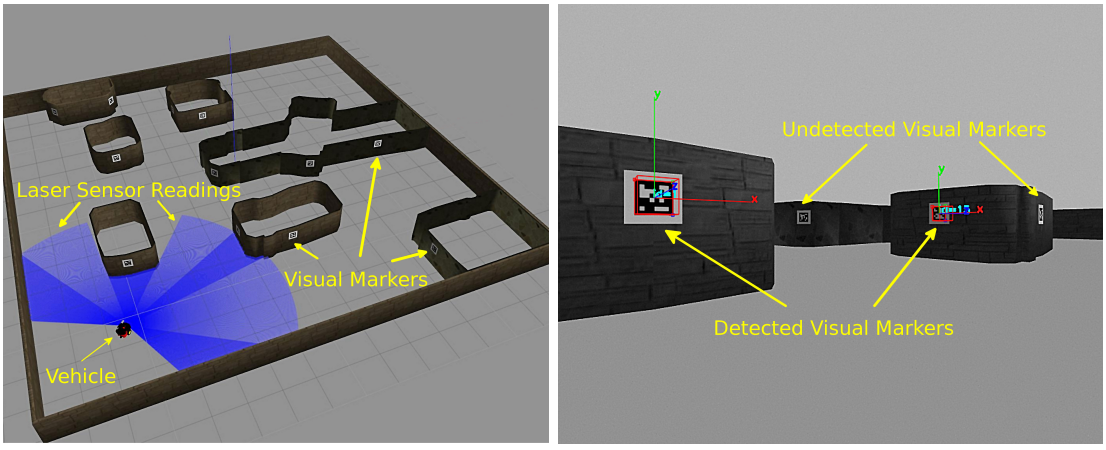
Estimating the position and orientation, i.e. the pose of a vehicle in an unknown environment is even with known methods a challenge. For this reason, a map of the environment first needs to be created. The Simultaneous Localization and Mapping (SLAM) problem covers both challenges: determining the vehicle's pose as well as creating a map. The theory of SLAM is well studied and several approaches solving the problem already exist. A common approach is EKF-SLAM, which makes use of the Extended Kalman Filter (EKF). Another problem arising in the context of SLAM is the data association of sensor measurements with the map. For this purpose, the Nearest Neighbor Standard Filter (NNSF) approach is used. This is a well-known approach, which only considers likely associations and accepts the most probable among them. This work focuses on feature-based maps, which means that outstanding features in the environment - in our case visual markers similar to QR codes on consumer products - are used for localization and mapping. The aim of this thesis is to implement known algorithms for feature-based EKF-SLAM. On the one hand, this covers exploiting the benefits of the visual markers' IDs for known correspondences between detected visual markers and map features. On the other hand, the EKF-SLAM result is improved by observations without IDs using NNSF resolving the unknown correspondences. This yields a hybrid approach for the data association problem. The scientific contribution is a measurement noise model for the visual marker detection used. For this purpose, measurements of visual markers are recorded and statistically evaluated in order to derive approximation functions for the measurement variance. Finally, the obtained measurement noise model and the discussed algorithms are put into practice by providing a package for the Robot Operating System (ROS). The resulting implementation has been shown to be applicable to a simulated environment.

Contents

CHAPTER 1

Introduction

One of the main issues in mobile robotics is determining the actual pose of a robot, where the pose comprises the robot's position and orientation. For this purpose, a map is needed. Unfortunately, such a map does not always exist and needs to be created first. Methods covering this kind of issue are known under the name *Simultaneous Localization and Mapping* (SLAM) or *Concurrent Mapping and Localization* (CML). Figure 1.1 depicts such a setting used within the thesis. The visual markers are mapped by the system and used to estimate the vehicle's pose. The laser sensor readings shown in figure 1.1a can be used for navigating the robot without crashing into obstacles.



(a) Environment containing visual markers with vehicle and laser sensor readings

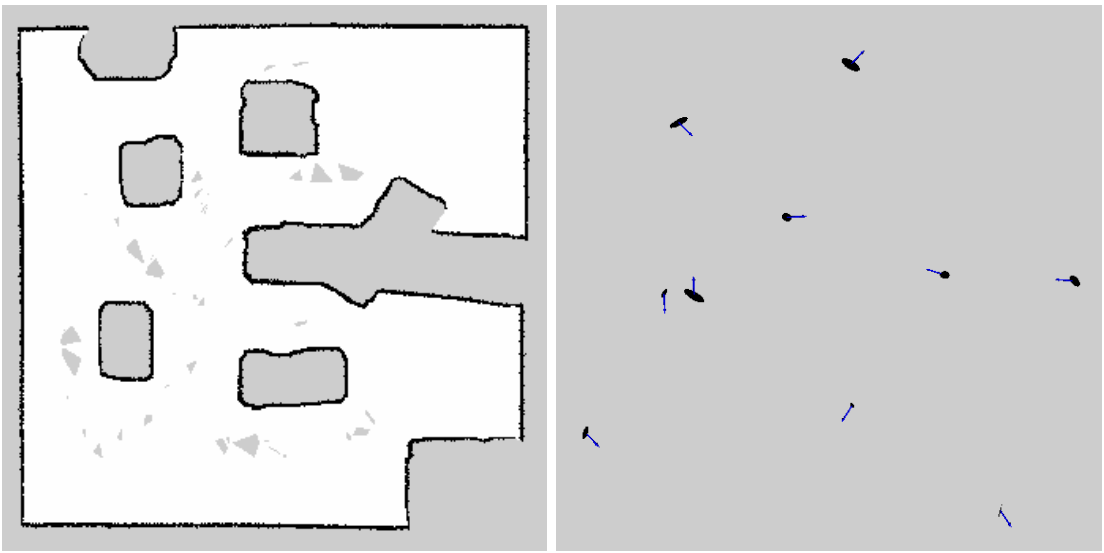
(b) Vehicle's camera perspective with detected and undetected visual markers used for SLAM

Figure 1.1: Used simulated environment with vehicle

1.1 Problem Statement

The challenging aspect of the SLAM task is that the robot needs both a map in order to localize itself and its pose to be able to create a map of its environment. Hence, for a long time SLAM was supposed to be a *chicken-or-egg* problem. Nevertheless, in the past decade a variety of techniques solving the SLAM problem has evolved. A newer approach is to handle the inherent uncertainty with probabilistic concepts. Important techniques in this field are EKF-SLAM, FastSLAM and Graph-based SLAM, which are all well documented in [1]. This thesis will cover SLAM based on *Extended Kalman Filters* (EKF-SLAM).

Creating a map is based on the question as to what is its actual purpose, whether it is just for pose estimation or also a reference for further tasks such as *path planning*? In the latter case, an extensive map describing all obstacles is required in order to incorporate these obstacles into the task of path planning. Possible representations of the environment in this field are *occupancy grid maps* or simple *grid maps*. Figure 1.2a depicts a recorded grid map of the environment shown in figure 1.1. The main drawbacks of grid maps are their high memory and processing consumptions. Fortunately, only establishing the robot's pose is required. This allows reducing the environment to features serving as reference points for localization. Figure 1.2b illustrates such a *feature-based map* of the detected markers shown in figure 1.1 using the contained visual markers for features. Beside the markers' poses (depicted as arrows in the figure) also their uncertainty (indicated by ellipses around) are stored in feature-based maps. The consequence is a considerably smaller map enabling a better pose estimation. The mentioned uncertainty comes within the estimation process and is a main concept of EKF-SLAM.



(a) 2D occupancy grid (b) 2D feature-based

Figure 1.2: Map representations

The thesis will make usage of visual markers for features as depicted in figure 1.1b. A main advantage of visual markers is that they not only represent a pose but can also encode additional information such as an ID with them ensuring known correspondences between observations and the created map.

1.2 Aim of the Work

The aim of this thesis is to implement known algorithms for feature based EKF-SLAM using visual markers with IDs and to enhance the setup to deal with unrecognized IDs as well. This means exploiting the benefits of IDs but also using observations without IDs for the further improvement of the SLAM results. Beside that, a measurement noise model for visual marker detection is created. For this purpose, several runs of marker detection are recorded. The results of these runs are then compared with the real poses of the detected visual markers in order to derive a proper noise model. In doing so, the impact of an accurate measurement noise model on the actual SLAM task is evaluated.

In order to make the implementation available and reusable to a broad community, we decided to implement a SLAM framework, which already supports EKF-SLAM, using the *Robot Operating System* (ROS) [2]. ROS follows a modular design. A module in ROS is named package and can consist of several subunits such as libraries or nodes. A ROS node in turn describes an executable. Via XML messages, several ROS nodes communicate with each other by publishing and subscribing to topics.

Thus, a ROS package is provided containing a ROS node representing the framework for SLAM. Built upon the framework, the node includes an already working implementation of feature based EKF-SLAM in 2D. For simulation and testing, there will be another ROS node in this package modeling measurement noise. Finally, the package also contains a node implementing a server for saving and providing recorded marker maps.

1.3 Related Work

At the beginning of this work, existing ROS packages were inspected in [3] to ensure a contribution to the ROS community. The most important of them for this thesis are listed below:

- `robot_localization`
  Author: Tom Moore
  License: BSD
  This ROS package provides a framework for ROS nodes performing state estimation for non-linear transitions in 3D space. In particular, two nodes are implemented yet: `ekf_localization_node` and `ukf_localization_node`. The first one makes usage of *Extended Kalman Filters* (EKFs) for non-linear state transitions. The latter exploits *Unscented Kalman Filters* (UKFs) for this purpose. The estimated state covers not only the position and the orientation but also the corresponding

velocities and accelerations yielding a 15-dimensional state vector. Both of these nodes are able to combine an arbitrary number of different sensor inputs for the correction of the state estimation. In [4], they compared the results of their EKF approach with varying sensor types, e.g. *odometry*, *Inertial Measurement Unit* (IMU), *Global Positioning System* (GPS). In contrast to this thesis, they focus only on the vehicle's state but are not interested in creating a map of the explored environment while estimating the vehicle's state.

- `robot_pose_ekf`
  Author: Wim Meeussen
  License: BSD
  Similar to the previous package, this ROS package takes advantage of EKFs for the state estimation of a vehicle in 3D space. But in contrast, only a 6-dimensional state consisting of the position and the orientation of the vehicle is supported. For the estimation itself, measurements from multiple sources covering the wanted state in parts can be utilized. Once again, this package targets only the localization problem in mobile robotics but does not treat the mapping problem.

- `hector_localization`
  Author: Johannes Meyer
  License: BSD
  This ROS package represents another solution for the localization-only problem in 3D space. The package contains multiple sub-packages, which are organized in a stack. In turn, the estimated state consists of 6 *Degrees of Freedom* (DOFs). Measurements of different sensor sources are compound by using an EKF. Despite the fact that IMUs are primary used for this purpose, there are also other kinds of sensors e.g. GPS, magnetometer, barometric pressure supported depending on the application.

- `hector_slam`
  Author: Stefan Kohlbrecher, Johannes Meyer
  License: BSD
  In this ROS Package both problems - localization as well as mapping - are treated the first time. In contrast to this thesis, not feature-based maps but occupancy grid maps of the environment are created using *Light Detection and Ranging* (LiDAR) sensors. In this context, the combination of a 9-dimensional state vector consisting of position, orientation and velocities in 3D space and a 2D grid map is interesting. For this purpose, the estimated state in 3D is projected into the xy-plane. For the state estimation a 3D motion model of the used platform is fed with sensor readings of an IMU and then filtered by an EKF. For more details [5] can be used.

- `stereo_slam`
  Author: Pep Lluis Negre
  License: BSD
  Another solution for the SLAM problem is presented in this ROS package. The

developed package targets underwater robotics and has been successfully tested in submarine environments. Again, occupancy grid maps are created instead of feature-based maps. But compared to the previous package the so derived maps are in 3D. For this purpose, a single stereo-camera is used, which enables a 3D reception of the environment.

- `vslam`
  Author: Kurt Konolige, Patrick Mihelich, Helen Oleynikova
  License: Public Domain,LGPL,LGPL/GPL
  This ROS package targets again the SLAM problem using visual camera images. In particular, images from a stereo-camera are utilized for creating 3D point clouds of indoor environments, in which the vehicle is localized. Unfortunately, the package is only little documented and is still in experimental usage for research.

- `rtabmap`
  Author: Mathieu Labbé
  License: BSD
  The *Real-Time Appearance-Based Mapping* (RTAB-Map) ROS Package solves the SLAM problem while satisfying real-time constraints. In doing so, an online global loop closure detection is used to identify already known places. In order to meet the real-time requirements the created map is divided into a *Working Memory* (WM) and *Long Time Memory* (LTM) [6], [7]. The former is used for the detection of loop closures because of its limited size. The map itself is organized by a graph, where each node expresses a location in the map enriched with visualization information. Thus, the provided solution is an instance of Graph-based SLAM. Depending on the used sensors the created map covers either 6 DOFs (e.g. stereo-camera) or 3 DOFs (e.g. LiDAR).

- `gmapping`
  Author: Brian Gerkey
  License: CreativeCommons-by-nc-sa-2.0
  This ROS package is actually a wrapper for OpenSLAM's implementation of a *Rao-Blackwellized particle filter* for the SLAM problem. In particular, such a Rao-Blackwellized particle filter implementation is an instance of FastSLAM. It uses particles for the vehicle's believed pose. Moreover, each particle comes up with its own map by incorporating laser sensor readings. Once again, the so created maps are 2D occupancy grid maps. Further information can be found in [8] and [9].

- `map_server`
  Author: Brian Gerkey, Tony Pratkanis
  License: BSD
  Contrary to the other packages, this ROS package neither provides state estimation nor creates maps. Instead it is a composition of the two ROS nodes: `map_saver` and `map_server`. The former can be used to save already generated occupancy grid maps. The maps are stored in an image file depicting the environment, which

is enhanced by an additional meta-data file. The second ROS node is able to read these files from the disk and to make the recorded grid maps available again. The so allocated maps can be used later on for e.g. localization-only or path-planning tasks. Thus, it served as a model for the feature-based map server coming along with this thesis.

Although EKFs have already been applied to localization-only tasks, they are rarely used in the context of SLAM in ROS. Furthermore, the majority of the discussed packages covering SLAM creates occupancy grid maps and none of them feature-based maps as supposed for this thesis. Finally, visual markers have neither been exploited for localization-only nor for SLAM tasks in any of the listed ROS packages.

Thus, Markus Bader provided for the aim of this thesis a visual marker detection for AR-ToolKit visual markers [10] based on the ARToolKitPlus library [11]. This visual marker detection was later on improved by Lukas Pfeifhofer using the ArUco library [12], [13]. Although both approaches can be found in the ROS package `tuw_marker_detection`, the results derived in this thesis are only based on the latter.

1.4 Methodological Approach

The overall methodological approach for this thesis can be broken down into the following phases:

1. *Literature study on the SLAM topic and ROS environment*
   At the beginning, EKF had to be revisited and extended to EKF-SLAM. For general purposes, especially [1] was used. Beside that, additional paper study of robotic conferences and journals was conducted. Also, existing ROS packages had to be studied and investigated so as to find out how ROS nodes need to be arranged. Then, our own ROS package was designed with the main goal to include an EKF-SLAM algorithm.

2. *Simulation environment and visualization*
   In order to rerun experiments and to reduce the resource overhead by using real robots, two different types of simulation environments were used during the work. In a first step, Stage [2] served as a 2D simulation environment for evaluating EKF-SLAM itself. Since Stage is not capable of simulating camera images as needed for visual marker detection, later on Gazebo [2] a more realistic 3D simulation environment was used. Furthermore, the ROS node for Stage did not support simple marker detection, but the Stage environment itself does. Thus, a marker detection with artificial measurement noise was added to the current ROS node of Stage. In the next step, one had to think about how the SLAM results are visualized. For this reason, RViz [2] was used enhanced by visualization plugins for measurements and detected visual markers.

3. *EKF-SLAM with known feature correspondences*
   In the first version, the EKF-SLAM supported only observations with IDs. Based on one ID, the observation could be assigned to the corresponding visual marker. In a first try, the estimated vehicle and visual markers' poses were iteratively corrected by the observations. As soon as this was working, the approach was extended to support correction using all observations in a single step.

4. *Extension to support unknown feature correspondences*
   In the first version, observations without IDs were remaining. Supposing they improve the estimated pose too, we wanted to use them. Thus, in the case of no observed ID the correspondence was achieved by assigning the observation to the most probable visual marker using a Nearest Neighbor Standard Filter (NNSF). Again two approaches were implemented. The first one simply assigns each observation without ID to the visual marker with the shortest Mahalanobis distance. This may result in local optima, e.g. two observations are assigned to the same visual marker. The second one finds a global optimum match between observations without IDs and visual markers based on their Mahalanobis distances. In both cases, only visual markers are considered, which do not correspond to an observation with ID yet.

5. *Measurement noise model*
   As soon as this was working, Gazebo was used further on as a simulation environment supporting camera images for visual marker detection. Until then, an artificial measurement noise based on adjustable sigma values had been applied for simulation and testing. To achieve a more realistic measurement noise for practical relevant testing, measurements of visual markers were recorded. Afterwards the recorded measurements were compared with the predicted measurements based on the real vehicle's pose and real visual markers' poses. From this comparison, we then deduced a more realistic measurement noise model.

6. *Results and application*
   Finally, a proper modular implementation was set up to provide a broader community with the framework. This is supposed to ensure a wide usage with long maintenance support and possible extensions. To this end, a simple marker map server was added so as to save and provide recorded marker maps for further use (e.g. visual marker based EKF localization).

1.5 Structure of the Work

The thesis itself is divided into 6 chapters, starting with this brief introduction into the topic of EKF-SLAM. In chapter 2, basic concepts of probability theory will be recapitulated. The provided theory is mainly based on [1], [14] and [15], which can be used for further details. This introduction into probability theory is needed to establish a common understanding for the remaining thesis. In chapter 3, the actual topic of the

thesis - EKF-SLAM using visual markers - will be introduced. In the chapter, algorithms solving the EKF-SLAM problem together will be elaborated. The presented information widely rests upon [1] and [16]. In chapter 4, the previously introduced algorithms will be enhanced with implementation-specific details regarding the provided ROS package related to the thesis. The latter will shortly be summed up at the end of the chapter. Beside further concepts from [1], especially optimizations developed in [17], [18] will be exploited for this purpose. Before the results of the provided implementation are discussed, a measurement noise model for visual markers used for map features will be elaborated in chapter 5. This noise model comprises the measurement error of the used visual marker detection. Finally, chapter 6 will conclude the thesis with a discussion of the drawbacks and challenges of the provided solution. In doing so, they will be compared with state of the art developments and an outlook on possible improvements will be given.

CHAPTER 2

Probabilistic Concepts in Robotics

"*Robotics is the science of perceiving and manipulating the physical world through computer-controlled devices.*" [1, p. 3] Consequently, a main issue in robotics is dealing with the interface between the real and virtual world. But this comes with an inherent uncertainty.

Let us think about the environment of a robot. In early days of robotics, the robot's space of action was enclosed, e.g. assembly lines. This reduced unpredictable process impact. Nowadays, the robotics' area of applications extends from closed manufactures towards to more open environments like private households, e.g. autonomous lawn mower or vacuum cleaner. This evolution makes it more and more harder to find appropriate models describing such an environment. But models are important for determining the actual state of a robot in its environment. [1]

Besides, the impact of the used sensors and actuators must be pointed out. No sensor or actuator is perfect. Instead one has to deal with so called *measurement noise* using sensors and *control noise* using actuators. These noises make robots in practice nondeterministic.

Thus, in the near past all these sources of uncertainty raised the question, why trying to hide the uncertainty and not dealing with it? That is the point where probabilistic robotics comes into play. Probabilistic robotics makes use of probabilistic concepts dealing with this uncertainty. As a side effect, probabilistic robotics turned out to be more robust toward weak models of environments as well as weak sensors and actuators than classical approaches based on logical consequences. [1], [15]

In the remaining chapter, the basics in probabilistic robotics are recapped in order to establish a common understanding for the theoretical concepts of EKF-SLAM. Thus, section 2.1 retains further used probabilistic principles. In section 2.2, Bayesian filters are derived from Bayes Networks and Hidden Markov Models, respectively. Built upon

Bayesian filters, Gaussian filters are introduced in section 2.3 with special care to Kalman Filters and their extension.

2.1 Probabilistic Theory

In this section, the basic probabilistic theory needed for EKF-SLAM is introduced. The covered theory and notation are mainly based on [14], which can be used for going further into detail.

A *random variable* x assigns each possible outcome of a random experiment within a given domain Ω a value. Furthermore, any subset $\mathcal{A}$ of the domain Ω and thus a possible outcome of the random experiment is called an *event*. The random variables considered in this thesis will be usually real-valued, thus $\Omega = \{\mathsf{x} \in \mathbb{R}\} = \{-\infty < \mathsf{x} < \infty\}$. A typical example of an event in this domain will be $\{\mathsf{x} \leq 0\}$.

Moreover let $\mathrm{P}\{\mathcal{A}\}$ denote the *probability* of event $\mathcal{A}$ whereas

- $0 \leq \mathrm{P}\{\mathcal{A}\} \leq 1$ with
- $\mathrm{P}\{\emptyset\} = 0$ (impossible event) and
- $\mathrm{P}\{\Omega\} = 1$ (certain event)

describe the major probability properties.

Although this section mainly focuses on continuous random scalars x in respect to continuous deterministic scalars x, all the introduced concepts can be extended to random vectors $\mathbf{x}$ and deterministic vectors $\boldsymbol{x}$ respectively, as well as to discrete and mixed random variables. Table 2.1 sums up the notation used throughout this thesis.

	deterministic	random
scalar	x	x
vector	$\boldsymbol{x}$	$\mathbf{x}$
matrix	$\boldsymbol{X}$	$\mathbf{X}$

Table 2.1: Notation [14]

Furthermore, the used vectors are supposed to be column vectors and the superscript $^\top$ denotes the *transposition* of a vector or matrix.

2.1.1 Cumulative Distribution Function (cdf)

The cdf $F_{\mathsf{x}}(x)$ of a real-valued random variable x denotes the probability of the event $\{\mathsf{x} \leq x\}$:

$$F_{\mathsf{x}}(x) = \mathrm{P}\{\mathsf{x} \leq x\} \tag{2.1}$$

Formally, it is defined as

$$F_{\mathsf{x}}(x) : \mathbb{R} \to [0, 1] \tag{2.2}$$

with the following properties:

- monotonically increasing

$$x_1 < x_2 \implies F_{\mathsf{x}}(x_1) \leq F_{\mathsf{x}}(x_2) \tag{2.3}$$

- lower limit equals zero

$$F_{\mathsf{x}}(-\infty) = \mathrm{P}\{\mathsf{x} \leq -\infty\} = \lim_{x \to -\infty} F_{\mathsf{x}}(x) = 0 \tag{2.4}$$

- upper limit equals one

$$F_{\mathsf{x}}(\infty) = \mathrm{P}\{\mathsf{x} \leq \infty\} = \lim_{x \to \infty} F_{\mathsf{x}}(x) = 1 \tag{2.5}$$

Thus, it can be shown that the probability of x lying inside an interval $(a, b]$ opened to the left is gained by

$$\mathrm{P}\{\mathsf{x} \in (a, b]\} = \mathrm{P}\{a < \mathsf{x} \leq b\} = F_{\mathsf{x}}(b) - F_{\mathsf{x}}(a) \tag{2.6}$$

and consequently

$$\mathrm{P}\{\mathsf{x} > x\} = 1 - F_{\mathsf{x}}(x) \tag{2.7}$$

holds since $F_{\mathsf{x}}(\infty) = 1$ by equation 2.35.

2.1.2 Probability Density Function (pdf)

The pdf $f_{\mathsf{x}}(x)$ of a real-valued random variable x describes now the relative limit of the event $\{\mathsf{x} = x\}$:

$$f_{\mathsf{x}}(x) = \lim_{\Delta x \to 0} \frac{\mathrm{P}\{x < \mathsf{x} \leq x + \Delta x\}}{\Delta x} = \frac{d}{dx} F_{\mathsf{x}}(x) \tag{2.8}$$

Notice: This inference implicitly uses equation 2.6.

The reason why the pdf describes just the relative limit is because in continuous space it is almost impossible that two events are really equal. Consider for example the shape of snowflakes, one can always find tiny differences between any two snowflakes [19], [20]. Thus, the probability of the event $\{\mathsf{x} = x\}$ is assumed to be 0:

$$\mathrm{P}\{\mathsf{x} = x\} = 0 \tag{2.9}$$

Formally, the pdf is defined as

$$f_{\mathsf{x}}(x) : \mathbb{R} \to [0, \infty) \tag{2.10}$$

with the following properties:

- nonnegative

$$f_{\mathsf{x}}(x) \geq 0 \tag{2.11}$$

- converges to zero

$$\lim_{x \to \pm\infty} f_{\mathsf{x}}(x) = 0 \tag{2.12}$$

- covered area equals one

$$\int_{-\infty}^{\infty} f_{\mathsf{x}}(x)\, dx = 1 \tag{2.13}$$

Immediately from equation 2.8 one can derive

$$\mathrm{P}\{\mathsf{x} \leq x\} = F_{\mathsf{x}}(x) = \int_{-\infty}^{x} f_{\mathsf{x}}(\xi)\ d\xi \tag{2.14}$$

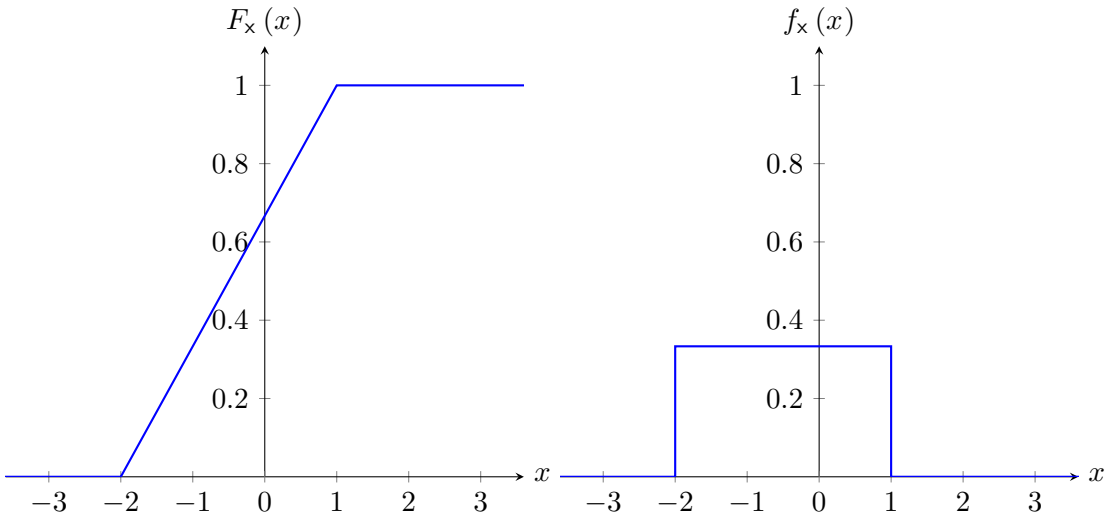
for the relation of the cdf and pdf in respect to the wanted probability.

Consequently, the probability of x lying inside an interval $(a, b]$ is determined by

$$\mathrm{P}\{\mathsf{x} \in (a, b]\} = \mathrm{P}\{a < \mathsf{x} \leq b\} = \int_{a}^{b} f_{\mathsf{x}}(x)\ dx \tag{2.15}$$

using equation 2.6 for cdfs related to intervals.

For a better understanding of cdf, pdf and their relation to each other figure 2.1 illustrates the cdf and pdf of a random variable x that is *uniformly distributed* in the interval $[-2, 1]$.



(a) Cumulative distribution function (b) Probabilistic density function

Figure 2.1: Uniform random variable $\mathsf{x} \sim \mathcal{U}(-2, 1)$

In general uniform distributions, denoted as $\mathsf{x} \sim \mathcal{U}(a, b)$, are specified by the cdf

$$F_{\mathsf{x}}(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & x > a \end{cases} \tag{2.16}$$

and the pdf

$$f_{\mathsf{x}}(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{else} \end{cases} \tag{2.17}$$

for the interval $[a, b]$.

2.1.3 Moments

Moments are characteristics of a random variable x. We distinguish between the k^{th} *moment*

$$m_{\mathsf{x}}^{(k)} \mathrel{\widehat{=}} \mathrm{E}\{\mathsf{x}^k\} = \int_{-\infty}^{\infty} x^k f_{\mathsf{x}}(x)\ dx \tag{2.18}$$

and the k^{th} *central moment*

$$m_{\mathsf{x}-\mu_{\mathsf{x}}}^{(k)} \mathrel{\widehat{=}} \mathrm{E}\{(\mathsf{x} - \mu_{\mathsf{x}})^k\} = \int_{-\infty}^{\infty} (x - \mu_{\mathsf{x}})^k f_{\mathsf{x}}(x)\ dx \tag{2.19}$$

with μ_{x} denoting the first moment, which is of special interest and therefore often simple called *mean* μ_{x}, *expected value* $\mathrm{E}\{\mathsf{x}\}$ or just *expectation*:

$$\mu_{\mathsf{x}} = \mathrm{E}\{\mathsf{x}\} \mathrel{\widehat{=}} \int_{-\infty}^{\infty} x f_{\mathsf{x}}(x)\ dx \tag{2.20}$$

An important property of the expectation is its linearity:

$$\mathrm{E}\{a\mathsf{x} + b\} = a\mathrm{E}\{\mathsf{x}\} + b = a\mu_{\mathsf{x}} + b \tag{2.21}$$

Proof.

$$\begin{aligned} \mathrm{E}\{a\mathsf{x} + b\} &= \int_{-\infty}^{\infty} (ax + b) f_{\mathsf{x}}(x)\ dx && \text{2.20} \\ &= \int_{-\infty}^{\infty} ax f_{\mathsf{x}}(x)\ dx + \int_{-\infty}^{\infty} b f_{\mathsf{x}}(x)\ dx \\ &= a\int_{-\infty}^{\infty} x f_{\mathsf{x}}(x)\ dx + b\int_{-\infty}^{\infty} f_{\mathsf{x}}(x)\ dx \\ &= a\mathrm{E}\{\mathsf{x}\} + b = a\mu_{\mathsf{x}} + b && \text{2.20, 2.13} \end{aligned}$$

□

Another important moment is the second central moment

$$\sigma_{\mathsf{x}}^2 = \mathrm{V}\{\mathsf{x}\} = \mathrm{E}\{(\mathsf{x} - \mu_{\mathsf{x}})^2\} \mathrel{\widehat{=}} \int_{-\infty}^{\infty} (x - \mu_{\mathsf{x}})^2 f_{\mathsf{x}}(x)\ dx \tag{2.22}$$

called *variance*. In contrast to the expectation, the variance is not linear:

$$\mathrm{V}\{a\mathsf{x} + b\} = a^2 \mathrm{V}\{\mathsf{x}\} = a^2 \sigma_{\mathsf{x}}^2 \tag{2.23}$$

Proof.

$$\begin{aligned}
\mathrm{V}\{a\mathsf{x} + b\} &= \mathrm{E}\{[(a\mathsf{x} + b) - (a\mu_{\mathsf{x}} + b)]^2\} && 2.22,\ 2.21 \\
&= \mathrm{E}\{[a(\mathsf{x} - \mu_{\mathsf{x}})]^2\} \\
&= a^2 \mathrm{E}\{(\mathsf{x} - \mu_{\mathsf{x}})^2\} && 2.21 \\
&= a^2 \mathrm{V}\{\mathsf{x}\} = a^2 \sigma_{\mathsf{x}}^2 && 2.22
\end{aligned}$$

□

2.1.4 Conditional cdf and pdf with one random variable

The *conditional* cdf $F_{\mathsf{x}}(x|\mathcal{A})$ of a real-valued random variable x given an event $\mathcal{A}$ denotes the probability of the event $\{(\mathsf{x} \leq x) \cap \mathcal{A}\}$ normalized by the probability of event $\mathcal{A}$:

$$F_{\mathsf{x}}(x|\mathcal{A}) = \mathrm{P}\{\mathsf{x} \leq x)|\mathcal{A}\} = \frac{\mathrm{P}\{(\mathsf{x} \leq x) \cap \mathcal{A}\}}{\mathrm{P}\{\mathcal{A}\}} \tag{2.24}$$

Similar to the ordinary cdf it is defined as

$$F_{\mathsf{x}}(x|\mathcal{A}) : \mathbb{R} \to [0, 1] \tag{2.25}$$

with the same properties, i.e. monotonically increasing, lower limit equals zero and upper limit equals one.

Consequently, the *conditional* pdf $f_{\mathsf{x}}(x|\mathcal{A})$ of a real-valued random variable x given an event $\mathcal{A}$ is defined as the first derivative of the cdf $F_{\mathsf{x}}(x|\mathcal{A})$:

$$f_{\mathsf{x}}(x|\mathcal{A}) = \frac{d}{dx} F_{\mathsf{x}}(x|\mathcal{A}) \tag{2.26}$$

Formally, the function

$$f_{\mathsf{x}}(x|\mathcal{A}) : \mathbb{R} \to [0, \infty) \tag{2.27}$$

satisfies all the properties of an ordinary pdf, i.e it is nonnegative, it converges to zero and its integral equals one.

In the context of conditional probability *Bayes theorem*

$$f_{\mathsf{x}}(x|\mathcal{A}) = \frac{\mathrm{P}\{\mathcal{A}|\mathsf{x} = x\}\, f_{\mathsf{x}}(x)}{\mathrm{P}\{\mathcal{A}\}} \tag{2.28}$$

is important. It relates the conditional pdf $f_{\mathsf{x}}(x|\mathcal{A})$ to the reverted conditional probability $\mathrm{P}\{\mathcal{A}|\mathsf{x} = x\}$.

2.1.5 Joint cdf and pdf

The *joint* cdf $F_{\mathsf{x},\mathsf{y}}(x,y)$ of two real-valued random variables x and y denotes the probability of the event $\{(\mathsf{x} \leq x) \cap (\mathsf{y} \leq y)\}$:

$$F_{\mathsf{x},\mathsf{y}}(x,y) = \mathrm{P}\{(\mathsf{x} \leq x) \cap (\mathsf{y} \leq y)\} \tag{2.29}$$

Similar to the cdf of one real-valued random variable, it is defined as

$$F_{\mathsf{x},\mathsf{y}}(x,y) : \mathbb{R}^2 \to [0,1] \tag{2.30}$$

with following properties:

- monotonically increasing in both arguments

$$x_1 < x_2,\ y_1 < y_2 \implies F_{\mathsf{x},\mathsf{y}}(x_1,y_1) \leq F_{\mathsf{x},\mathsf{y}}(x_2,y_2) \tag{2.31}$$

- limits

$$F_{\mathsf{x},\mathsf{y}}(x,-\infty) = F_{\mathsf{x},\mathsf{y}}(-\infty,y) = 0 \tag{2.32}$$

$$F_{\mathsf{x},\mathsf{y}}(x,\infty) = F_{\mathsf{x}}(x) \tag{2.33}$$

$$F_{\mathsf{x},\mathsf{y}}(\infty,y) = F_{\mathsf{y}}(y) \tag{2.34}$$

$$F_{\mathsf{x},\mathsf{y}}(\infty,\infty) = 1 \tag{2.35}$$

Analogous to the pdf of one real-valued random variable, the *joint* pdf $f_{\mathsf{x},\mathsf{y}}(x,y)$ of two real-valued random variables x and y is defined as the first derivative of the cdf $F_{\mathsf{x},\mathsf{y}}(x,y)$:

$$f_{\mathsf{x},\mathsf{y}}(x,y) = \frac{\partial}{\partial x}\frac{\partial}{\partial y}F_{\mathsf{x},\mathsf{y}}(x,y) \tag{2.36}$$

Formally, the joint pdf is now defined as

$$f_{\mathsf{x},\mathsf{y}}(x,y) : \mathbb{R}^2 \to [0,\infty) \tag{2.37}$$

with the following properties:

- nonnegative

$$f_{\mathsf{x},\mathsf{y}}(x,y) \geq 0 \tag{2.38}$$

- converges to zero

$$\lim_{x\to\pm\infty} f_{\mathsf{x},\mathsf{y}}(x,y) = 0 \tag{2.39}$$

$$\lim_{y\to\pm\infty} f_{\mathsf{x},\mathsf{y}}(x,y) = 0 \tag{2.40}$$

- covered area equals one

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{\mathsf{x},\mathsf{y}}(x,y)\ dx\,dy = 1 \tag{2.41}$$

Again, one can immediately derive from equation 2.36

$$\mathrm{P}\{(\mathsf{x}\leq x)\cap(\mathsf{y}\leq y)\} = F_{\mathsf{x},\mathsf{y}}(x,y) = \int_{-\infty}^{x}\int_{-\infty}^{y} f_{\mathsf{x},\mathsf{y}}(\xi,\zeta)\ d\xi\,d\zeta \tag{2.42}$$

the relation of the joint cdf and pdf in respect to the wanted probability.

Based on this and additionally motivated by the limit properties 2.33 and 2.34 of the joint cdf, one of the two variables can be integrated out yielding

$$f_{\mathsf{x}}(x) = \int_{-\infty}^{\infty} f_{\mathsf{x},\mathsf{y}}(x,y)\ dy \tag{2.43}$$

and

$$f_{\mathsf{y}}(y) = \int_{-\infty}^{\infty} f_{\mathsf{x},\mathsf{y}}(x,y)\ dx \tag{2.44}$$

respectively. This process is called *marginalization* and $f_{\mathsf{x}}(x)$ and $f_{\mathsf{y}}(y)$ are called *marginal* pdfs of the joint pdf $f_{\mathsf{x},\mathsf{y}}(x,y)$.

2.1.6 Conditional cdf and pdf with two random variables

Let us now reconsider the equations 2.24 and 2.26 for conditional cdf and pdf in the context of two random variables x and y. In particular, let event $\mathcal{A}$ denote $\{\mathsf{y}=y\}$. Since y is continuous the probability of the event $\{\mathsf{y}=y\}$ can be indicated by $\mathrm{P}\{\mathsf{y}=y\}=0$ according to equation 2.9. Consequently, this also applies to $\mathrm{P}\{(\mathsf{x}\leq x)\cap(\mathsf{y}=y)\}=0$ resulting in the indeterminate term $\frac{0}{0}$ for the definition of the conditional cdf.

Nevertheless, one can restate equation 2.24 for the conditional cdf of two random variables x and y by

$$F_{\mathsf{x}|\mathsf{y}}(x|y) = \mathrm{P}\{\mathsf{x}\leq x|\mathsf{y}=y\} = \frac{\int_{-\infty}^{x} f_{\mathsf{x},\mathsf{y}}(\xi,y)\ d\xi}{f_{\mathsf{y}}(y)} \tag{2.45}$$

with the joint pdf $f_{\mathsf{x},\mathsf{y}}(x,y)$ of x and y and the marginal pdf $f_{\mathsf{y}}(y)$ of y. The deterministic value y in the event $\{\mathsf{y}=y\}$ serves in this context as evidence or witness of the random variable y.

Proof. Referred to [14], we have:

$$\begin{aligned} F_{\mathsf{x}|\mathsf{y}}(x|y) = \mathrm{P}\{\mathsf{x} \leq x|\mathsf{y} = y\} &= \lim_{\Delta y \to 0} \mathrm{P}\{\mathsf{x} \leq x|y < \mathsf{y} \leq y + \Delta y\} \\ &= \lim_{\Delta y \to 0} \frac{\mathrm{P}\{(\mathsf{x} \leq x) \cap (y < \mathsf{y} \leq y + \Delta y)\}}{\mathrm{P}\{y < \mathsf{y} \leq y + \Delta y\}} && 2.24 \\ &= \lim_{\Delta y \to 0} \frac{\int_{-\infty}^{x} \int_{y}^{y+\Delta y} f_{\mathsf{x},\mathsf{y}}(\xi, \zeta)\ d\xi\, d\zeta}{\int_{y}^{y+\Delta y} f_{\mathsf{y}}(\zeta)\ d\zeta} && 2.15,\ 2.29,\ 2.42 \\ &= \lim_{\Delta y \to 0} \frac{\int_{-\infty}^{x} f_{\mathsf{x},\mathsf{y}}(\xi, y)\ d\xi \cdot \Delta y}{f_{\mathsf{y}}(y) \cdot \Delta y} \\ &= \frac{\int_{-\infty}^{x} f_{\mathsf{x},\mathsf{y}}(\xi, y)\ d\xi}{f_{\mathsf{y}}(y)} \end{aligned}$$

Notice: In the penultimate line the approximations $\int_{y}^{y+\Delta y} f_{\mathsf{x},\mathsf{y}}(\xi, \zeta)\ d\zeta \approx f_{\mathsf{x},\mathsf{y}}(\xi, y) \cdot \Delta y$ and $\int_{y}^{y+\Delta y} f_{\mathsf{y}}(\zeta)\ d\zeta \approx f_{\mathsf{y}}(y) \cdot \Delta y$ are used, which become exact in the limit $\Delta y \to 0$. □

Consequently, the *conditional* pdf

$$f_{\mathsf{x}|\mathsf{y}}(x|y) = \frac{f_{\mathsf{x},\mathsf{y}}(x, y)}{f_{\mathsf{y}}(y)} \tag{2.46}$$

follows from equation 2.26 immediately.

Furthermore, by simple exchange of x and y and re-inserting this for $f_{\mathsf{x},\mathsf{y}}(x, y)$ one can directly derive

$$f_{\mathsf{x}|\mathsf{y}}(x|y) = \frac{f_{\mathsf{y}|\mathsf{x}}(y|x)\, f_{\mathsf{x}}(x)}{f_{\mathsf{y}}(y)} \tag{2.47}$$

Bayes theorem as specified in equation 2.28. In literature, this deviation is also often called *Bayesian inference*.

Finally, one can place the derived conditional pdf into the marginalization relation 2.43

$$f_{\mathsf{x}}(x) = \int_{-\infty}^{\infty} f_{\mathsf{x}|\mathsf{y}}(x|y)\, f_{\mathsf{y}}(y)\ dy \tag{2.48}$$

describing the *total probability* theorem.

2.1.7 Statistical and conditional independence

Two random variables x and y are called *statistically independent* or just *independent* if their conditional pdfs satisfy following conditions:

$$f_{\mathsf{x}|\mathsf{y}}(x|y) = f_{\mathsf{x}}(x) \text{ and } f_{\mathsf{y}|\mathsf{x}}(y|x) = f_{\mathsf{y}}(y) \tag{2.49}$$

Notice: One relation implies the other but for the sake of completeness both are specified at this point.

Thus, by equation 2.46 the joint pdf of two statistically independent random variables x and y can be indicated by

$$f_{\mathsf{x},\mathsf{y}}(x,y) = f_{\mathsf{x}}(x)\, f_{\mathsf{y}}(y) \tag{2.50}$$

with their two marginal pdfs $f_{\mathsf{x}}(x)$ and $f_{\mathsf{y}}(y)$.

Another important property in this context is *conditional independence*. For this purpose, let us consider for a moment three random variables x, y and z. The random variables x and z are said to be conditional independent if they are independent given an evidence y of random variable y:

$$f_{\mathsf{x}|\mathsf{y},\mathsf{z}}(x|y,z) = f_{\mathsf{x}|\mathsf{y}}(x|y) \tag{2.51}$$

Notice: In general conditional independence $f_{\mathsf{x}|\mathsf{y},\mathsf{z}}(x|y,z) = f_{\mathsf{x}|\mathsf{y}}(x|y)$ does not automatically imply statistically independence $f_{\mathsf{x}|\mathsf{z}}(x|z) = f_{\mathsf{x}}(x)$

2.1.8 Perspectives and covariance matrix

In the remaining thesis, $F_{\mathsf{x}}(x)$ and $f_{\mathsf{x}}(x)$ are abbreviated with $F(x)$ and $f(x)$, respectively. Instead of writing $\mathrm{P}\{\mathsf{x} \leq x\}$, simple $\mathrm{P}\{\mathsf{x}\}$ is used. Furthermore, all the so far introduced formulas and deviations can be easily extended supporting random vectors $\mathbf{x}$ instead of random variables x.

Because of its relevance in the remaining thesis the *covariance matrix* $\mathbf{\Sigma}_{\mathbf{x}}$ of a random vector $\mathbf{x}$ is emphasized at this point extra. The covariance matrix of a random vector $\mathbf{x} = (\mathsf{x}_1 \ldots \mathsf{x}_n)^\top$ of dimension n is defined analogously to the variance of a random variable x in equation 2.22:

$$\mathbf{\Sigma}_{\mathbf{x}} = \mathrm{V}\{\mathbf{x}\} = \mathrm{E}\{(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})^\top\} \mathrel{\widehat{=}} \int_{\mathbb{R}^N} (\boldsymbol{x}-\boldsymbol{\mu}_{\mathbf{x}})(\boldsymbol{x}-\boldsymbol{\mu}_{\mathbf{x}})^\top f_{\mathbf{x}}(\boldsymbol{x})\; d\boldsymbol{x} \tag{2.52}$$

Hence, $\mathbf{\Sigma}_{\mathbf{x}}$ is a quadratic matrix of size $n \times n$

$$\mathbf{\Sigma}_{\mathbf{x}} = \begin{pmatrix} \sigma^2_{\mathsf{x}_1} & \Sigma_{\mathsf{x}_1\mathsf{x}_2} & \cdots & \Sigma_{\mathsf{x}_1\mathsf{x}_n} \\ \Sigma_{\mathsf{x}_2\mathsf{x}_1} & \sigma^2_{\mathsf{x}_2} & \cdots & \Sigma_{\mathsf{x}_2\mathsf{x}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{\mathsf{x}_n\mathsf{x}_1} & \Sigma_{\mathsf{x}_n\mathsf{x}_2} & \cdots & \sigma^2_{\mathsf{x}_n} \end{pmatrix} \tag{2.53}$$

composed of the covariances

$$\Sigma_{\mathsf{x}_i\mathsf{x}_j} = \mathrm{E}\{(\mathsf{x}_i - \mu_{\mathsf{x}_i})(\mathsf{x}_j - \mu_{\mathsf{x}_j})\},\ i \neq j \in [1,n] \tag{2.54}$$

and the variances

$$\sigma^2_{\mathsf{x}_k} = \mathrm{E}\{(\mathsf{x}_k - \mu_{\mathsf{x}_k})^2\},\ k \in [1,n] \tag{2.55}$$

in the diagonal. Obviously, $\mathbf{\Sigma_x}$ is symmetric

$$\mathbf{\Sigma_x} = \mathbf{\Sigma_x}^\top \tag{2.56}$$

and semi-positive definite

$$\boldsymbol{a}^\top \mathbf{\Sigma_x} \boldsymbol{a} \geq 0 \tag{2.57}$$

for any vector $\boldsymbol{a}$ of dimension n [14].

Proof. $\mathbf{\Sigma_x}$ is symmetric:

$$\mathbf{\Sigma_x}^\top = \mathrm{E}\{(\mathbf{x} - \boldsymbol{\mu_x})(\mathbf{x} - \boldsymbol{\mu_x})^\top\}^\top = \mathrm{E}\{(\mathbf{x} - \boldsymbol{\mu_x})(\mathbf{x} - \boldsymbol{\mu_x})^\top\} = \mathbf{\Sigma_x}$$

□

Proof. $\mathbf{\Sigma_x}$ semi-positive definite:

$$\begin{aligned}
\boldsymbol{a}^\top \mathbf{\Sigma_x} \boldsymbol{a} &= \boldsymbol{a}^\top \mathrm{E}\{(\mathbf{x} - \boldsymbol{\mu_x})(\mathbf{x} - \boldsymbol{\mu_x})^\top\}\boldsymbol{a} && 2.52 \\
&= \mathrm{E}\{\boldsymbol{a}^\top(\mathbf{x} - \boldsymbol{\mu_x})(\mathbf{x} - \boldsymbol{\mu_x})^\top \boldsymbol{a}\} && 2.21 \\
&= \mathrm{E}\{\boldsymbol{a}^\top(\mathbf{x} - \boldsymbol{\mu_x}) \cdot \boldsymbol{a}^\top(\mathbf{x} - \boldsymbol{\mu_x})\} \\
&= \mathrm{E}\{[\boldsymbol{a}^\top(\mathbf{x} - \boldsymbol{\mu_x})]^2\} \geq 0
\end{aligned}$$

Hint: $\boldsymbol{a}^\top(\mathbf{x} - \boldsymbol{\mu_x})$ describes a scalar. □

Finally, the non-linearity of the variance in equation 2.23 is re-stated at this point for a random vector $\mathbf{x}$ of dimension n:

$$\mathrm{V}\{\boldsymbol{A}\mathbf{x} + \boldsymbol{b}\} = \boldsymbol{A}\mathrm{V}\{\mathrm{x}\}\boldsymbol{A}^\top = \boldsymbol{A}\mathbf{\Sigma_x}\boldsymbol{A}^\top \tag{2.58}$$

In this equation, $\boldsymbol{A}$ denotes a matrix of size $n \times k$ and $\boldsymbol{b}$ denotes a vector of arbitrary dimension k.

Proof.

$$\begin{aligned}
\mathrm{V}\{\boldsymbol{A}\mathbf{x} + \boldsymbol{b}\} &= \mathrm{E}\{[(\boldsymbol{A}\mathbf{x} + \boldsymbol{b}) - (\boldsymbol{A}\boldsymbol{\mu_x} + \boldsymbol{b})][(\boldsymbol{A}\mathbf{x} + \boldsymbol{b}) - (\boldsymbol{A}\boldsymbol{\mu_x} + \boldsymbol{b})]^\top\} && 2.52,\ 2.21 \\
&= \mathrm{E}\{[\boldsymbol{A}(\mathbf{x} - \boldsymbol{\mu_x})][\boldsymbol{A}(\mathbf{x} - \boldsymbol{\mu_x})]^\top\} \\
&= \mathrm{E}\{\boldsymbol{A}(\mathbf{x} - \boldsymbol{\mu_x})(\mathbf{x} - \boldsymbol{\mu_x})^\top \boldsymbol{A}^\top\} \\
&= \boldsymbol{A}\mathrm{E}\{(\mathbf{x} - \boldsymbol{\mu_x})(\mathbf{x} - \boldsymbol{\mu_x})^\top\}\boldsymbol{A}^\top && 2.21 \\
&= \boldsymbol{A}\mathrm{V}\{\mathbf{x}\}\boldsymbol{A}^\top = \boldsymbol{A}\mathbf{\Sigma_x}\boldsymbol{A}^\top && 2.52
\end{aligned}$$

□

2.2 Bayesian Filter

This section describes the basic idea of Bayesian filters and their categories. The related theory is based on [1] and [15], which can be used as reference for further details. Bayesian filters describe the fundamental theory of EKF-SLAM and are essentially an extension of the Bayesian inference introduced previously in equation 2.47. In order to derive a Bayesian filter, for our purpose a *Hidden Markov Model* (HMM) is used. A HMM is a special case of a *Bayesian Network* (BN) describing a *Markov Chain* of unobservable random variables. In turn, a BN is a *Directed Acyclic Graph* (DAG) where the nodes represent random variables and the edges encode conditional independence. Detailed information about HMMs and BNs and how to deal with them can be found in [15].

For the aim of this thesis, it suffices to recap the concepts of statistical independence and conditional independence introduced in subsection 2.1.7 by means of figures 2.2 to 2.4. They show simple BNs depicting the essential relations between three random variables x, y and z. Shaded nodes denote given evidence of the correspondent random variable in the sense of conditional probability introduced in subsection 2.1.6. Let us now inspect these essential relations closer:

- *Causal chain:* $f_{\mathsf{x},\mathsf{y},\mathsf{z}}(x,y,z) = f_{\mathsf{z}|\mathsf{y}}(z|y)\, f_{\mathsf{y}|\mathsf{x}}(y|x)\, f_{\mathsf{x}}(x)$
  In a causal chain as illustrated in figure 2.2a, x and z are independent given an evidence y:
  $$f_{\mathsf{z}|\mathsf{x},\mathsf{y}}(z|x,y) = f_{\mathsf{z}|\mathsf{y}}(z|y) \tag{2.59}$$

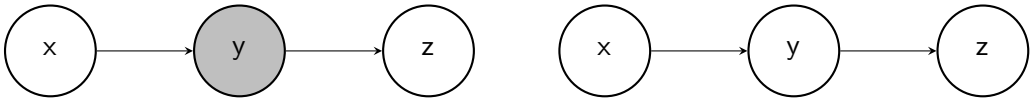
  Proof.

  $$f_{\mathsf{z}|\mathsf{x},\mathsf{y}}(z|x,y) = \frac{f_{\mathsf{x},\mathsf{y},\mathsf{z}}(x,y,z)}{f_{\mathsf{x},\mathsf{y}}(x,y)} = \frac{f_{\mathsf{z}|\mathsf{y}}(z|y)\, f_{\mathsf{y}|\mathsf{x}}(y|x)\, f_{\mathsf{x}}(x)}{f_{\mathsf{y}|\mathsf{x}}(y|x)\, f_{\mathsf{x}}(x)} = f_{\mathsf{z}|\mathsf{y}}(z|y)$$

  □

  Thus, by equation 2.51 we say x and z are conditionally independent.

  But without evidence of y as depicted in figure 2.2b, x and z are in general *not* statistically independent:
  $$f_{\mathsf{z}|\mathsf{x}}(z|x) \neq f_{\mathsf{z}}(z) \tag{2.60}$$
  One can easily show this by providing a counter example.



(a) with evidence (b) without evidence

Figure 2.2: Causal chain

- *Common cause*: $f_{\mathsf{x},\mathsf{y},\mathsf{z}}(x,y,z) = f_{\mathsf{z}|\mathsf{y}}(z|y)\, f_{\mathsf{x}|\mathsf{y}}(x|y)\, f_{\mathsf{y}}(y)$
  If x and z have a common cause y as illustrated in figure 2.3a then x and z are independent given an evidence y:

$$f_{\mathsf{z}|\mathsf{x},\mathsf{y}}(z|x,y) = f_{\mathsf{z}|\mathsf{y}}(z|y) \tag{2.61}$$

  Proof.

$$f_{\mathsf{z}|\mathsf{x},\mathsf{y}}(z|x,y) = \frac{f_{\mathsf{x},\mathsf{y},\mathsf{z}}(x,y,z)}{f_{\mathsf{x},\mathsf{y}}(x,y)} = \frac{f_{\mathsf{z}|\mathsf{y}}(z|y)\, f_{\mathsf{x}|\mathsf{y}}(x|y)\, f_{\mathsf{y}}(y)}{f_{\mathsf{x}|\mathsf{y}}(x|y)\, f_{\mathsf{y}}(y)} = f_{\mathsf{z}|\mathsf{y}}(z|y)$$

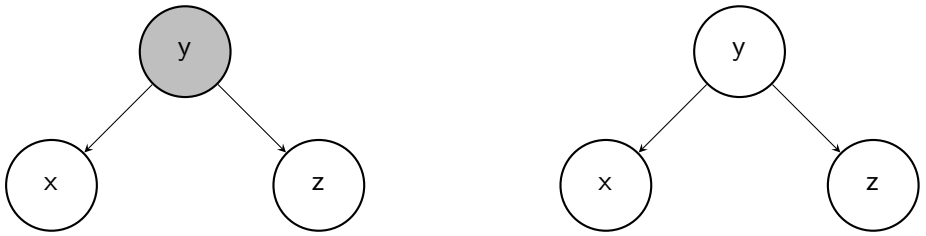
  □

  Thus, by equation 2.51 we say x and z are conditionally independent.

  Again, the witness y is crucial. In the case of no evidence of y as depicted in figure 2.3b, x and z are in general *not* statistically independent:

$$f_{\mathsf{z}|\mathsf{x}}(z|x) \neq f_{\mathsf{z}}(z) \tag{2.62}$$

  One can easily show this by providing a counter example.



(a) with evidence (b) without evidence

Figure 2.3: Common cause

- *Common effect:* $f_{\mathsf{x},\mathsf{y},\mathsf{z}}(x,y,z) = f_{\mathsf{y}|\mathsf{x},\mathsf{z}}(y|x,z)\, f_{\mathsf{x}}(x)\, f_{\mathsf{z}}(z)$
  The last case distinguishes from the others insofar as in the event of a common effect y of x and z as illustrated in figure 2.4a the random variables x and z are statistically independent as long as no evidence of y is given:

$$f_{\mathsf{z}|\mathsf{x}}(z|x) = f_{\mathsf{z}}(z) \tag{2.63}$$

  Proof.

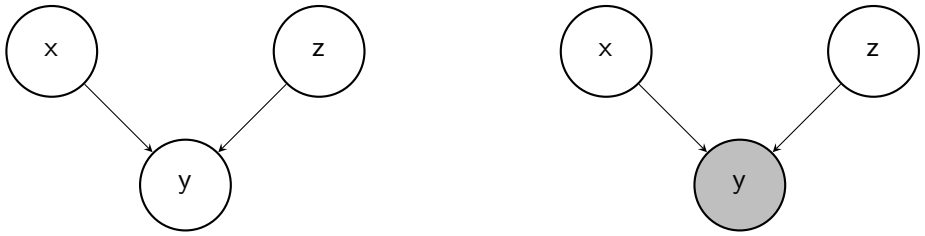
$$\begin{aligned} f_{\mathsf{z}|\mathsf{x}}(z|x) &= \frac{f_{\mathsf{x},\mathsf{z}}(x,z)}{f_{\mathsf{x}}(x)} = \frac{\int_{-\infty}^{\infty} f_{\mathsf{x},\mathsf{y},\mathsf{z}}(x,y,z)\ dy}{f_{\mathsf{x}}(x)} = \frac{\int_{-\infty}^{\infty} f_{\mathsf{y}|\mathsf{x},\mathsf{z}}(y|x,z)\, f_{\mathsf{x}}(x)\, f_{\mathsf{z}}(z)\ dy}{f_{\mathsf{x}}(x)} \\ &= f_{\mathsf{z}}(z) \int_{-\infty}^{\infty} f_{\mathsf{y}|\mathsf{x},\mathsf{z}}(y|x,z)\ dy = f_{\mathsf{z}}(z) \end{aligned} \tag{2.13}$$

  □

As soon as there is an evidence y one can provide a counter example showing that the random variables x and z are no longer independent.

$$f_{\mathsf{z}|\mathsf{x},\mathsf{y}}(z|x,y) \neq f_{\mathsf{z}|\mathsf{y}}(z|y) \tag{2.64}$$

Thus, we say x and z are *not* conditionally independent.



(a) without evidence (b) with evidence

Figure 2.4: Common effect

By means of this recapitulation we can now have a closer look at figure 2.5. The figure depicts a HMM for a moving vehicle with cyclic measurements. The vehicle's pose at time t is expressed by the real-valued random vector $\mathbf{x}_t$ of dimension n. The pose itself is unobserved and can only be influenced by the control commands for movement at time t denoted by the real-valued random vector $\mathbf{u}_t$ of dimension m and estimated using the measurements at time t described by the real-valued random vector $\mathbf{z}_t$ of dimension k.

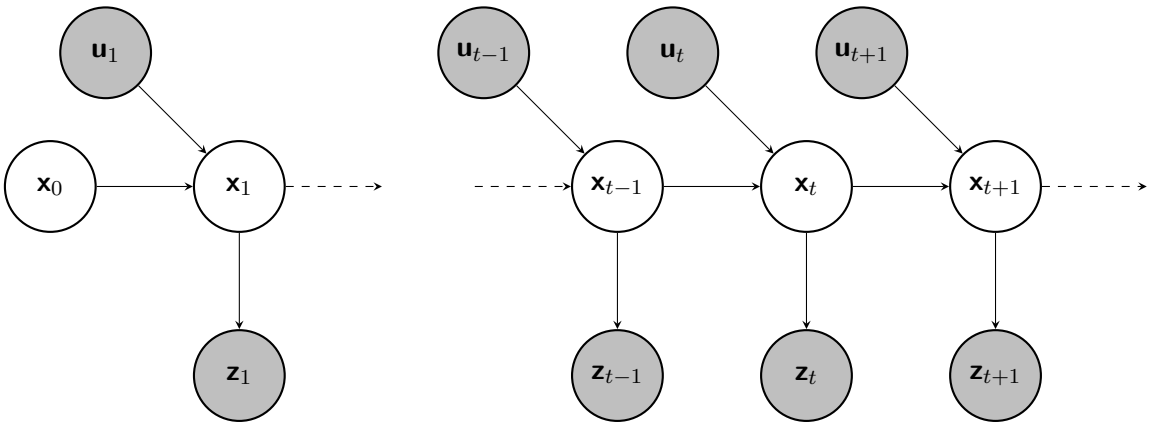


Figure 2.5: Hidden Markov Model underlying Bayesian filters [1]

Consequently, the actual target is estimating the current vehicle's pose $\mathbf{x}_t$ at its best by incorporating all control commands and measurements up to time t. Referred to [1], the estimated pose is then expressed by the believed state

$$\text{bel}(\mathbf{x}_t) = \mathbf{x}_t|\boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t} \tag{2.65}$$

with

$$\boldsymbol{u}_{1:t} = \boldsymbol{u}_1, \ldots, \boldsymbol{u}_t \tag{2.66}$$

denoting the control commands up to time t and

$$\boldsymbol{z}_{1:t} = \boldsymbol{z}_1, \ldots, \boldsymbol{z}_t \tag{2.67}$$

denoting the measurements up to time t.

Furthermore, it has been proven to be useful emphasizing additionally the state after applying the actual control commands but before incorporating the current measurements. Thus, considering only the control commands up to time t and the measurements up to time $t-1$ yields the predicted state

$$\overline{\text{bel}}\left(\mathbf{x}_t\right) = \mathbf{x}_t | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t-1} \tag{2.68}$$

which has not been corrected by the present measurement $\boldsymbol{z}_t$ yet [1].

In order to derive an expression for the believed state $\text{bel}\left(\mathbf{x}_t\right)$, we distinguish following fundamental conditional probabilities by exploiting conditional independence on the HMM depicted in figure 2.5:

- The *transition probability* $\text{P}\left\{\mathbf{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_t\right\}$ describes the probability of being in pose $\mathbf{x}_t$ given the old pose $\boldsymbol{x}_{t-1}$ and applying the current control command $\boldsymbol{u}_t$ to the vehicle.

- The *measurement probability* $\text{P}\left\{\mathbf{z}_t | \boldsymbol{x}_t\right\}$ denotes the probability of measurement $\mathbf{z}_t$ supposing the pose $\boldsymbol{x}_t$ for the vehicle at this time.

Beside the conditional independence encoded in the HMM, we further assume statistical independence between the current control commands $\boldsymbol{u}_t$ and the last pose $\boldsymbol{x}_{t-1}$ as they did in [1]:

$$f\left(\boldsymbol{u}_t | \boldsymbol{x}_{t-1}\right) = f\left(\boldsymbol{u}_t\right) \text{ and } f\left(\boldsymbol{x}_{t-1} | \boldsymbol{u}_t\right) = f\left(\boldsymbol{x}_{t-1}\right) \tag{2.69}$$

This assumption is quiet reasonable in static environments such as considered for EKF-SLAM and allows us to derive the following recursive equation for the *pdf* of the believed state $\text{bel}\left(\mathbf{x}_t\right)$ at time t:

$$f\left(\boldsymbol{x}_t | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}\right) = \eta\, f\left(\boldsymbol{z}_t | \boldsymbol{x}_t\right) \int_{\mathbb{R}^n} f\left(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_t\right)\; f\left(\boldsymbol{x}_{t-1} | \boldsymbol{u}_{1:t-1}, \boldsymbol{z}_{1:t-1}\right)\; d\boldsymbol{x}_{t-1} \tag{2.70}$$

Proof. By exploiting Bayesian inference in the context of conditional independence they derived in [1]

$$
\begin{aligned}
f\left(\boldsymbol{x}_t|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t}\right) &= \eta\ f\left(\boldsymbol{x}_t,\boldsymbol{z}_t|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right) && 2.46\\
&= \eta\ f\left(\boldsymbol{z}_t|\boldsymbol{x}_t,\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right)\ f\left(\boldsymbol{x}_t|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right) && 2.46\\
&= \eta\ f\left(\boldsymbol{z}_t|\boldsymbol{x}_t\right)\ f\left(\boldsymbol{x}_t|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right) && 2.51\\
&= \eta\ f\left(\boldsymbol{z}_t|\boldsymbol{x}_t\right)\int_{\mathbb{R}^n} f\left(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right)d\boldsymbol{x}_{t-1} && 2.48\\
&= \eta\ f\left(\boldsymbol{z}_t|\boldsymbol{x}_t\right)\int_{\mathbb{R}^n} f\left(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right)\ f\left(\boldsymbol{x}_{t-1}|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right)d\boldsymbol{x}_{t-1} && 2.46\\
&= \eta\ f\left(\boldsymbol{z}_t|\boldsymbol{x}_t\right)\int_{\mathbb{R}^n} f\left(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_t\right)\ f\left(\boldsymbol{x}_{t-1}|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right)d\boldsymbol{x}_{t-1} && 2.51\\
&= \eta\ f\left(\boldsymbol{z}_t|\boldsymbol{x}_t\right)\int_{\mathbb{R}^n} f\left(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_t\right)\ f\left(\boldsymbol{x}_{t-1}|\boldsymbol{u}_{1:t-1},\boldsymbol{z}_{1:t-1}\right)d\boldsymbol{x}_{t-1} && 2.69
\end{aligned}
$$

with

$$\eta = \frac{1}{f\left(\boldsymbol{z}_t|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right)}$$

denoting a normalization factor. □

Notice: In the context of Bayesian inference often a normalization factor η is used avoiding too complex equations and allowing to focus on the essentials. This factor does not represent a specific term or value and is thus often mixed up together with other normalization factors without explicit indication. In the future, an explicit specification of such a normalization factor η is therefore omitted.

In equation 2.70, $f\left(\boldsymbol{z}_t|\boldsymbol{x}_t\right)$ denotes the pdf of the mentioned measurement probability $\mathrm{P}\left\{\mathbf{z}_t|\boldsymbol{x}_t\right\}$ and $f\left(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_t\right)$ the pdf of the mentioned transition probability $\mathrm{P}\left\{\mathbf{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_t\right\}$. Furthermore, equation 2.70 of the Bayesian filter can be split into a prediction step

$$f_{\mathrm{pred}}\left(\boldsymbol{x}_t\right) = \int_{\mathbb{R}^n} f\left(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_t\right)\ f_{\mathrm{corr}}\left(\boldsymbol{x_{t-1}}\right)\ d\boldsymbol{x}_{t-1} \tag{2.71}$$

and a correction step

$$f_{\mathrm{corr}}\left(\boldsymbol{x}_t\right) = \eta\ f\left(\boldsymbol{z}_t|\boldsymbol{x}_t\right)\ f_{\mathrm{pred}}\left(\boldsymbol{x}_t\right) \tag{2.72}$$

distinguishing the two pdfs for the predicted state $\overline{\mathrm{bel}}\left(\mathbf{x}_t\right)$ (cp. line 3 of equation 2.70's proof) and the actual believed state $\mathrm{bel}\left(\mathbf{x}_t\right)$, respectively.

2.3 Gaussian Filter

The problem of Bayesian filters is that they are in general not implementable. The reason therefore arises from the infinity character of continuous values as indicated by

equation 2.70 for Bayesian filters. As in the previous section, the introduced concepts in this section and its subsections strongly refer to [1].

An exception in this context are *Gaussian distributions* or also named *normal distributions*, since two single moments suffice to describe the whole distribution. Furthermore, the properties of Gaussian distributions ensure that normally distributed random variables stay Gaussian when applying Bayesian filters to them. In general, we refer to this approach as Gaussian filters.

In particular, for a normally distributed random variable x the two mentioned moments are the mean μ_x and the variance σ_x^2 as introduced in equations 2.20 and 2.22, respectively. Consequently, a random variable x with Gaussian distribution is stated by $\mathsf{x} \sim \mathcal{N}(\mu_\mathsf{x}, \sigma_\mathsf{x}^2)$ with the pdf:

$$f_\mathsf{x}(x) = \frac{1}{\sqrt{2\pi\sigma_\mathsf{x}^2}}\, e^{-\frac{(x-\mu_\mathsf{x})^2}{2\sigma_\mathsf{x}^2}} \tag{2.73}$$

In the case of Gaussian distributions over vectors one says a random vector $\mathbf{x}$ is *n-variate normally distributed* with n denoting the dimension of $\mathbf{x}$. But for simplicity most times just *multivariate normally distributed* is used. The pdf of a multivariate Gaussian $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x})$ is given by

$$f_\mathbf{x}(\boldsymbol{x}) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_\mathbf{x}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_\mathbf{x})^\top \boldsymbol{\Sigma}_\mathbf{x}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_\mathbf{x})\right) \tag{2.74}$$

where $|\boldsymbol{\Sigma}_\mathbf{x}|$ denotes the *determinant* and $\boldsymbol{\Sigma}_\mathbf{x}^{-1}$ the *inverse* of the matrix $\boldsymbol{\Sigma}_\mathbf{x}$.

Figure 2.6 depicts the pdfs of Gaussian distributions over a scalar and a 2 dimensional vector. Both times they are zero-centered. In the scalar case the variance is 1 and in the vector case the covariance matrix is indicated by the *identity matrix* $\boldsymbol{I}$ of size 2×2. In general, the identity matrix

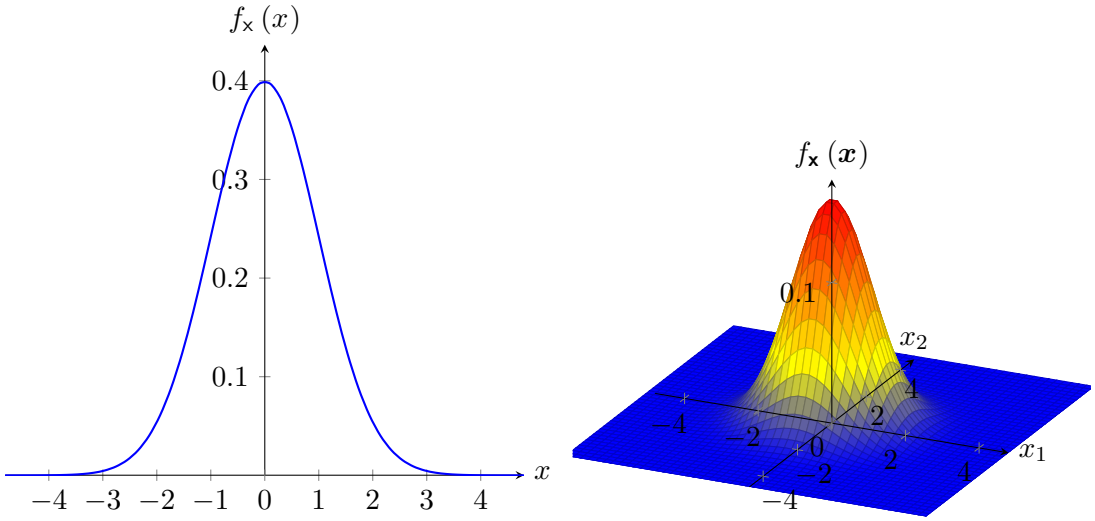
$$\boldsymbol{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{2.75}$$

denotes a sparse matrix of size $n \times n$. The reason why $\boldsymbol{I}$ is named identity matrix, is because of its main property

$$\boldsymbol{D} \cdot \boldsymbol{I} = \boldsymbol{D} \tag{2.76}$$

where $\boldsymbol{D}$ is an arbitrary matrix of size $k \times n$.

From figure 2.6, we can also deduce a common interpretation of the believed state $\mathrm{bel}(\mathbf{x}_t)$ in Gaussian filters. The most probable state - the mean $\boldsymbol{\mu}_{\mathbf{x}_t}$ - is supposed as the actual believed state and the covariance $\boldsymbol{\Sigma}_{\mathbf{x}_t}$ determines the uncertainty of this assumption. The latter can be indicated in general by hyper-ellipsoids (projection into the plane).



(a) $\mathsf{x} \sim \mathcal{N}(0, 1)$ (b) $\mathbf{x} = (x_1\ x_2)^\top \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Figure 2.6: Gaussian distributions

An important property of multivariate normal distributions used within the remaining thesis is linearity regarding transformations. Consider a multivariate Gaussian $\mathbf{x}$ and the linear transformation:

$$\mathbf{y} = \boldsymbol{A}\mathbf{x} + \boldsymbol{b} \tag{2.77}$$

Then the random vector $\mathbf{y}$ is also multivariate normally distributed according to

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}} + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Sigma}_{\mathbf{x}}\boldsymbol{A}^\top) \tag{2.78}$$

with $\boldsymbol{\mu}_{\mathbf{x}}$ and $\boldsymbol{\Sigma}_{\mathbf{x}}$ denoting the mean and covariance of $\mathbf{x}$.

Proof. By [14] the pdf of a random vector $\mathbf{y}$ as depicted in equation 2.77 can be stated by

$$f_{\mathbf{y}}(\boldsymbol{y}) = \frac{1}{|\boldsymbol{A}|} f_{\mathsf{x}}\left(\boldsymbol{A}^{-1}(\mathbf{y} - \boldsymbol{b})\right)$$

where $f_{\mathsf{x}}(x)$ denotes the pdf of the random vector $\mathbf{x}$.

From this, we can directly derive the pdf of $\mathbf{y}$:

$$\begin{aligned}
f_{\mathbf{y}}(y) &= \frac{1}{|\boldsymbol{A}|}\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_{\mathbf{x}}|}}\exp\left(-\frac{1}{2}\left[\boldsymbol{A}^{-1}(\mathbf{y}-\boldsymbol{b})-\boldsymbol{\mu}_{\mathbf{x}}\right]^{\top}\boldsymbol{\Sigma}_{\mathbf{x}}^{-1}\left[\boldsymbol{A}^{-1}(\mathbf{y}-\boldsymbol{b})-\boldsymbol{\mu}_{\mathbf{x}}\right]\right)\\
&= \frac{1}{\sqrt{|\boldsymbol{A}|^{2}|2\pi\boldsymbol{\Sigma}_{\mathbf{x}}|}}\exp\left(-\frac{1}{2}\left[\boldsymbol{A}^{-1}(\mathbf{y}-\boldsymbol{b}-\boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}})\right]^{\top}\boldsymbol{\Sigma}_{\mathbf{x}}^{-1}\left[\boldsymbol{A}^{-1}(\mathbf{y}-\boldsymbol{b}-\boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}})\right]\right)\\
&= \frac{1}{\sqrt{|\boldsymbol{A}||2\pi\boldsymbol{\Sigma}_{\mathbf{x}}||\boldsymbol{A}^{\top}|}}\exp\left(-\frac{1}{2}\left[\mathbf{y}-(\boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}}+\boldsymbol{b})\right]^{\top}\left(\boldsymbol{A}^{\top}\right)^{-1}\boldsymbol{\Sigma}_{\mathbf{x}}^{-1}\boldsymbol{A}^{-1}\left[\mathbf{y}-(\boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}}+\boldsymbol{b})\right]\right)\\
&= \frac{1}{\sqrt{|2\pi\boldsymbol{A}\boldsymbol{\Sigma}_{\mathbf{x}}\boldsymbol{A}^{\top}|}}\exp\left(-\frac{1}{2}\left[\mathbf{y}-(\boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}}+\boldsymbol{b})\right]^{\top}\left(\boldsymbol{A}\boldsymbol{\Sigma}_{\mathbf{x}}\boldsymbol{A}^{\top}\right)^{-1}\left[\mathbf{y}-(\boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}}+\boldsymbol{b})\right]\right)
\end{aligned}$$

Comparing this pdf with equation 2.74, one can immediately read out the mean

$$\boldsymbol{\mu}_{\mathbf{y}} = \boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}} + \boldsymbol{b}$$

and the covariance

$$\boldsymbol{\Sigma}_{\mathbf{y}} = \boldsymbol{A}\boldsymbol{\Sigma}_{\mathbf{x}}\boldsymbol{A}^{\top}$$

of the random vector $\mathbf{y}$ confirming the linearity of multivariate Gaussian transformations:

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}} + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Sigma}_{\mathbf{x}}\boldsymbol{A}^{\top})$$

□

2.3.1 Kalman Filter

The *Kalman Filter* (KF) is probably the most popular implementation of Gaussian filters. It was invented and therefore named by R. E. Kálmán in 1960 [21] by means of P. Swerling and his work in 1959 [22]. The Kalman Filter is a technique for prediction and filtering in order to obtain a believed state in *continuous linear Gaussian systems.*

The following three assumptions are made in continuous linear Gaussian systems used for Kalman Filters [1]:

- *initial probability*: $\mathrm{P}\{\mathbf{x}_0\}$
  The believed state is assumed to be initially normally distributed:
  $$\mathrm{bel}(\mathbf{x}_0) = \mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_0}, \boldsymbol{\Sigma}_{\mathbf{x}_0}) \tag{2.79}$$
- *transition probability*: $\mathrm{P}\{\mathbf{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t\}$
  The state transition probability is assumed to be linear with added Gaussian control noise:
  $$\mathbf{x}_t = \boldsymbol{A}_t\boldsymbol{x}_{t-1} + \boldsymbol{B}_t(\boldsymbol{u}_t + \mathbf{c}_t) \tag{2.80}$$

$\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ are matrices of size $n \times n$ and $n \times m$, respectively, and $\mathbf{c}_t$ denotes a zero-mean multivariate normally distributed control noise of dimension m and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{c}_t}$ of size $m \times m$.
Notice: $\mathbf{c}_t$ reflects the mentioned uncertainty introduced by state transitions.

Thus, we derive by equation 2.21 for the expectation

$$\boldsymbol{\mu}_{\mathbf{x}_t} = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{B}_t \boldsymbol{u}_t \tag{2.81}$$

and by equation 2.58 for the covariance

$$\boldsymbol{\Sigma}_{\mathbf{x}_t} = \boldsymbol{B}_t \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{B}_t^\top \tag{2.82}$$

giving us by means of equations 2.77 and 2.78 the multivariate Gaussian:

$$\mathbf{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_t \sim \mathcal{N}(\boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{B}_t \boldsymbol{u}_t, \boldsymbol{B}_t \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{B}_t^\top) \tag{2.83}$$

- *measurement probability*: $\mathrm{P}\{\mathbf{z}_t | \boldsymbol{x}_t\}$
  The state measurement probability is assumed to be linear with added Gaussian measurement noise:

  $$\mathbf{z}_t = \boldsymbol{C}_t \boldsymbol{x}_t + \mathbf{d}_t \tag{2.84}$$

  $\boldsymbol{C}_t$ is a matrix of size $k \times n$ and $\mathbf{d}_t$ denotes a zero-mean multivariate normally distributed measurement noise of dimension k and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{d}_t}$ of size $k \times k$.
  Notice: $\mathbf{d}_t$ reflects the mentioned uncertainty coming along with measurements.

  Thus, we derive by equation 2.21 for the expectation

  $$\boldsymbol{\mu}_{\mathbf{z}_t} = \boldsymbol{C}_t \boldsymbol{x}_t \tag{2.85}$$

  and for the covariance

  $$\boldsymbol{\Sigma}_{\mathbf{z}_t} = \boldsymbol{\Sigma}_{\mathbf{d}_t} \tag{2.86}$$

  giving us by means of equations 2.77 and 2.78 the multivariate Gaussian:

  $$\mathbf{z}_t | \boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{C}_t \boldsymbol{x}_t, \boldsymbol{\Sigma}_{\mathbf{d}_t}) \tag{2.87}$$

In [1], it was shown that these three assumptions suffice to guarantee Gaussian distributions for the predicted state

$$\overline{\mathrm{bel}}\,(\mathbf{x}_t) \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}, \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}) = \mathcal{N}(\boldsymbol{A}_t \boldsymbol{\mu}_{\mathbf{x}_{t-1}} + \boldsymbol{B}_t \boldsymbol{u}_t, \boldsymbol{A}_t \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} \boldsymbol{A}_t^\top + \boldsymbol{B}_t \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{B}_t^\top) \tag{2.88}$$

and the actual believed state

$$\mathrm{bel}\,(\mathbf{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_t}, \boldsymbol{\Sigma}_{\mathbf{x}_t}) = \mathcal{N}(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t} + \boldsymbol{K}_t(\boldsymbol{z}_t - \boldsymbol{C}_t \bar{\boldsymbol{\mu}}_{\mathbf{x}_t}), (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{C}_t) \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}) \tag{2.89}$$

where

$$\boldsymbol{K}_t = \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} \boldsymbol{C}_t^\top (\boldsymbol{C}_t \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} \boldsymbol{C}_t^\top + \boldsymbol{\Sigma}_{\mathbf{d}_t})^{-1} \tag{2.90}$$

denotes the *Kalman gain*, a matrix of size $n \times k$. The Kalman gain determines the impact of the current measurement $\boldsymbol{z}_t$ when correcting the predicted state $\overline{\text{bel}}(\mathbf{x}_t)$.

From equations 2.88 to 2.90, one can directly derive algorithm 2.1 for the Kalman Filter. Line 1 and 2 implement the prediction step 2.71 based on 2.88, in line 3 the Kalman gain $\boldsymbol{K}_t$ is calculated and line 4 and 5 implement the correction step 2.72 based on 2.89.

The computational complexity of algorithm 2.1 is determined by the cubic complexity of matrix multiplication and inversion. Because of the symmetric nature of covariance matrices, they can be stored in triangular matrices. Multiplications with triangular matrices can be performed in quadratic time [23]. The same holds for sparse matrices as assumed for $\boldsymbol{K}_t\boldsymbol{C}_t$ in line 5. Furthermore, today's most efficient matrix inversion algorithms solve the inversion in line 3 in $\mathcal{O}(k^{2.4})$ time [24]. Thus, the complexity of the Kalman Filter algorithm can be stated by $\mathcal{O}(n^2 + k^{2.4})$, where n denotes the dimension of the random state vector $\mathbf{x}_t$ and k the dimension of the measurement noise $\mathbf{d}_t$.

Algorithm 2.1: Kalman Filter

input : $\boldsymbol{\mu}_{\mathbf{x}_{t-1}}$, $\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}$, $\boldsymbol{u}_t$, $\boldsymbol{z}_t$
output: $\boldsymbol{\mu}_{\mathbf{x}_t}$, $\boldsymbol{\Sigma}_{\mathbf{x}_t}$

1 $\bar{\boldsymbol{\mu}}_{\mathbf{x}_t} = \boldsymbol{A}_t\boldsymbol{\mu}_{\mathbf{x}_{t-1}} + \boldsymbol{B}_t\boldsymbol{u}_t$;
2 $\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} = \boldsymbol{A}_t\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}\boldsymbol{A}_t^\top + \boldsymbol{B}_t\boldsymbol{\Sigma}_{\mathbf{c}_t}\boldsymbol{B}_t^\top$;
3 $\boldsymbol{K}_t = \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}\boldsymbol{C}_t^\top(\boldsymbol{C}_t\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}\boldsymbol{C}_t^\top + \boldsymbol{\Sigma}_{\mathbf{d}_t})^{-1}$;
4 $\boldsymbol{\mu}_{\mathbf{x}_t} = \bar{\boldsymbol{\mu}}_{\mathbf{x}_t} + \boldsymbol{K}_t(\boldsymbol{z}_t - \boldsymbol{C}_t\bar{\boldsymbol{\mu}}_{\mathbf{x}_t})$;
5 $\boldsymbol{\Sigma}_{\mathbf{x}_t} = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{C}_t)\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}$;

Informally one can imagine the prediction step as applying the linear transition 2.81 to the most probable previous state - the expectation $\boldsymbol{\mu}_{\mathbf{x}_{t-1}}$. The previous uncertainty presented by $\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}$ is transformed in a way considering the covariance's quadratic nature and adding additional uncertainty depicted by $\boldsymbol{\Sigma}_{\mathbf{c}_t}$, which has to be transformed in the same way. In the correction step, the currently predicted state denoted by $\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}$ is adjusted by the deviation of the predicted measurement $\boldsymbol{C}_t\ \bar{\boldsymbol{\mu}}_{\mathbf{x}_t}$ and the actual measurement $\boldsymbol{z}_t$ according to the Kalman gain $\boldsymbol{K}_t$. The final uncertainty is obtained by incorporating the measurements uncertainty depicted by $\boldsymbol{\Sigma}_{\mathbf{d}_t}$ into the previous uncertainty denoted by $\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}$ via the Kalman gain $\boldsymbol{K}_t$ again.

2.3.2 Extended Kalman Filter

In the previous subsection we have seen that linear functions for the transition and measurement probability are essential for Kalman Filters. These assumptions allow the computed believed state to stay Gaussian during the progress of time and, thus, making Kalman Filters so efficient. Unfortunately, linear transition and measurement functions cannot be guaranteed in practice. Hence, in order to extend the Kalman Filter approach to a wider area of application these assumptions are omitted, which is therefore referred to as *Extended Kalman Filter* (EKF).

Consequently, in EKF in general non-linear functions for the transition and measurement probability are allowed. The central idea in EKF is now to re-establish the conditions for the achievements of Kalman Filters introduced in the previous subsection by *linearization*. For linearization, several methods in literature exist. EKF makes usage of the *Taylor series*

$$g(x) \approx \sum_{n=0}^{\infty} \frac{g^{(n)}(a)}{n!}(x-a)^n \tag{2.91}$$

of a function g around a point a where $g^{(n)}(a)$ denotes the n^{th} derivative of the function evaluated at a and $n!$ the n^{th} factorial with $0! = 1$.

In particular, the *first order Taylor expansion*

$$g(x) \approx g(a) + g'(a)(x-a) \tag{2.92}$$

is used within EKF in order to re-establish linear functions. Obviously, this is just a rough approximation, which may be appropriate but sometimes also a source of error.

Notice: In the case of a multidimensional vector-valued function

$$\boldsymbol{g}\left(\boldsymbol{x}\right) = \begin{pmatrix} g_1(x_1, \ldots, x_n) \\ g_2(x_1, \ldots, x_n) \\ \vdots \\ g_m(x_1, \ldots, x_n) \end{pmatrix} \tag{2.93}$$

the first derivative

$$\boldsymbol{g}'(\boldsymbol{x}) = \frac{d\boldsymbol{g}}{d\boldsymbol{x}} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_n} \end{pmatrix} \tag{2.94}$$

evaluates to the *Jacobian matrix* consisting of all first-order partial derivatives.

Let us now consider the transition and measurement probability in the scope of EKF as we did for Kalman Filters in detail:

- *transition probability*: $\mathrm{P}\left\{\mathbf{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_t\right\}$
  The state transition probability assumption is weakened in a way accepting in general non-linear transitions with added Gaussian control noise:

  $$\mathbf{x}_t = \boldsymbol{g}\left(\boldsymbol{x}_{t-1}, \bar{\mathbf{u}}_t\right) = \boldsymbol{g}\left(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t + \mathbf{c}_t\right) \tag{2.95}$$

  Again, $\mathbf{c}_t$ denotes a zero-mean multivariate normally distributed control noise of dimension m and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{c}_t}$ of size $m \times m$ reflecting the mentioned uncertainty introduced by state transitions.

This transition is approximated by linearizing it around the most probable values of its arguments at time t. Obviously, this is the expected value $\boldsymbol{\mu}_{\mathbf{x}_{t-1}}$ of the previous believed state $\text{bel}(\mathbf{x}_{t-1})$ and the current control command $\boldsymbol{u}_t$. From this follows

$$\mathbf{x}_t \approx \boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right) + \boldsymbol{G}_t \begin{pmatrix} \boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{\mathbf{x}_{t-1}} \\ \mathbf{c}_t \end{pmatrix} \tag{2.96}$$

$$\approx \boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right) + \boldsymbol{G}_{x_{t-1}}(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{\mathbf{x}_{t-1}}) + \boldsymbol{G}_{u_t}\mathbf{c}_t \tag{2.97}$$

with

$$\boldsymbol{G}_t = \frac{d\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)}{d\left(\boldsymbol{x}_{t-1}^\top \ \bar{\mathbf{u}}_t^\top\right)^\top} = \left(\frac{d\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)}{d\boldsymbol{x}_{t-1}} \ \frac{d\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)}{d\bar{\mathbf{u}}_t}\right) = \left(\boldsymbol{G}_{\boldsymbol{x}_{t-1}} \ \boldsymbol{G}_{\boldsymbol{u}_t}\right) \tag{2.98}$$

denoting the Jacobian matrix of function $\boldsymbol{g}$.

Thus, we derive by equation 2.21 for the expectation

$$\boldsymbol{\mu}_{\mathbf{x}_t} = \boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right) + \boldsymbol{G}_{\boldsymbol{x}_{t-1}}(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{\mathbf{x}_{t-1}}) \tag{2.99}$$

and by equation 2.58 for the covariance

$$\boldsymbol{\Sigma}_{\mathbf{x}_t} = \boldsymbol{G}_{\boldsymbol{u}_t}\boldsymbol{\Sigma}_{\mathbf{c}_t}\boldsymbol{G}_{\boldsymbol{u}_t}^\top \tag{2.100}$$

giving us by means of equations 2.77 and 2.78 the multivariate Gaussian:

$$\mathbf{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t \sim \mathcal{N}(\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right) + \boldsymbol{G}_{\boldsymbol{x}_{t-1}}(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{\mathbf{x}_{t-1}}), \boldsymbol{G}_{\boldsymbol{u}_t}\boldsymbol{\Sigma}_{\mathbf{c}_t}\boldsymbol{G}_{\boldsymbol{u}_t}^\top) \tag{2.101}$$

- *measurement probability*: $\text{P}\left\{\mathbf{z}_t|\boldsymbol{x}_t\right\}$
  The state measurement probability assumption is weakened in a way accepting in general non-linear functions with added Gaussian measurement noise:

$$\mathbf{z}_t = \boldsymbol{h}\left(\boldsymbol{x}_t\right) + \mathbf{d}_t \tag{2.102}$$

  Again, $\mathbf{d}_t$ denotes a zero-mean multivariate normally distributed measurement noise of dimension k and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{d}_t}$ of size $k \times k$ reflecting the mentioned uncertainty introduced by measurements.

  Just like before, this function is approximated by linearizing it around the most probable value of its argument at time t, which is in this case the mean $\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}$ of the currently predicted state $\overline{\text{bel}}(\mathbf{x}_t)$. Hence, we can re-state

$$\mathbf{z}_t \approx \boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\right) + \boldsymbol{H}_t(\boldsymbol{x}_t - \bar{\boldsymbol{\mu}}_{\mathbf{x}_t}) + \mathbf{d}_t \tag{2.103}$$

  with

$$\boldsymbol{H}_t = \frac{d\boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\right)}{d\boldsymbol{x}_t} \tag{2.104}$$

  denoting the Jacobian matrix of function $\boldsymbol{h}$.

Thus, we derive by equation 2.21 for the expectation

$$\boldsymbol{\mu}_{\mathbf{z}_t} = \boldsymbol{h}\left(\boldsymbol{x}_t\right) \approx \boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\right) + \boldsymbol{H}_t(\boldsymbol{x}_t - \bar{\boldsymbol{\mu}}_{\mathbf{x}_t}) \tag{2.105}$$

and for the covariance

$$\boldsymbol{\Sigma}_{\mathbf{z}_t} = \boldsymbol{\Sigma}_{\mathbf{d}_t} \tag{2.106}$$

giving us by means of equations 2.77 and 2.78 the multivariate Gaussian:

$$\mathbf{z}_t|\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\right) + \boldsymbol{H}_t(\boldsymbol{x}_t - \bar{\boldsymbol{\mu}}_{\mathbf{x}_t}), \boldsymbol{\Sigma}_{\mathbf{d}_t}) \tag{2.107}$$

By simple comparison of equations 2.88, 2.89 and 2.90 of the Kalman Filter with equations 2.101 and 2.107, respectively, we derive

$$\overline{\text{bel}}\left(\mathbf{x}_t\right) \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}, \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}) = \mathcal{N}(\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right), \boldsymbol{G}_{x_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} \boldsymbol{G}_{x_{t-1}}^\top + \boldsymbol{G}_{u_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{G}_{u_t}^\top) \tag{2.108}$$

for the believed predicted state,

$$\text{bel}\left(\mathbf{x}_t\right) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_t}, \boldsymbol{\Sigma}_{\mathbf{x}_t}) = \mathcal{N}(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t} + \boldsymbol{K}_t(\boldsymbol{z}_t - \boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\right)), (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t)\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}) \tag{2.109}$$

for the actual believed state and

$$\boldsymbol{K}_t = \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} \boldsymbol{H}_t^\top (\boldsymbol{H}_t \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} \boldsymbol{H}_t^\top + \boldsymbol{\Sigma}_{\mathbf{d}_t})^{-1} \tag{2.110}$$

for the Kalman gain of the Extended Kalman Filter.

Similar to the previous subsection, we can now easily specify algorithm 2.2 for the Extended Kalman Filter from equations 2.108, 2.109 and 2.110. As we will see in succeeding chapters, $\boldsymbol{g}$ and $\boldsymbol{G}_t$, respectively, apply the given *motion model*, and $\boldsymbol{h}$ and $\boldsymbol{H}_t$, respectively, apply the *measurement model*, which implements a coordinate transformation from the world's coordinate system into the vehicle's coordinate system.

Algorithm 2.2: Extended Kalman Filter

input : $\boldsymbol{\mu}_{\mathbf{x}_{t-1}}$, $\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}$, $\boldsymbol{u}_t$, $\boldsymbol{z}_t$
output: $\boldsymbol{\mu}_{\mathbf{x}_t}$, $\boldsymbol{\Sigma}_{\mathbf{x}_t}$

1 $\bar{\boldsymbol{\mu}}_{\mathbf{x}_t} = \boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)$;
2 $\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} = \boldsymbol{G}_{x_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} \boldsymbol{G}_{x_{t-1}}^\top + \boldsymbol{G}_{u_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{G}_{u_t}^\top$;
3 $\boldsymbol{K}_t = \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} \boldsymbol{H}_t^\top (\boldsymbol{H}_t \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} \boldsymbol{H}_t^\top + \boldsymbol{\Sigma}_{\mathbf{d}_t})^{-1}$;
4 $\boldsymbol{\mu}_{\mathbf{x}_t} = \bar{\boldsymbol{\mu}}_{\mathbf{x}_t} + \boldsymbol{K}_t(\boldsymbol{z}_t - \boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\right))$;
5 $\boldsymbol{\Sigma}_{\mathbf{x}_t} = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t)\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}$;

Based on the Kalman Filter and especially its extension, we are now able to present a solution for the SLAM problem based on EKF in the next chapter

CHAPTER 3

EKF-SLAM using Visual Markers

EKF-SLAM is a technique solving the Simultaneous Localization and Mapping problem by exploiting Extended Kalman Filters as introduced in the previous chapter. For the purpose of this thesis, it is assumed that the explored environments contain visual markers as features for localizing and for creating 2D feature-based maps. Therefore, the robot is equipped with a visual marker detection. Such a setting is depicted in figure 3.1.

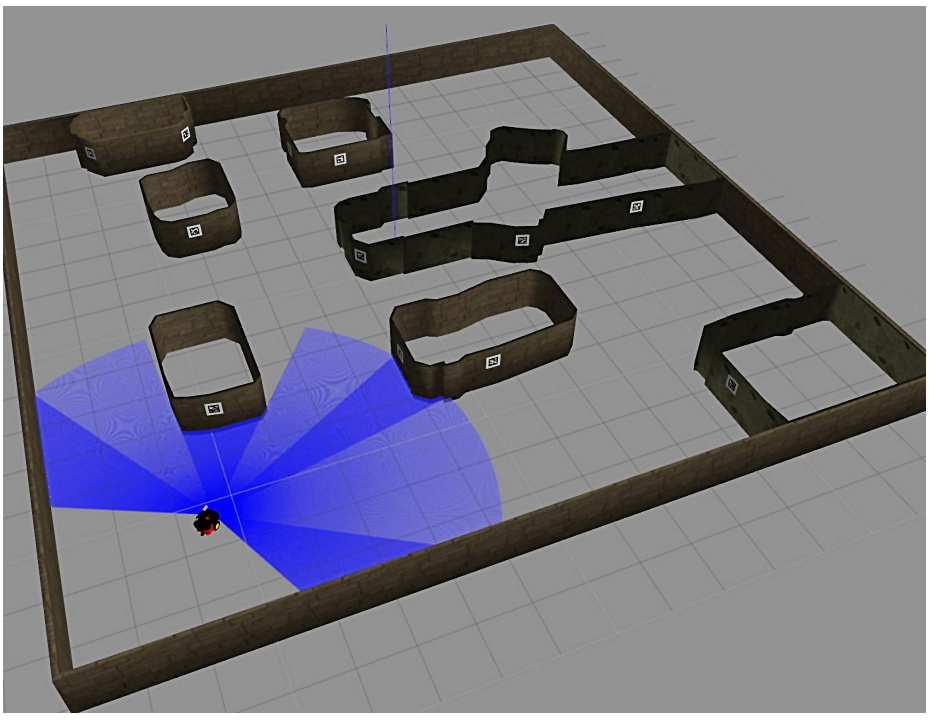


Figure 3.1: Setting of EKF-SLAM using visual markers

This chapter introduces the theoretical foundations of EKF-SLAM based on [1] used later on for the implementation. In doing so, the concepts of Bayesian filters and their realization in EKFs presented in the previous chapter are extended to the SLAM topic.

Before getting into the actual topic, section 3.1 briefly sums up the theoretical setting of SLAM. In section 3.2, the SLAM setting is adapted to fit for EKF-SLAM and a generic solution for the problem is presented. This generic solution is then elaborated in section 3.3 for known correspondences between map features and measurements using the IDs of visual markers. Finally, section 3.4 extends the presented solution by incorporating also measurements containing no IDs of visual markers and thus supporting also unknown correspondences.

3.1 SLAM

In SLAM, the HMM underlying Bayesian filters introduced in chapter 2 and shown in figure 2.5 is extended by a map $\mathbf{m}$. Figure 3.2 depicts this new setting where extracts of the map are received via the cyclic measurements. Again, shaded nodes denote given evidence of the correspondent random variable in the sense of conditional probability introduced in chapter 2. The remaining nodes represent the actual target of SLAM:

- robot's pose $\mathbf{x}_t$
- map $\mathbf{m}$

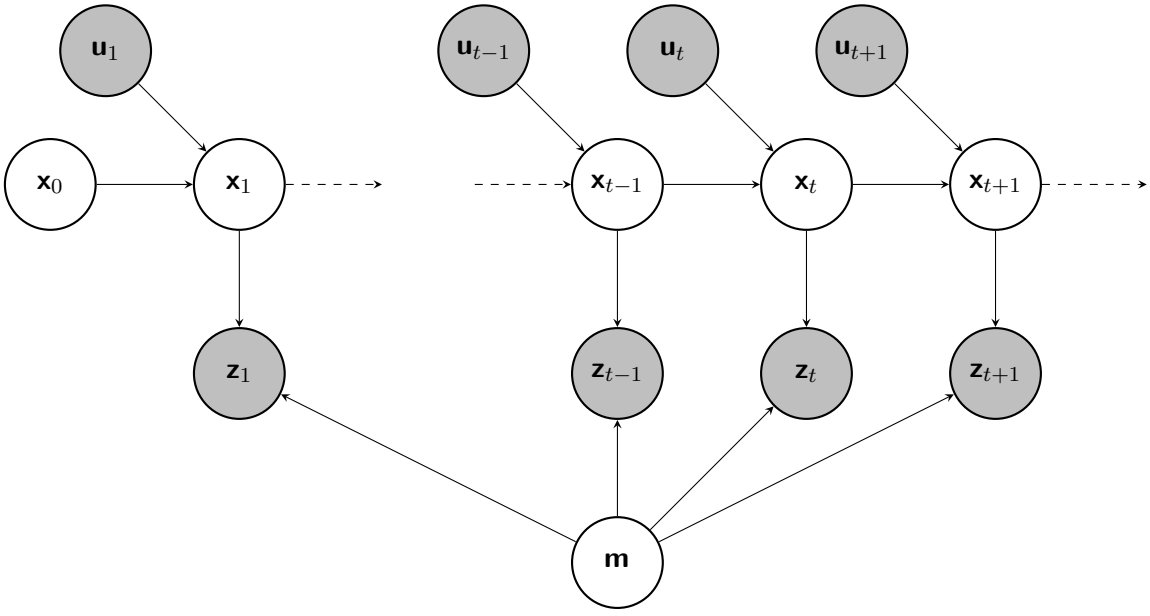


Figure 3.2: Hidden Markov Model underlying SLAM [1]

This setup allows us now to distinguish between *full* SLAM

$$\mathbf{x}_{0:t}, \mathbf{m} | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t} \tag{3.1}$$

and *online* SLAM:

$$\mathbf{x}_t, \mathbf{m} | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t} \tag{3.2}$$

In the full SLAM problem, the whole trajectory of the vehicle during exploration of its environment is desired. For the online SLAM problem, only the current vehicle's pose is of interest. Thus, online SLAM is just a special case of full SLAM. In particular, the pdf of online SLAM is just the marginalization of the pdf of full SLAM:

$$f\left(\boldsymbol{x}_t, \boldsymbol{m} | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}\right) = \int_{\mathbb{R}^n} \dots \int_{\mathbb{R}^n} f\left(\boldsymbol{x}_{0:t}, \boldsymbol{m} | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}\right) d\boldsymbol{x}_{t-1} \dots d\boldsymbol{x}_0 \tag{3.3}$$

In practice the full SLAM is hardly relevant because of its computational complexity. But for the online SLAM problem several techniques exist, which reduce the computational complexity by exploiting the recursive formula of Bayesian filters introduced in equation 2.70 of the previous chapter. One of these techniques is EKF-SLAM covered in this thesis. In order to apply equation 2.71 for the prediction step and equation 2.72 for the correction step of Bayesian filters to online SLAM, we have to adapt the predicted and believed state.

In particular, the predicted state introduced in equation 2.68 is enhanced with the map $\mathbf{m}$:

$$\overline{\text{bel}}\left(\mathbf{x}_t\right) = \mathbf{x}_t, \mathbf{m} | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t-1} \tag{3.4}$$

The same applies to the believed state introduced in equation 2.65:

$$\text{bel}\left(\mathbf{x}_t\right) = \mathbf{x}_t, \mathbf{m} | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t} \tag{3.5}$$

This adaptation is also often emphasized by replacing the state vector $\mathbf{x}_t$ with an extended state vector

$$\mathbf{y}_t = \begin{pmatrix} \mathbf{x}_t \\ \mathbf{m} \end{pmatrix} \tag{3.6}$$

in the recursive formula of Bayesian filters.

3.2 EKF-SLAM

In EKF-SLAM, the estimated map $\mathbf{m}$ is a composition of N map features:

$$\mathbf{m} = \left(\mathbf{m}_1^\top \dots \mathbf{m}_N^\top\right)^\top \tag{3.7}$$

Since this thesis deals in detail with EKF-SLAM in 2D, a feature's absolute pose in the map can be given in *Cartesian coordinates* by

$$\mathbf{m}_j = \begin{pmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{\theta} \end{pmatrix}, \ \mathbf{\Sigma}_{\mathbf{m}_j} = \begin{pmatrix} \sigma_{\mathsf{x}}^2 & \Sigma_{\mathsf{xy}} & \Sigma_{\mathsf{x\theta}} \\ \Sigma_{\mathsf{yx}} & \sigma_{\mathsf{y}}^2 & \Sigma_{\mathsf{y\theta}} \\ \Sigma_{\mathsf{\theta x}} & \Sigma_{\mathsf{\theta y}} & \sigma_{\mathsf{\theta}}^2 \end{pmatrix}, \ j \in [1, N], \tag{3.8}$$

with x, $\mathsf{y} \in \mathbb{R}$ and $\mathsf{\theta} \in (-\pi, \pi)$ (cp. figure 3.3a). For the sake of completeness, the covariance matrix of the map feature is additionally depicted in equation 3.8.

Similarly, the estimated vehicle's pose can be indicated by

$$\mathbf{x} = \begin{pmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{\theta} \end{pmatrix}, \ \mathbf{\Sigma}_{\mathbf{x}} = \begin{pmatrix} \sigma_{\mathsf{x}}^2 & \Sigma_{\mathsf{xy}} & \Sigma_{\mathsf{x\theta}} \\ \Sigma_{\mathsf{yx}} & \sigma_{\mathsf{y}}^2 & \Sigma_{\mathsf{y\theta}} \\ \Sigma_{\mathsf{\theta x}} & \Sigma_{\mathsf{\theta y}} & \sigma_{\mathsf{\theta}}^2 \end{pmatrix} \tag{3.9}$$

with x, $\mathsf{y} \in \mathbb{R}$ and $\mathsf{\theta} \in (-\pi, \pi)$. Again, for the sake of completeness the vehicle's covariance matrix is additionally depicted in equation 3.9.

This allows us now to refine equation 3.6 for the extended state vector by

$$\mathbf{y} = \begin{pmatrix} \mathbf{x} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{pmatrix}, \ \mathbf{\Sigma}_{\mathbf{y}} = \begin{pmatrix} \mathbf{\Sigma}_{\mathbf{x}} & \mathbf{\Sigma}_{\mathbf{x}\mathbf{m}_1} & \cdots & \mathbf{\Sigma}_{\mathbf{x}\mathbf{m}_N} \\ \mathbf{\Sigma}_{\mathbf{m}_1\mathbf{x}} & \mathbf{\Sigma}_{\mathbf{m}_1} & \cdots & \mathbf{\Sigma}_{\mathbf{m}_1\mathbf{m}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{\Sigma}_{\mathbf{m}_N\mathbf{x}} & \mathbf{\Sigma}_{\mathbf{m}_N\mathbf{m}_1} & \cdots & \mathbf{\Sigma}_{\mathbf{m}_N} \end{pmatrix} \tag{3.10}$$

giving a random vector of dimension $3(N+1)$ and a covariance matrix of size $3(N+1) \times 3(N+1)$.

Furthermore, the control command is indicated by

$$\boldsymbol{u}_t = (v \ \omega)^\top \tag{3.11}$$

with v, $\omega \in \mathbb{R}$ denoting the given velocity and angular rate at time t, respectively.

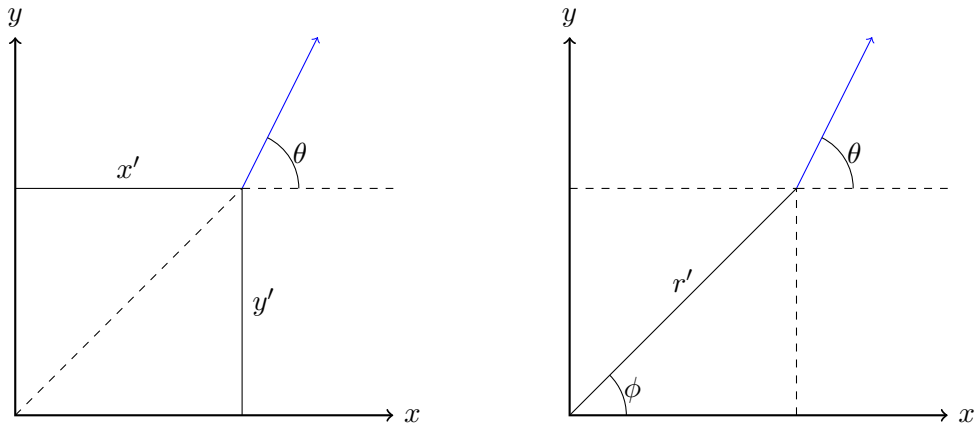
Finally, the measurement at time t

$$\boldsymbol{z}_t = (\boldsymbol{z}_{t,1}^\top \dots \boldsymbol{z}_{t,M}^\top)^\top \tag{3.12}$$

is a composition of

$$\boldsymbol{z}_{t,i} = (id \ r \ \phi \ \theta)^\top, \ i \in [1, M] \tag{3.13}$$

representing an observed feature's 2D pose in *spherical coordinates* (cp. figure 3.3b) with $r \in \mathbb{R}$ and ϕ, $\theta \in (-\pi, \pi)$ relative to the vehicle's current pose and an optional ID $id \in \{\mathbb{Z} \cup \emptyset\}$. The reason, why observed features are assumed to be given in spherical coordinates, is due to the way how measurements are taken. It is worth noting, that the number of observed features M is typically smaller than the actual number of map features N, except right at the beginning.



(a) Cartesian coordinates $(x'\ y'\ \theta)^\top$ (b) Spherical coordinates $(r\ \phi\ \theta)^\top$

Figure 3.3: Different representations of a 2D pose and their relationship

During the remaining thesis, $\mathbf{m}_j$ is called *map feature* or simply *feature* and $\boldsymbol{z}_{t,i}$ is called *observed feature* or *observation*. In principle, both are describing the mentioned visual markers. But in the first time they are considered from the map's point of view and the other time they are observed by the vehicle. Thus they are considered from the vehicle's point of view. equations 3.7 and 3.9 represent the actual target of the SLAM problem: localization and mapping. Further, equations 3.11 and 3.12 represent given evidence during the progress of time helping to achieve this goal. That is why the former are depicted as random vectors and the latter as deterministic vectors given at time t.

By means of this, we are now ready to introduce algorithm 3.1 for EKF-SLAM. The stated algorithm is also referred to as EKF-SLAM cycle, since it embraces all required steps repeated for each point in time. The algorithm itself is at this point rather generic and will be complemented in the next section with appropriate algorithms for the particular function calls. But at this moment it is sufficient for a general understanding.

Algorithm 3.1: EKF-SLAM

input : $\boldsymbol{\mu}_{\mathbf{y}_{t-1}}$, $\boldsymbol{\Sigma}_{\mathbf{y}_{t-1}}$, $\boldsymbol{u}_t$, $\boldsymbol{z}_t$
output: $\boldsymbol{\mu}_{\mathbf{y}_t}$, $\boldsymbol{\Sigma}_{\mathbf{y}_t}$

1 $(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}) =$ `Prediction(`$\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{\Sigma}_{\mathbf{y}_{t-1}}, \boldsymbol{u}_t$`);`
2 $\boldsymbol{c}_t =$ `Data Association(`$\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \boldsymbol{z}_t$`);`
3 $(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}) =$ `Update(`$\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}, \boldsymbol{z}_t, \boldsymbol{c}_t$`);`
4 $(\boldsymbol{\mu}_{\mathbf{y}_t}, \boldsymbol{\Sigma}_{\mathbf{y}_t}) =$ `Integration(`$\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}, \boldsymbol{z}_t, \boldsymbol{c}_t$`);`

Line 1 realizes the prediction step of Bayesian filters. In this step, the old believed state

$$\text{bel}\left(\mathbf{y}_{t-1}\right) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{\Sigma}_{\mathbf{y}_{t-1}}) \tag{3.14}$$

is moved according to the current control command $\boldsymbol{u}_t$, giving the predicted state:

$$\overline{\text{bel}}\,(\mathbf{y}_t) \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}) \tag{3.15}$$

In line 2, the correspondences between observed features $\boldsymbol{z}_{t,i}$ and map features $\mathbf{m}_j$ are found and stored in the correspondence vector

$$\boldsymbol{c}_t = (c_1 \dots c_M)^\top \tag{3.16}$$

with

$$c_i = \begin{cases} \emptyset & \text{observed feature } \boldsymbol{z}_{t,i} \text{ corresponds to no map feature} \\ j & \text{observed feature } \boldsymbol{z}_{t,i} \text{ corresponds to map feature } \mathbf{m}_j,\ j \in [1, N] \\ \text{new} & \text{observed feature } \boldsymbol{z}_{t,i} \text{ represents a new map feature} \end{cases} \tag{3.17}$$

for $i \in [1, M]$. The correction step of Bayesian filters happens in line 3. In this step, the previous predicted state is updated by the measurements $\boldsymbol{z}_t$ according to the found correspondences $\boldsymbol{c}_t$. Unlike in ordinary EKF this step returns just an intermediate believed state

$$\overline{\overline{\text{bel}}}\,(\mathbf{y}_t) \sim \mathcal{N}(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}) \tag{3.18}$$

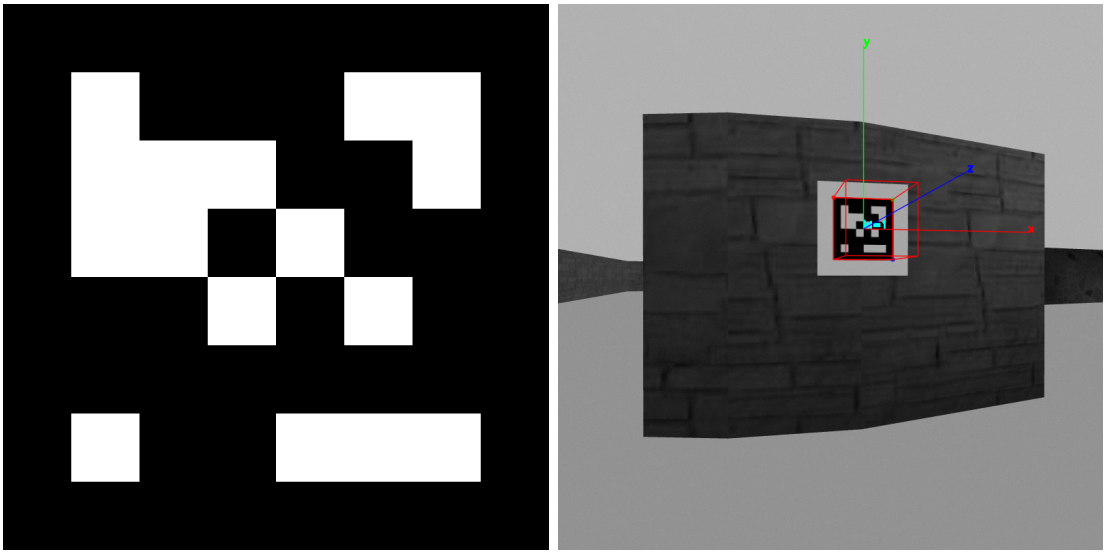
which still needs to be complemented by newly found features in $\boldsymbol{z}_t$ according to $\boldsymbol{c}_t$. This integration of new features into the current map is indicated in line 4 of the algorithm yielding the final believed state:

$$\text{bel}\,(\mathbf{y}_t) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{y}_t}, \boldsymbol{\Sigma}_{\mathbf{y}_t}) \tag{3.19}$$

In the next section, we will have a more detailed look on the so far roughly explained steps of algorithm 3.1. In doing so, we will recognize particular parts of the already introduced algorithm 2.2 for EKF. Assuming known correspondences between observations and map features allows us to focus on the essentials of EKF-SLAM.

3.3 Known Feature Correspondences

In this section, we will explicitly state algorithms for the functions called in the generic algorithm 3.1 of EKF-SLAM. At this moment the found correspondences between observed features and map features only base upon the recognized IDs of visual markers. Figure 3.4 depicts such a visual marker provided by ARToolKit [10] and used within this thesis. In each visual marker a unique ID is encoded. In the case the ID could not be detected, i.e. $\boldsymbol{z}_{t,i}.id = \emptyset,\ i \in [1, M]$ the particular measurement is neglected at this point. For this reason, we say the correspondences are known, because there is no uncertainty left.



(a) ARToolKit visual marker with ID 1 [10] (b) Observed visual marker in Gazebo

Figure 3.4: Visual marker used for features within the thesis

3.3.1 Prediction

The first function call in algorithm 3.1 indicates the prediction step of Bayesian filters. Thus, one can directly derive algorithm 3.2 for the prediction part of the EKF-SLAM cycle from lines 1 and 2 of algorithm 2.2 for EKF. But instead of $\mathbf{x}_t$ the extended random vector $\mathbf{y}_t$ introduced in equation 3.10 is now used.

Algorithm 3.2: Prediction

input : $\boldsymbol{\mu}_{\mathbf{y}_{t-1}}$, $\boldsymbol{\Sigma}_{\mathbf{y}_{t-1}}$, $\boldsymbol{u}_t$

output: $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$

1 $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t} = \bar{\boldsymbol{g}}\left(\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{u}_t\right)$;

2 $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t} = \bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{y}_{t-1}} \bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}}^{\top} + \bar{\boldsymbol{G}}_{\boldsymbol{u}_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \bar{\boldsymbol{G}}_{\boldsymbol{u}_t}^{\top}$;

The computational complexity of algorithm 3.2 highly depends on the dimension of the extended state vector $\mathbf{y}_t$ and thus on the feature number N of the map $\mathbf{m}$. Assuming linear complexity for $\bar{\boldsymbol{g}}\left(\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{u}_t\right)$ - which is realistic as we will see in chapter 4 - the crucial factor proves to be the matrix multiplication. Again, this can be accomplished in $\mathcal{O}(N^2)$ for triangular covariance matrices.

3.3.2 Data Association

In the next step of the EKF-SLAM cycle, the correspondences between measurements and map need to be found in order to correct the previous predicted state. This data

association is depicted in algorithm 3.3. As already mentioned, we are only interested in correspondences determined by perceived IDs of visual markers at this moment. This is ensured in line 3 of the algorithm, where all measurements without IDs are neglected, i.e. their correspondences stay initialized to $\emptyset$. In line 4, the actual correspondence between detected ID and map feature is tried to be established. This can be accomplished with a simple lookup table maintained during the progress of the EKF-SLAM cycle. Finally, depending on whether a correspondence is found or not, the correspondence c_i is either set accordingly or a new visual marker has been detected.

Algorithm 3.3: Data Association

input : $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t} = (\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\ \bar{\boldsymbol{\mu}}_{\mathbf{m}_1} \dots \bar{\boldsymbol{\mu}}_{\mathbf{m}_N})^\top$, $\boldsymbol{z}_t = (\boldsymbol{z}_{t,1} \dots \boldsymbol{z}_{t,M})^\top$
output: $\boldsymbol{c}_t = (c_1 \dots c_M)^\top$

```
1  c_t = (∅ ... ∅)^⊤;
2  for i = 1 to M do
3      if z_{t,i}.id ≠ ∅ then
4          j = index of feature μ̄_{m_j} corresponding to visual marker with ID z_{t,i}.id;
5          if j ≠ ∅ then // known visual marker
6              c_i = j;
7          else // new visual marker
8              c_i = new;
9          end
10     end
11 end
```

Obviously, the computational complexity of algorithm 3.3 is determined by the surrounding for-loop. Since the remaining assignments can be accomplished in constant time, we obtain an over all complexity of $\mathcal{O}(M)$. Keep in mind, the number of observations M is usually limited by the current number of map features N.

3.3.3 Update

As soon as we have associated the observed features with the map features, we can incorporate the measurements into the predicted state in order to correct it. This update is indicated in line 3 of algorithm 3.1 for EKF-SLAM and corresponds to the correction step in Bayesian filters. Hence, we can directly derive algorithm 3.4 for the update in EKF-SLAM cycle from lines 3 to 5 of algorithm 2.2 for EKF.

Similar to the prediction part of the EKF-SLAM cycle the random vector $\mathbf{x}_t$ is replaced by the extended random vector $\mathbf{y}_t$ introduced in equation 3.10. Furthermore the correspondence vector $\boldsymbol{c}_t$ is added in line 2 of algorithm 3.4. This indicates that the measurement function $\boldsymbol{h}$, which maps the map features to observed features, depends on their association.

Algorithm 3.4: Update

input : $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t$, $\boldsymbol{c}_t$
output: $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}$, $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$

1 $\boldsymbol{K}_t = \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t} \boldsymbol{H}_t^\top (\boldsymbol{H}_t \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t} \boldsymbol{H}_t^\top + \boldsymbol{\Sigma}_{\mathbf{d}_t})^{-1}$;
2 $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t} = \bar{\boldsymbol{\mu}}_{\mathbf{y}_t} + \boldsymbol{K}_t (\boldsymbol{z}_t - \boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \boldsymbol{c}_t\right))$;
3 $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t) \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$;

The computational complexity of algorithm 3.4 conforms with the complexity of algorithm 2.1 for the Kalman Filter: $\mathcal{O}(N^2 + M^{2.4})$. Since the number M of observations is typically at least one magnitude smaller than the overall number N of map features, it is most of the time simple denoted by $\mathcal{O}(N^2)$.

3.3.4 Integration

In the last step of the EKF-SLAM cycle depicted in algorithm 3.1 newly found map features are integrated into the intermediate believed state $\overline{\overline{\text{bel}}}\left(\mathbf{y}_t\right)$. Adding all new map features gives then the final believed state $\text{bel}\left(\mathbf{y}_t\right)$. For this purpose, we introduce the integration function $\bar{\boldsymbol{p}}$ as following:

$$\mathbf{y}_t = \bar{\boldsymbol{p}}\left(\bar{\bar{\mathbf{y}}}_t, \boldsymbol{z}_{t,i} + \mathbf{d}_{t,i}\right) = \bar{\boldsymbol{p}}\left(\bar{\bar{\mathbf{y}}}_t, \bar{\mathbf{z}}_{t,i}\right) \tag{3.20}$$

The function integrates the measurement $\boldsymbol{z}_{t,i}$ with a zero-mean multivariate Gaussian measurement noise $\mathbf{d}_{t,i}$ into the extended state vector $\bar{\bar{\mathbf{y}}}_t = (\mathbf{x}_t^\top \ \mathbf{m}_1^\top \dots \mathbf{m}_N^\top)^\top$ such that $\mathbf{y}_t = (\mathbf{x}_t^\top \ \mathbf{m}_1^\top \dots \mathbf{m}_N^\top \ \mathbf{m}_{N+1}^\top)^\top$ describes the augmented state vector. Similarly as we did for the EKF, this function is approximated by linearizing it around the most probable values of its arguments at time t. Obviously, this is the expected value $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}$ of the current believed state $\overline{\overline{\text{bel}}}\left(\mathbf{y}_t\right)$ and the actual measurement $\boldsymbol{z}_{t,i}$. From this follows

$$\begin{aligned} \mathbf{y}_t &\approx \bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right) + \bar{\boldsymbol{P}}_t (\bar{\bar{\mathbf{y}}}_t - \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t} \ \mathbf{d}_{t,i})^\top && (3.21) \\ &\approx \bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right) + \bar{\boldsymbol{P}}_{\boldsymbol{y}_t} (\bar{\bar{\mathbf{y}}}_t - \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}) + \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}} \mathbf{d}_{t,i} && (3.22) \end{aligned}$$

with

$$\bar{\boldsymbol{P}}_t = \frac{d\bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right)}{d\left(\bar{\bar{\mathbf{y}}}_t^\top \ \bar{\mathbf{z}}_{t,i}^\top\right)^\top} = \left(\frac{d\bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right)}{d\bar{\bar{\mathbf{y}}}_t} \ \frac{d\bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right)}{d\bar{\mathbf{z}}_{t,i}}\right) = \left(\bar{\boldsymbol{P}}_{\boldsymbol{y}_t} \ \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}}\right) \tag{3.23}$$

denoting the Jacobian matrix of function $\bar{\boldsymbol{p}}$.

Thus, by equation 2.21 we derive for the expectation

$$\boldsymbol{\mu}_{\mathbf{y}_t} = \bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right) \tag{3.24}$$

and by equation 2.23 for the covariance:

$$\boldsymbol{\Sigma}_{\mathbf{y}_t} = \bar{\boldsymbol{P}}_{\boldsymbol{y}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} \bar{\boldsymbol{P}}_{\boldsymbol{y}_t}^\top + \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}}^\top \tag{3.25}$$

From equations 3.24 and 3.25, one can now directly derive algorithm 3.5 for the integration part of the EKF-SLAM cycle. The for-loop in combination with line 4 ensures that all newly found map features are added to the current believed state. In lines 5 and 6 the actual integration happens.

Algorithm 3.5: Integration

input : $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}$, $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t = (\boldsymbol{z}_{t,1} \dots \boldsymbol{z}_{t,M})^\top$, $\boldsymbol{c}_t = (c_1 \dots c_M)^\top$
output: $\boldsymbol{\mu}_{\mathbf{y}_t}$, $\boldsymbol{\Sigma}_{\mathbf{y}_t}$

1 $\boldsymbol{\mu}_{\mathbf{y}_t} = \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}$;
2 $\boldsymbol{\Sigma}_{\mathbf{y}_t} = \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$;
3 **for** $i = 1$ **to** M **do**
4 | **if** $c_i \neq$ new **then continue**;
5 | $\boldsymbol{\mu}_{\mathbf{y}_t} = \bar{\boldsymbol{p}}\left(\boldsymbol{\mu}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right)$;
6 | $\boldsymbol{\Sigma}_{\mathbf{y}_t} = \bar{\boldsymbol{P}}_{\boldsymbol{y}_t} \boldsymbol{\Sigma}_{\mathbf{y}_t} \bar{\boldsymbol{P}}_{\boldsymbol{y}_t}^\top + \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}}^\top$;
7 **end**

The computational complexity of algorithm 3.5 is on the one hand determined by the surrounding for-loop and on the other hand by the matrix multiplications inside. This implicitly assumes linear complexity for $\bar{\boldsymbol{p}}\left(\boldsymbol{\mu}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right)$, similarly as we did in algorithm 3.2 for the prediction. This is realistic as we will see later on in chapter 4. Consequently, we can denote the complexity by $\mathcal{O}(M \cdot N^2)$. Nevertheless, lines 5 and 6 are executed exactly N times for establishing a map composed of N features. Thus, the complexity for creating the map itself is separated from the remaining EKF-SLAM cycle and denoted by $\mathcal{O}(N^3)$.

Under these considerations, summing up the complexities of the remaining algorithms gives an overall complexity of $\mathcal{O}(N^2)$ for the EKF-SLAM cycle depicted in algorithm 3.1. This is quiet efficient and enables us to reasonable implement the EKF-SLAM cycle later on.

In the next section, the introduced solution for EKF-SLAM with known correspondences will be enhanced by supporting also unknown correspondences. In doing so, unknown correspondences are tried to be resolved based on the poses of the measurements and map features. This enables us to benefit from all received measurements, rather than only observations containing a visual marker ID.

3.4 Unknown Feature Correspondences

Beside measurements coming along with visual marker IDs, one might also want to use the remaining observations without IDs in order to improve the update step. Otherwise these measurements are discarded, resulting in an information loss. For this purpose, this thesis will make use of the *Nearest Neighbor Standard Filter* (NNSF) [16] approach, which is basically a *maximum likelihood estimator* associating an observed feature with a map feature of minimal *Mahalanobis Distance*. The latter will be discussed in detail in subsection 3.4.1. Until then, we simply denote it as the *most probable* association. Consequently, algorithm 3.1 presented in section 3.2 has to be adapted to support an additional step enhancing the certain correspondence data $\boldsymbol{c}_t$ with further most probable correspondences resulting in $\bar{\boldsymbol{c}}_t$. For the sake of completeness, algorithm 3.6 recaps this adaptation.

Algorithm 3.6: EKF-SLAM (supporting unknown correspondences)

input : $\boldsymbol{\mu}_{\mathbf{y}_{t-1}}$, $\boldsymbol{\Sigma}_{\mathbf{y}_{t-1}}$, $\boldsymbol{u}_t$, $\boldsymbol{z}_t$
output: $\boldsymbol{\mu}_{\mathbf{y}_t}$, $\boldsymbol{\Sigma}_{\mathbf{y}_t}$

1 $(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}) =$ `Prediction(`$\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{\Sigma}_{\mathbf{y}_{t-1}}, \boldsymbol{u}_t$`);`
2 $\boldsymbol{c}_t =$ `Data Association(`$\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \boldsymbol{z}_t$`);`
3 $\bar{\boldsymbol{c}}_t =$ `NNSF(`$\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}, \boldsymbol{z}_t, \boldsymbol{c}_t$`);`
4 $(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}) =$ `Update(`$\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}, \boldsymbol{z}_t, \bar{\boldsymbol{c}}_t$`);`
5 $(\boldsymbol{\mu}_{\mathbf{y}_t}, \boldsymbol{\Sigma}_{\mathbf{y}_t}) =$ `Integration(`$\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}, \boldsymbol{z}_t, \bar{\boldsymbol{c}}_t$`);`

In line 3 of algorithm 3.6 the mentioned NNSF approach is invoked, yielding enhanced correspondence data $\bar{\boldsymbol{c}}_t$, which is then continued to be used instead of $\boldsymbol{c}_t$. Moreover, there are two slightly different versions of NNSF considered in this thesis, namely *local* and *global* NNSF [25].

The first one - depicted in algorithm 3.7 - examines every observed feature $\boldsymbol{z}_{t,i}$, which has not been associated yet with a map feature based on an ID and tries to find the most probable corresponding map feature $\mathbf{m}_j$. Again, only map features $\mathbf{m}_j$ are considered, which are not already associated with an observation including an ID. These two limitations are ensured in lines 3 and 7. The reason why algorithm 3.7 is labeled *local*, is because, it does not avoid local optima such as two observations $\boldsymbol{z}_{t,i}$ and $\boldsymbol{z}_{t,i'}$ without IDs associated with the same map feature $\mathbf{m}_j$. Typical proceedings covering this drawback take the nearer one of $\boldsymbol{z}_{t,i}$ and $\boldsymbol{z}_{t,i'}$ or reject both of them since both have been proven to be supposable. The latter one describes a more conservative approach, since the association decision is hard and cannot be withdrawn. Unfortunately, the data association has a tremendous impact on the the further evolution of the estimation process. We will see this in chapter 5, when discussing the results of the provided solution. Thus, the conservative proceeding was chosen for the implementation trying to avoid wrong associations.

Algorithm 3.7: Nearest Neighbor Standard Filter (local)

input : $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t = (\boldsymbol{z}_{t,1} \dots \boldsymbol{z}_{t,M})^\top$, $\boldsymbol{c}_t = (c_1 \dots c_M)^\top$
output: $\bar{\boldsymbol{c}}_t = (\bar{c}_1 \dots \bar{c}_M)^\top$

1 $\bar{\boldsymbol{c}}_t = \boldsymbol{c}_t$;
2 **for** $i = 1$ **to** M **do**
3 **if** $\boldsymbol{z}_{t,i}.id \neq \emptyset$ **then continue**;// no association needed
4 $min_j = \emptyset$;
5 $min_d = \infty$;
6 **for** $j = 1$ **to** N **do**
7 **if** $j \in \boldsymbol{c}_t$ **then continue**;// already associated
8 $d^2_{t,ij}$ = Mahalanobis Distance($\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t$, i, j);
9 **if** $d^2_{t,ij} < min_d$ **then**
10 $min_j = j$;
11 $min_d = d^2_{t,ij}$;
12 **end**
13 **end**
14 **if** $min_d < \gamma$ **then** $\bar{c}_i = min_j$;
15 **end**

The second version considered for the NNSF approach is depicted in algorithm 3.8. This version avoids local optima as described before by investigating each possible combination of observed features $\boldsymbol{z}_{t,i}$ and map features $\mathbf{m}_j$ in lines 10 to 14. Like before, only observed features $\boldsymbol{z}_{t,i}$ and map features $\mathbf{m}_j$ are considered, which have not been associated based on given IDs, yet. This is ensured in lines 2 to 9 with the help of the two arrays A and B. Then, in line 15 a minimum assignment presenting the most probable association of the considered features is obtained. Thus, line 15 describes an optimization problem, which is a typical task in the field of linear programming. In the case of the implementation corresponding to this thesis, an implementation of Munkre's algorithm [26], [27] is used for solving this optimization problem. Finally, in the lines 16 to 20 the correspondence data is updated.

It is worth noting, although the global NNSF approach avoids local optima, it still might yield wrong correspondences. Imagine, for instance, a huge uncertainty in the estimated pose of the robot as well as of the map features. Received measurements can be associated with any map feature. In this case, a correct association is rather unlikely. Thus, keeping the uncertainty small is a key issue when using NNSF.

Algorithm 3.8: Nearest Neighbor Standard Filter (global)

input : $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t = (\boldsymbol{z}_{t,1} \dots \boldsymbol{z}_{t,M})^\top$, $\boldsymbol{c}_t = (c_1 \dots c_M)^\top$
output: $\bar{\boldsymbol{c}}_t = (\bar{c}_1 \dots \bar{c}_M)^\top$

1 $\bar{\boldsymbol{c}}_t = \boldsymbol{c}_t$;
2 $k = 0$;
3 **for** $i = 1$ **to** M **do**
4 | **if** $\boldsymbol{z}_{t,i}.id = \emptyset$ **then** $\mathsf{A}[k++] = i$;`// else no association needed`
5 **end**
6 $k = 0$;
7 **for** $j = 1$ **to** N **do**
8 | **if** $j \notin \boldsymbol{c}_t$ **then** $\mathsf{B}[k++] = j$;`// else already associated`
9 **end**
10 **for** $x = 1$ **to** `sizeof(A)` **do**
11 | **for** $y = 1$ **to** `sizeof(B)` **do**
12 | | $\boldsymbol{D}[x][y] =$ `Mahalanobis Distance(`$\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t$, $\mathsf{A}[x]$, $\mathsf{B}[y]$`)`;
13 | **end**
14 **end**
15 $C =$ minimum assignment $\{(x_1, y_1), (x_2, y_2), \dots\}$ of $\boldsymbol{D}$;
16 **for** $(x, y) \in C$ **do**
17 | $i = \mathsf{A}[x]$;
18 | $j = \mathsf{B}[y]$;
19 | **if** $\boldsymbol{D}[x][y] < \gamma$ **then** $\bar{c}_i = j$;
20 **end**

3.4.1 Mahalanobis Distance

Until now, we have not got into detail what is meant with the *most probable* correspondence and how it is determined. Algorithm 3.7 as well as algorithm 3.8 use for this purpose both the *Mahalanobis distance*

$$\mathsf{d} = \sqrt{(\mathbf{r} - \boldsymbol{\mu}_\mathbf{r})^\top \boldsymbol{\Sigma}_\mathbf{r}^{-1} (\mathbf{r} - \boldsymbol{\mu}_\mathbf{r})} \tag{3.26}$$

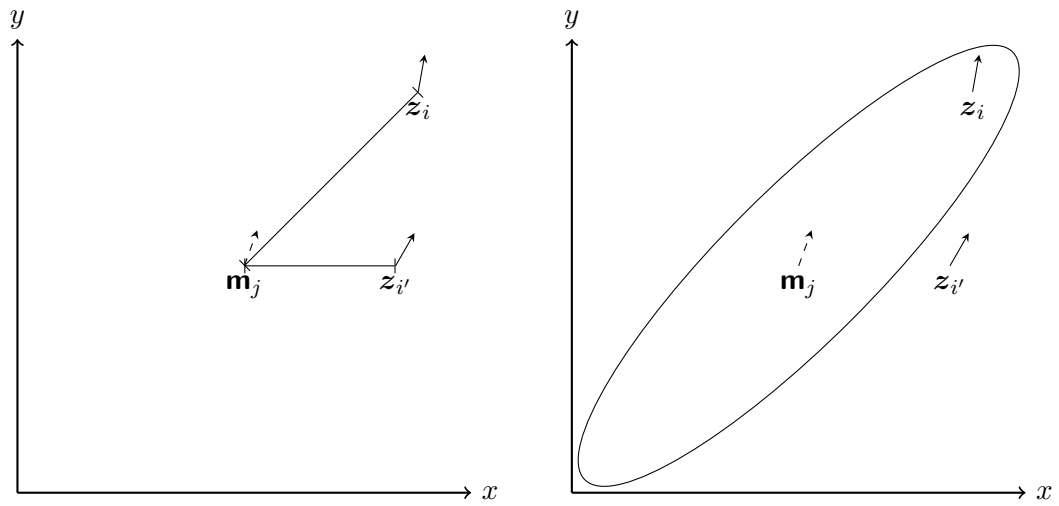
in line 8 and 12, respectively, denoting the distance between a random vector $\mathbf{r}$ and its expectation $\boldsymbol{\mu}_\mathbf{r}$. Similar to the Euclidean distance

$$d = \sqrt{(\boldsymbol{a} - \boldsymbol{b})^\top (\boldsymbol{a} - \boldsymbol{b})} \tag{3.27}$$

between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ a smaller Mahalanobis distance indicates more probable correspondences. But in contrast to the Euclidean distance, the Mahalanobis distance also incorporates the inherent uncertainty depicted in $\boldsymbol{\Sigma}_\mathbf{r}$ coming along in the task of SLAM. That is, why the Mahalanobis distance is expressed by a random variable. Using the Mahalanobis distance instead of the Euclidean distance may yield contradictory results as figure 3.5 illustrates. Although figure 3.5a suggests associating the map feature

$\mathbf{m}_j$ with the observation $\boldsymbol{z}_{i'}$, figure 3.5b suggests the opposite under consideration of the given uncertainty.

Notice: The ellipse in figure 3.5b indicates the assumed distribution for measurements of the map feature $\mathbf{m}_j$ based on the measurement noise. We will come back to this soon.



(a) Euclidean distance (b) Mahalanobis distance

Figure 3.5: Euclidean distance vs. Mahalanobis distance

In order to derive an expression for the Mahalanobis distance between a map feature $\mathbf{m}_j$ and an observation $\boldsymbol{z}_i$ let us assume for the remaining thesis independence between the individual observed features of a measurement at time t as suggested in [1]:

$$f\left(\boldsymbol{z}_t\right) = \prod_{i}^{M} f\left(\boldsymbol{z}_{t,i}\right) \tag{3.28}$$

Typically, this is a good approximation, especially for static environments as covered in this thesis. Apart from that, it enables us to specify the overall measurement probability during one cycle of EKF-SLAM by its individuals:

$$f\left(\boldsymbol{z}_t | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t-1}\right) = \prod_{i}^{M} f\left(\boldsymbol{z}_{t,i} | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t-1}\right) \tag{3.29}$$

The individual measurement probabilities for the Mahalanobis distance are now deduced

as follows:

$$
\begin{aligned}
f\left(\boldsymbol{z}_{t,i}|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right) &= \int_{\mathbb{R}^{3(N+1)}} f\left(\boldsymbol{z}_{t,i},\boldsymbol{y}_t|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right) d\boldsymbol{y}_t && 2.48\\
&= \int_{\mathbb{R}^{3(N+1)}} f\left(\boldsymbol{z}_{t,i}|\boldsymbol{y}_t,\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right)\ f\left(\boldsymbol{y}_t|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right) d\boldsymbol{y}_t && 2.46\\
&= \int_{\mathbb{R}^{3(N+1)}} f\left(\boldsymbol{z}_{t,i}|\boldsymbol{y}_t\right)\ f\left(\boldsymbol{y}_t|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1}\right) d\boldsymbol{y}_t && 2.51
\end{aligned}
$$

This integral describes a convolution of two multivariate Gaussians similar to equation 2.71 of the prediction step. The first multivariate Gaussian

$$\mathbf{z}_{t,i}|\boldsymbol{y}_t \sim \mathcal{N}(\boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right) + \boldsymbol{H}_{t,j}(\boldsymbol{y}_t - \bar{\boldsymbol{\mu}}_{\mathbf{y}_t}), \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}}) \tag{3.30}$$

can be derived by the equations 2.105 and 2.106 with

$$\boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right) = \begin{pmatrix} \boldsymbol{h}_{j_1}\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right) \\ \boldsymbol{h}_{j_2}\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right) \\ \vdots \end{pmatrix}, \quad \boldsymbol{H}_{t,j} = \frac{d\boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right)}{d\boldsymbol{y}_t} = \begin{pmatrix} \boldsymbol{H}_{t,j_1} \\ \boldsymbol{H}_{t,j_2} \\ \vdots \end{pmatrix} \tag{3.31}$$

and

$$\boldsymbol{\Sigma}_{\mathbf{d}_t} = \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i_1}} & \mathbf{0} & \cdots \\ \mathbf{0} & \boldsymbol{\Sigma}_{\mathbf{d}_{t,i_2}} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{3.32}$$

for $j_1, j_2, \ldots \in [1, N]$ and $i_1, i_2, \ldots \in [1, M]$, because of the independent measurement assumption. The second multivariate Gaussian

$$\mathbf{y}_t|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1} \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}) \tag{3.33}$$

describes the predicted state $\overline{\text{bel}}\left(\mathbf{y}_t\right)$ at the time of data association.

Consequently, the measurement probability of a single observation can be derived by comparison with equation 2.108 analogous to [1] giving again a multivariate Gaussian:

$$\mathbf{z}_{t,i}|\boldsymbol{u}_{1:t},\boldsymbol{z}_{1:t-1} \sim \mathcal{N}(\boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right), \boldsymbol{H}_{t,j}\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}\boldsymbol{H}_{t,j}^{\top} + \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}}) \tag{3.34}$$

This result should not come as a surprise, since the mean $\boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right)$ and the covariance $\boldsymbol{H}_{t,j}\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}\boldsymbol{H}_{t,j}^{\top} + \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}}$ are already used in the update step in the entire form of $\boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right)$ and $\boldsymbol{H}_t$ describing the predicted measurements and its uncertainty. Remember, in algorithm 3.4 $\boldsymbol{c}_t$ is additionally handed over to $\boldsymbol{h}\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right)$ emphasizing the correspondences (supposed at this time) between measurements and map features.

Hence, from the definition of the Mahalanobis distance in equation 3.26 and the deviation of the individual measurement's expectation and covariance in equation 3.34 one can now directly derive algorithm 3.9. This algorithm is used for the calculation of the squared

Algorithm 3.9: Mahalanobis Distance

input : $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t = (\boldsymbol{z}_{t,1} \dots \boldsymbol{z}_{t,M})^\top$, i, j
output: $d_{t,ij}$

1 $\bar{\boldsymbol{\mu}}_{\mathbf{z}_{t,i}} = \boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right)$;
2 $\boldsymbol{v}_{t,ij} = \boldsymbol{z}_{t,i} - \bar{\boldsymbol{\mu}}_{\mathbf{z}_{t,i}}$;
3 $\bar{\boldsymbol{\Sigma}}_{\mathbf{z}_{t,i}} = \boldsymbol{H}_{t,j}\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}\boldsymbol{H}_{t,j}^\top + \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}}$;
4 $d_{t,ij}^2 = \boldsymbol{v}_{t,ij}^\top \bar{\boldsymbol{\Sigma}}_{\mathbf{z}_{t,i}}^{-1} \boldsymbol{v}_{t,ij}$;

Mahalanobis distance in the algorithms 3.7 and 3.8 implementing the two mentioned NNSF approaches.

Finally, we have a closer look on the still left open γ, introduced in the algorithms 3.7 and 3.8 in line 14 and 19, respectively. As the usage already suggests, it is a threshold distinguishing acceptable correspondences from unacceptable ones. Meaning, even if an observed feature $\boldsymbol{z}_{t,i}$ and a map feature $\boldsymbol{m}_j$ are the nearest neighbors in the sense of the Mahalanobis distance it might happen that they are still too far-off for reasonable association with each other.

Let's have a closer look on the Mahalanobis distance defined in equation 3.26 in order to interpret γ appropriately. For this purpose, we assume a random vector $\mathbf{r} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{r}, \boldsymbol{\Sigma}_\mathbf{r})$ of dimension n. Then we obtain for the squared Mahalanobis distance

$$\mathsf{d}^2 = (\mathbf{r} - \boldsymbol{\mu}_\mathbf{r})^\top \boldsymbol{\Sigma}_\mathbf{r}^{-1} (\mathbf{r} - \boldsymbol{\mu}_\mathbf{r}) \sim \chi_n^2 \tag{3.35}$$

where χ_n^2 denotes a *chi-square* distribution with n degrees of freedom.

A chi-square distribution with n degrees of freedom is defined as the sum of n squared statistically independent standard normally distributed random variables s_k [14]:

$$\chi_n^2 \sim \sum_{k=1}^{n} \mathsf{s}_k^2 \text{ with } \mathsf{s}_k \sim \mathcal{N}(0,1),\ k \in [1,n] \tag{3.36}$$

Proof. In order to show that the squared Mahalanobis distance is chi-squared distributed with n degrees of freedom, we will use similar as in [16] following substitution:

$$\mathbf{s} = \boldsymbol{R}^{-1}(\mathbf{r} - \boldsymbol{\mu}_\mathbf{r})$$

The matrix $\boldsymbol{R}$ is received via a Cholesky decomposition from $\boldsymbol{\Sigma}_\mathbf{r}$:

$$\boldsymbol{\Sigma}_\mathbf{r} = \boldsymbol{R}\boldsymbol{R}^\top$$

Thus, we obtain for the random vector $\mathbf{s}$ a standard multivariate Gaussian:

$$\mathbf{s} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

This can be easily verified by equation 2.21 for the mean

$$\boldsymbol{\mu}_{\mathsf{s}} = \boldsymbol{R}^{-1}(\boldsymbol{\mu}_{\mathsf{r}} - \boldsymbol{\mu}_{\mathsf{r}}) = \mathbf{0}$$

and by equation 2.58 for the covariance:

$$\boldsymbol{\Sigma}_{\mathsf{s}} = \boldsymbol{R}^{-1}\boldsymbol{\Sigma}_{\mathsf{r}}(\boldsymbol{R}^{-1})^{\top} = \boldsymbol{R}^{-1}\boldsymbol{R}\boldsymbol{R}^{\top}(\boldsymbol{R}^{-1})^{\top} = \boldsymbol{I}$$

Finally, we can derive the desired statement by

$$\begin{aligned}
\mathsf{d}^2 &= (\mathsf{r} - \boldsymbol{\mu}_{\mathsf{r}})^{\top}\boldsymbol{\Sigma}_{\mathsf{r}}^{-1}(\mathsf{r} - \boldsymbol{\mu}_{\mathsf{r}}) \\
&= (\mathsf{r} - \boldsymbol{\mu}_{\mathsf{r}})^{\top}(\boldsymbol{R}\boldsymbol{R}^{\top})^{-1}(\mathsf{r} - \boldsymbol{\mu}_{\mathsf{r}}) \\
&= (\mathsf{r} - \boldsymbol{\mu}_{\mathsf{r}})^{\top}(\boldsymbol{R}^{-1})^{\top}\boldsymbol{R}^{-1}(\mathsf{r} - \boldsymbol{\mu}_{\mathsf{r}}) \\
&= \mathsf{s}^{\top}\mathsf{s} = \sum_{k=1}^{n} \mathsf{s}_k^2
\end{aligned}$$

with

$$\mathsf{s}_k \sim \mathcal{N}(0,1),\ k \in [1,n]$$

since $\mathsf{s} = (\mathsf{s}_1 \dots \mathsf{s}_n)^{\top} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and in [14] it was shown that the individuals are then also standard Gaussian. □

Hence, the Mahalanobis distance calculated in line 4 of algorithm 3.9 is based on a chi-square distribution with 3 degrees of freedom. There are 3 degrees of freedom because $\boldsymbol{z}_{t,i}$ describes a measured 2D pose in our case, which is a vector of dimension 3. For better imagination figure 3.6 depicts the cdf and pdf of a chi-square distribution with 3 degrees of freedom.

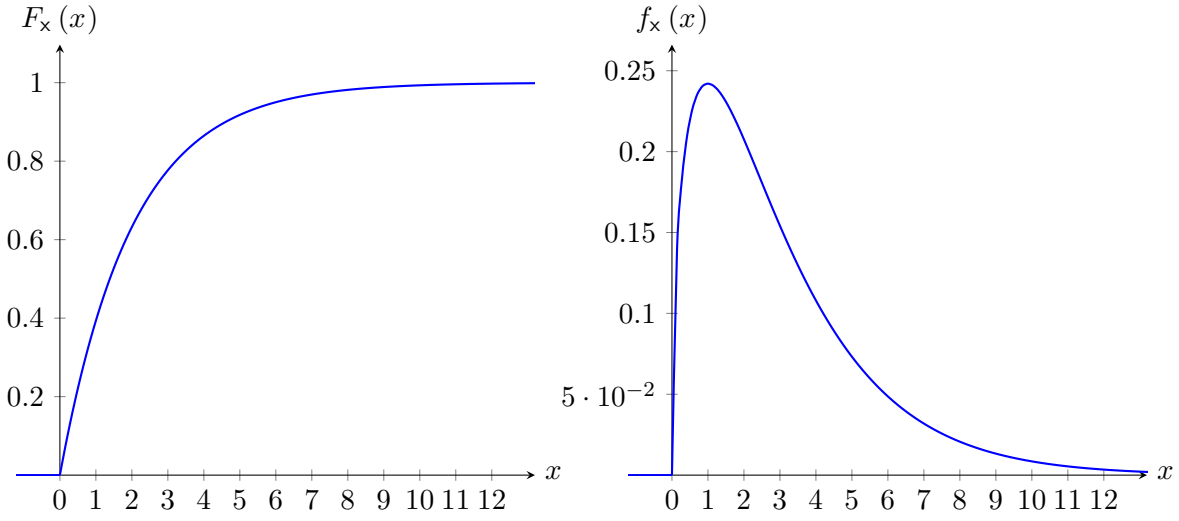
Based on this insight, we can now interpret γ as the threshold at which

$$\alpha = F_{\mathsf{d}^2}(\gamma) \tag{3.37}$$

of measurements $\boldsymbol{z}_{t,i}$ are accepted. Vice versa, γ is calculated by the inverse cdf of a chi-square distribution with 3 degrees of freedom at the level α:

$$\gamma = F_{\mathsf{d}^2}^{-1}(\alpha) \tag{3.38}$$

In practice, typical values for α are between 0.9 and 0.95. Because of the underlying Gaussian distribution 3.34 the threshold γ describes again a hyper-ellipsoid of the same dimension as the measurements $\boldsymbol{z}_{t,i}$. This is called the *validation region* or *validation gate* in [16] and is actually indicated by the ellipses in the figure 3.5b.



(a) Cumulative distribution function

(b) Probability density function

Figure 3.6: Chi-square random variable $\mathsf{x} \sim \chi^2_3$

Summed up, NNSF tries to resolve unknown correspondences as follows:

1. Predict the remaining unassociated map features $\mathbf{m}_j$ by means of $\boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right)$ obtaining expected measurements $\bar{\boldsymbol{\mu}}_{\mathbf{z}_{t,i}}$,
2. find the observed feature $\boldsymbol{z}_{t,i}$ with the shortest Mahalanobis distance and
3. check if this feature lies inside the validation region defined by the threshold γ,
4. in which case the observed feature $\boldsymbol{z}_{t,i}$ is associated with the map feature $\mathbf{m}_j$ in the updated correspondence data $\bar{\boldsymbol{c}}_t$.

The next chapter revisits the introduced Algorithms for EKF-SLAM by focusing more on implementation details and further improvements. Beside that, the used geometry and the associated transformation of different coordinate systems are touched.

CHAPTER 4

Implementation

In the previous chapter, a general solution for EKF-SLAM with visual markers was presented. In this chapter, we will revisit the introduced algorithms for EKF-SLAM and enhance them with implementation details related to our setting. This includes specific models for transition and measurement probabilities as well as integration and implementation relevant optimizations. The finally implemented algorithms are freely available as ROS package in [3].

The chapter starts with a short introduction into geometry related to mobile robotics in section 4.1. This includes transformations from one coordinate system into another, which are frequently needed in mobile robotics. After that the previous presented algorithms for prediction, data association, update and integration are revisited in sections 4.2 to 4.5. In section 4.2 the prediction algorithm is optimized and a motion model is introduced. In the context of data association a model for the measurement probability is presented in section 4.3. Section 4.4 proposes two different algorithms for the update step. In section 4.5, the integration function is specified and the related integration algorithm is optimized accordingly. Finally, section 4.6 concludes the chapter with an overview of the provided ROS package.

4.1 Geometry

This thesis covers EKF-SLAM in 2D. In 2D space, a pose is a composition of a 2D position and a single orientation angle. We have already seen in chapter 3 in figure 3.3 that a pose can be either given in *Cartesian coordinates*

$$(x\ y\ \theta)^\top \tag{4.1}$$

or in *spherical coordinates*:

$$(r\ \phi\ \theta)^\top \tag{4.2}$$

For the sake of convenience, both representations are recapitulated at this place.

Every pose corresponds to a specific coordinate system, in which it is defined. Throughout the remaining thesis, we will distinguish between following coordinate systems, which are also referred to as frames:

map This coordinate system describes the actual target coordinate system in which the robot's pose $\mathbf{x}$ and the poses of the map features $\mathbf{m}_1, \ldots, \mathbf{m}_N$ are estimated. If needed for better understanding poses in this frame are depicted with an upper M.

robot In this coordinate system the control commands $\boldsymbol{u}_t$ are applied. Typically, in SLAM applications the origin of this frame equals initially the map's origin, but not necessarily. If needed for better understanding poses in this frame are depicted with an upper R.

sensor Within this coordinate system, the measurements $\boldsymbol{z}_t$ are taken. If needed for better understanding poses in this frame are depicted with an upper S.

In our context, the sensor pose is usually given in the robot coordinate system since the sensor, i.e. the visual marker detection is attached to the robot. Figure 4.1 depicts this setting. Although it may seem unusual to have the x-axis upright, it has been proven useful to define the x-axis in the viewing direction of a vehicle in the context of orientation.

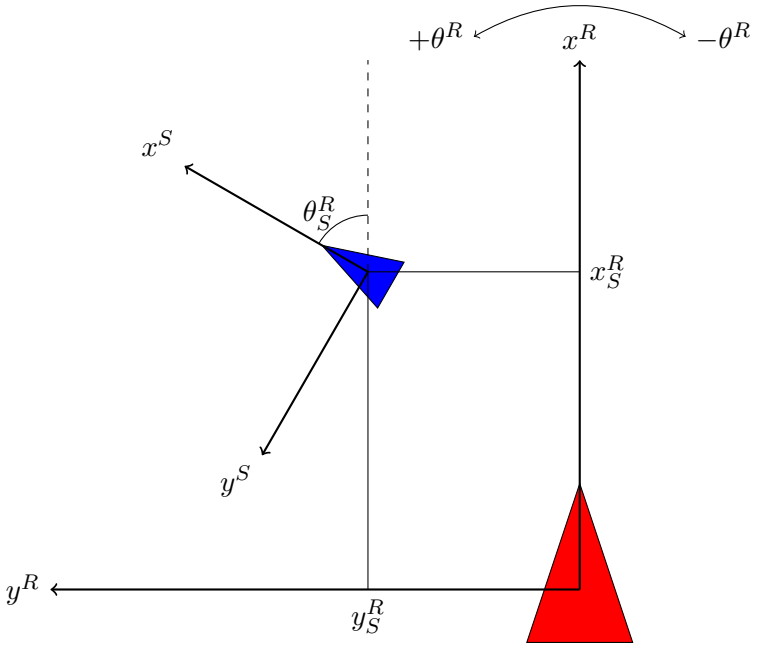


Figure 4.1: Sensor's pose $(x_S^R \; y_S^R \; \theta_S^R)^\top$ in robot coordinate system

In figure 4.2, it is illustrated how a feature, i.e. a visual marker is observed. Obviously, features are detected by the sensor. Consequently, their poses are given in the sensor coordinate system. But in contrast to the sensor pose, which was specified in Cartesian coordinates, the poses of observed features are assumed to be in spherical coordinates. As already stated in the previous chapter, the reason therefore is due to the way how measurements are taken and the noise is described.

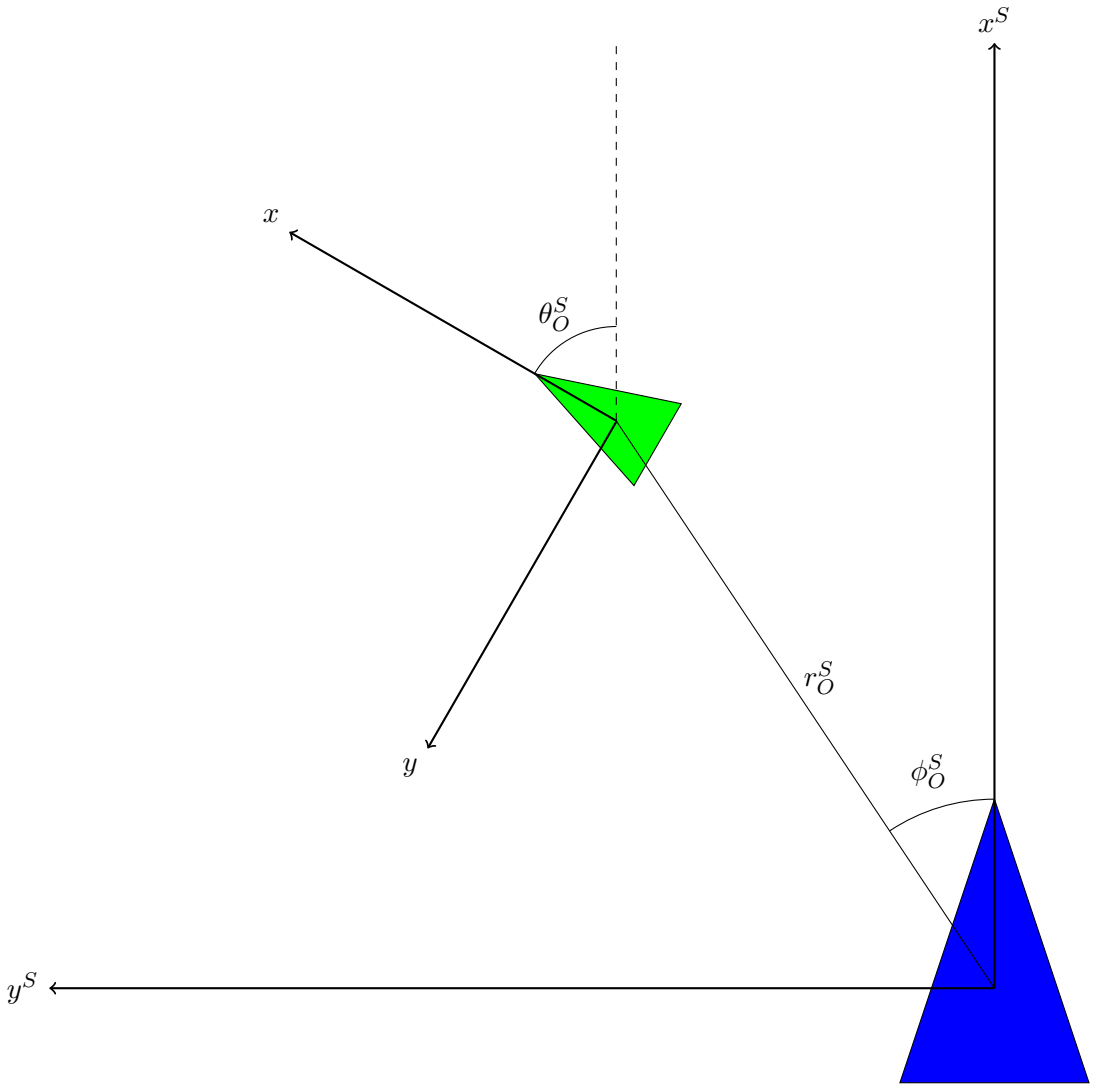


Figure 4.2: Visual marker's pose $(r_O^S\ \phi_O^S\ \theta_O^S)^\top$ in sensor coordinate system

Finally, figure 4.3 incorporates the sensor and robot poses into the map coordinate system. Additionally the dashed arrows illustrate how the different coordinate systems are interconnected. It is kind of natural to embed the robot coordinate system into the map and, thus, to specify the robot's pose in Cartesian coordinates relative to the map's origin. Although the map's origin can be defined initially arbitrarily in the context of SLAM, it is usual to define it overlapping with the robot's initially pose, i.e. $\boldsymbol{x}_0 = (0^M \ 0^M \ 0^M)^\top$.

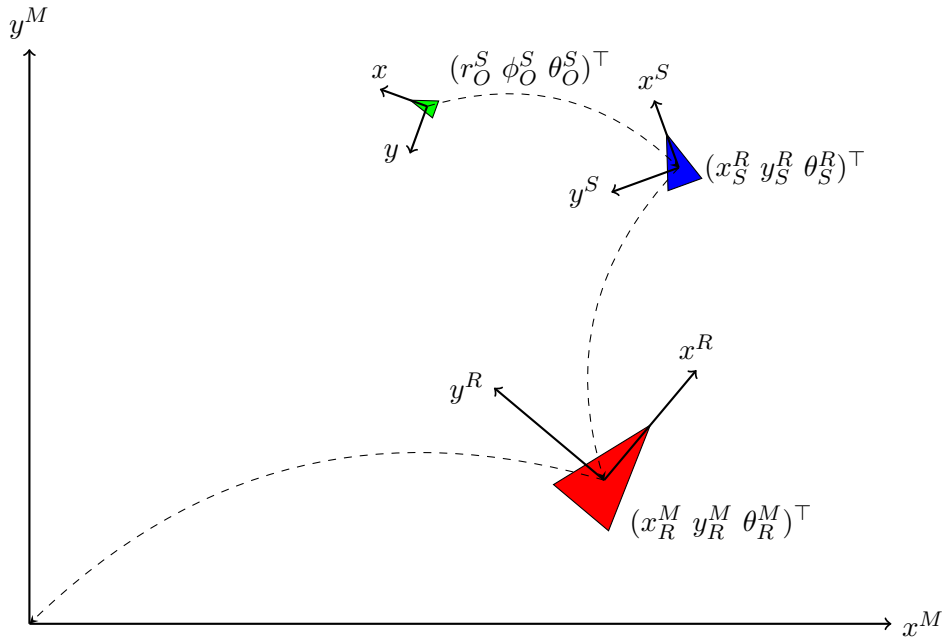


Figure 4.3: Map coordinate system

The remaining question is how to transform poses from one coordinate system into another. A typical setting considered in the topic of SLAM is illustrated in figure 4.4. A visual marker is detected by the sensor and, thus, its pose is given in sensor coordinates and needs to be transformed into the robot frame for further processing. By considering trigonometric relations, one derives for the 2D position of the observed visual marker in the robot frame

$$\begin{pmatrix} x_O^R \\ y_O^R \end{pmatrix} = \underbrace{\begin{pmatrix} x_S^R \\ y_S^R \end{pmatrix}}_{\text{Translation}} + \underbrace{\begin{pmatrix} \cos\left(\theta_S^R\right) & -\sin\left(\theta_S^R\right) \\ \sin\left(\theta_S^R\right) & \cos\left(\theta_S^R\right) \end{pmatrix}}_{\text{Rotation}} \begin{pmatrix} x_O^S \\ y_O^S \end{pmatrix} \tag{4.3}$$

with the implicit conversion of the visual marker's position in the sensor frame from spherical coordinates into Cartesian coordinates:

$$\begin{pmatrix} x_O^S \\ y_O^S \end{pmatrix} = \begin{pmatrix} r_O^S \cos\left(\phi_O^S\right) \\ r_O^S \sin\left(\phi_O^S\right) \end{pmatrix} \tag{4.4}$$

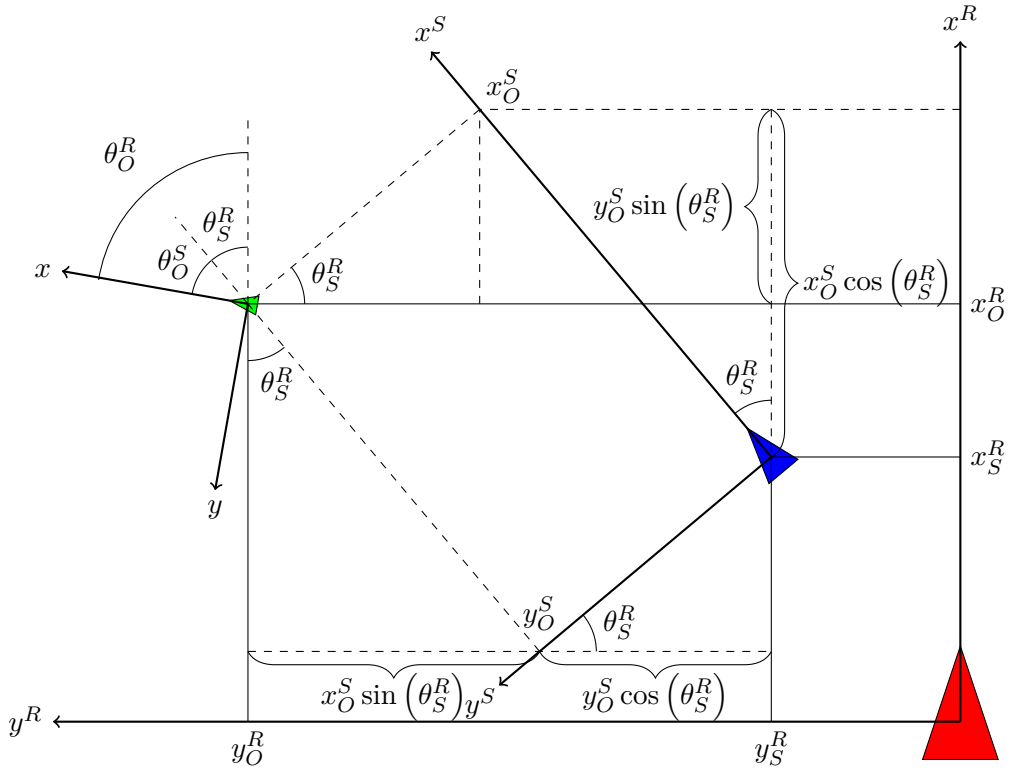


Figure 4.4: Transformation from sensor coordinate system into robot coordinate system

Thus, a transformation is in general a composition of a translation and a rotation.

In order to get a single transformation matrix including translation and rotation, often *homogeneous vectors* are used:

$$\begin{pmatrix} x_O^R \\ y_O^R \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\left(\theta_S^R\right) & -\sin\left(\theta_S^R\right) & x_S^R \\ \sin\left(\theta_S^R\right) & \cos\left(\theta_S^R\right) & y_S^R \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_O^S \\ y_O^S \\ 1 \end{pmatrix} \tag{4.5}$$

Finally, we can extend this approach to cover not only positions but also poses in 2D:

$$\underbrace{\begin{pmatrix} x_O^R \\ y_O^R \\ \theta_O^R \\ 1 \end{pmatrix}}_{\underline{\boldsymbol{x}}_O^R} = \underbrace{\begin{pmatrix} \cos\left(\theta_S^R\right) & -\sin\left(\theta_S^R\right) & 0 & x_S^R \\ \sin\left(\theta_S^R\right) & \cos\left(\theta_S^R\right) & 0 & y_S^R \\ 0 & 0 & 1 & \theta_S^R \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{{}^R\boldsymbol{T}^S} \underbrace{\begin{pmatrix} x_O^S \\ y_O^S \\ \theta_O^S \\ 1 \end{pmatrix}}_{\underline{\boldsymbol{x}}_O^S} \tag{4.6}$$

Notice: Special attention needs to be given to ensure $\theta_S^R + \theta_O^S \in (-\pi, \pi)$.

The underline in $\underline{\boldsymbol{x}}_O^R$ emphasizes the fact that we are dealing with homogeneous poses. Furthermore, the matrix ${}^R\boldsymbol{T}^S$ can be read as a transformation from the sensor frame S into the robot frame R. Sometimes it is intentionally parametrized, i.e. ${}^R\boldsymbol{T}^S_{\boldsymbol{x}_S^R}$, distinguishing the underlying source of transformation.

An important property of the transformation matrix can be immediately derived by equation 4.6:

$$ {}^S\boldsymbol{T}^R = \left({}^R\boldsymbol{T}^S\right)^{-1} \tag{4.7}$$

This allows the change of the coordinate system in both directions with just one transformation matrix given.

By means of this introduction into geometry used in mobile robotics, we can now proceed with the particular steps in the cycle of EKF-SLAM for implementation.

4.2 Prediction

In algorithm 3.2 for the prediction step in the EKF-SLAM cycle the function $\bar{\boldsymbol{g}}\left(\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{u}_t\right)$ in line 1 describes the transition probability. By definition, in static environments as considered for this thesis, the map does not change during the vehicle's movement. Furthermore, it is legitimate to assume in feature-based SLAM that observed features have no impact on the current control commands. This is because the features serve only for localization but do not indicate obstacles in general. Thus, $\bar{\boldsymbol{g}}\left(\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{u}_t\right)$ can be in our case rewritten as

$$ \bar{\boldsymbol{\mu}}_{\mathbf{y}_t} = \begin{pmatrix} \bar{\boldsymbol{\mu}}_{\mathbf{x}_t} \\ \bar{\boldsymbol{\mu}}_{\mathbf{m}_1} \\ \vdots \\ \bar{\boldsymbol{\mu}}_{\mathbf{m}_N} \end{pmatrix} = \bar{\boldsymbol{g}}\left(\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{u}_t\right) = \begin{pmatrix} \boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right) \\ \boldsymbol{\mu}_{\mathbf{m}_1} \\ \vdots \\ \boldsymbol{\mu}_{\mathbf{m}_N} \end{pmatrix} \tag{4.8}$$

with function $\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)$ as introduced in equation 2.95 analogously to [17] and [18].

Furthermore, $\bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}}$ and $\bar{\boldsymbol{G}}_{\boldsymbol{u}_t}$ in line 2 of algorithm 3.2 can be derived analogously to equation 2.98, yielding under the given assumptions

$$ \bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}} = \frac{d\bar{\boldsymbol{g}}\left(\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{u}_t\right)}{d\boldsymbol{y}_{t-1}} = \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{x}_{t-1}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{I} \end{pmatrix} \tag{4.9}$$

and

$$ \bar{\boldsymbol{G}}_{\boldsymbol{u}_t} = \frac{d\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{y}_{t-1}}, \boldsymbol{u}_t\right)}{d\bar{\boldsymbol{u}}_t} = \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{u}_t} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \tag{4.10}$$

respectively, with $\boldsymbol{G}_{\boldsymbol{x}_{t-1}}$ and $\boldsymbol{G}_{\boldsymbol{u}_t}$ as introduced in the equation 2.98.

Thus, by means of equations 3.10, 4.9 and 4.10 the composition of $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$ can be examined by

$$\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t} = \begin{pmatrix} \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} & \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t\mathbf{m}_1} & \cdots & \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t\mathbf{m}_N} \\ \bar{\boldsymbol{\Sigma}}_{\mathbf{m}_1\mathbf{x}_t} & \boldsymbol{\Sigma}_{\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{m}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\boldsymbol{\Sigma}}_{\mathbf{m}_N\mathbf{x}_t} & \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_N} \end{pmatrix} \tag{4.11}$$

with

$$\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} = \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top + \boldsymbol{G}_{\boldsymbol{u}_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{G}_{\boldsymbol{u}_t}^\top \tag{4.12}$$

and

$$\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t\mathbf{m}_j} = \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_j}, \; j \in [1, N] \tag{4.13}$$

whereas

$$\bar{\boldsymbol{\Sigma}}_{\mathbf{m}_j\mathbf{x}_t} = \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t\mathbf{m}_j}^\top \tag{4.14}$$

because of the covariance symmetry:

$$\boldsymbol{\Sigma}_{\mathbf{m}_j\mathbf{x}_{t-1}} = \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_j}^\top \tag{4.15}$$

Proof. In order to show equations 4.11, 4.12 and 4.13, line 2 of algorithm 3.2 is first split into its two additive terms $\bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{y}_{t-1}} \bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}}^\top$ and $\bar{\boldsymbol{G}}_{\boldsymbol{u}_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \bar{\boldsymbol{G}}_{\boldsymbol{u}_t}^\top$. After inspecting these two terms separately their results are combined to obtain the desired statements. The proof itself was inspired by [17] and [18].

The first term denotes a big matrix multiplication. The affected columns and rows are highlighted for better readability during evaluation:

$$\begin{aligned}
&\bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{y}_{t-1}} \bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}}^\top = \\
&= \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{x}_{t-1}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} & \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_N} \\ \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{x}_{t-1}} & \boldsymbol{\Sigma}_{\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{m}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{x}_{t-1}} & \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_N} \end{pmatrix} \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{I} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{x}_{t-1}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_N} \\ \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{\Sigma}_{\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{m}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_N} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_1} & \cdots & \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_N} \\ \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{\Sigma}_{\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{m}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_N} \end{pmatrix}
\end{aligned}$$

The second additive is also simply expanded, yielding a sparse matrix of size $3(N+1)$ with a single entry:

$$\begin{aligned}
\bar{\boldsymbol{G}}_{\boldsymbol{u}_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \bar{\boldsymbol{G}}_{\boldsymbol{u}_t}^\top &= \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{u}_t} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \boldsymbol{\Sigma}_{\mathbf{c}_t} \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{u}_t}^\top & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{u}_t} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{G}_{\boldsymbol{u}_t}^\top & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{u}_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{G}_{\boldsymbol{u}_t}^\top & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix}
\end{aligned}$$

Now both terms are added together in order to achieve the desired results. Again, for better readability the affected columns and rows are highlighted:

$$\begin{aligned}
\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t} &= \bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{y}_{t-1}} \bar{\boldsymbol{G}}_{\boldsymbol{y}_{t-1}}^\top + \bar{\boldsymbol{G}}_{\boldsymbol{u}_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \bar{\boldsymbol{G}}_{\boldsymbol{u}_t}^\top \\
&= \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_1} & \cdots & \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_N} \\ \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{\Sigma}_{\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{m}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_N} \end{pmatrix} \\
&+ \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{u}_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{G}_{\boldsymbol{u}_t}^\top & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top + \boldsymbol{G}_{\boldsymbol{u}_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{G}_{\boldsymbol{u}_t}^\top & \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_1} & \cdots & \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_N} \\ \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{\Sigma}_{\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{m}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^\top & \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_N} \end{pmatrix} \\
&= \begin{pmatrix} \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} & \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t\mathbf{m}_1} & \cdots & \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t\mathbf{m}_N} \\ \bar{\boldsymbol{\Sigma}}_{\mathbf{m}_1\mathbf{x}_t} & \boldsymbol{\Sigma}_{\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{m}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\boldsymbol{\Sigma}}_{\mathbf{m}_N\mathbf{x}_t} & \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_N} \end{pmatrix}
\end{aligned}$$

□

Equation 4.11 emphasizes the fact that during prediction only the uncertainty of the vehicle's pose and the uncertainty of the map associated with it is affected, but not the map by itself. This knowledge can be exploited in order to achieve a more efficient implementation of the prediction. Algorithm 4.1 illustrates this by reducing the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. This is due to the remaining matrices in algorithm 4.1, which are independent of the feature number N of the map and thus of constant size, i.e. in our case $\boldsymbol{G}_{\boldsymbol{x}_{t-1}}$, $\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}$ and $\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_j}$ are of size 3×3 and the matrices $\boldsymbol{G}_{\boldsymbol{u}_t}$ and $\boldsymbol{\Sigma}_{\mathbf{c}_t}$ are of size 3×2 and 2×2, respectively. Consequently, matrix multiplication yields only constant computation costs. The determining factor here is the for-loop covering lines 3 to 5.

Algorithm 4.1: Prediction (implemented)

input : $\boldsymbol{\mu}_{\mathbf{y}_{t-1}}$, $\boldsymbol{\Sigma}_{\mathbf{y}_{t-1}}$, $\boldsymbol{u}_t$
output: $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$

1 $\bar{\boldsymbol{\mu}}_{\mathbf{x}_t} = \boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)$;
2 $\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t} = \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}} \boldsymbol{G}_{\boldsymbol{x}_{t-1}}^{\top} + \boldsymbol{G}_{\boldsymbol{u}_t} \boldsymbol{\Sigma}_{\mathbf{c}_t} \boldsymbol{G}_{\boldsymbol{u}_t}^{\top}$;
3 **for** $j = 1$ **to** N **do**
4 $\quad \bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t\mathbf{m}_j} = \boldsymbol{G}_{\boldsymbol{x}_{t-1}} \boldsymbol{\Sigma}_{\mathbf{x}_{t-1}\mathbf{m}_j}$;
5 **end**

Notice: Algorithm 4.1 implicitly assumes sub-matrix access in order to establish $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$ from $\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t}$ and $\bar{\boldsymbol{\Sigma}}_{\mathbf{x}_t\mathbf{m}_j}$ as depicted in equations 4.8 and 4.11. This assumption is also exploited in further algorithms.

For function $\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)$ in line 1 of algorithm 4.1 the *motion model*

$$\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right) = \boldsymbol{\mu}_{\mathbf{x}_{t-1}} + \begin{pmatrix} -\frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega} \sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) + \frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega} \sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta + \boldsymbol{u}_t.\omega\Delta t\right) \\ \frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega} \cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) - \frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega} \cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta + \boldsymbol{u}_t.\omega\Delta t\right) \\ \boldsymbol{u}_t.\omega\Delta t \end{pmatrix} \tag{4.16}$$

introduced in [1] is used, where Δt indicates the time between two consecutive cycles of EKF-SLAM. Consequently, by equation 2.98 we derive

$$\boldsymbol{G}_{\boldsymbol{x}_{t-1}} = \frac{d\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)}{d\boldsymbol{x}_{t-1}} = \begin{pmatrix} 1 & 0 & -\frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega} \cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) + \frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega} \cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta + \boldsymbol{u}_t.\omega\Delta t\right) \\ 0 & 1 & -\frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega} \sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) + \frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega} \sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta + \boldsymbol{u}_t.\omega\Delta t\right) \\ 0 & 0 & 1 \end{pmatrix} \tag{4.17}$$

and

$$\boldsymbol{G}_{\boldsymbol{u}_t} = \frac{d\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)}{d\bar{\mathbf{u}}_t} = \begin{pmatrix} \frac{-\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)+\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta+\boldsymbol{u}_t.\omega\Delta t\right)}{\boldsymbol{u}_t.\omega} \\ \frac{\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)-\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta+\boldsymbol{u}_t.\omega\Delta t\right)}{\boldsymbol{u}_t.\omega} \\ 0 \end{pmatrix}$$
$$\begin{pmatrix} \frac{\boldsymbol{u}_t.v\left(\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)-\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta+\boldsymbol{u}_t.\omega\Delta t\right)\right)}{\boldsymbol{u}_t.\omega^2} + \frac{\boldsymbol{u}_t.v\Delta t\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta+\boldsymbol{u}_t.\omega\Delta t\right)}{\boldsymbol{u}_t.\omega} \\ -\frac{\boldsymbol{u}_t.v\left(\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)-\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta+\boldsymbol{u}_t.\omega\Delta t\right)\right)}{\boldsymbol{u}_t.\omega^2} + \frac{\boldsymbol{u}_t.v\Delta t\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta+\boldsymbol{u}_t.\omega\Delta t\right)}{\boldsymbol{u}_t.\omega} \\ \Delta t \end{pmatrix} \tag{4.18}$$

for $\boldsymbol{G}_t = \left(\boldsymbol{G}_{\boldsymbol{x}_{t-1}}\ \boldsymbol{G}_{\boldsymbol{u}_t}\right)$ denoting the Jacobian matrix of $\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)$.

In the case of $\boldsymbol{u}_t.\omega = 0$, the motion model is simplified to

$$\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right) = \boldsymbol{\mu}_{\mathbf{x}_{t-1}} + \begin{pmatrix} \boldsymbol{u}_t.v\Delta t\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) \\ \boldsymbol{u}_t.v\Delta t\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) \\ 0 \end{pmatrix} \tag{4.19}$$

describing a straight line movement.

Proof. In order to show equation 4.19, one can either consider trigonometric relations or simply take the limit of equation 4.16:

$$\begin{aligned}
\bar{\boldsymbol{\mu}}_{\mathbf{x}_{t-1}}.x &= \lim_{\boldsymbol{u}_t.\omega\to 0} \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.x - \frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega}\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) + \frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega}\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta + \boldsymbol{u}_t.\omega\Delta t\right) \\
&= \lim_{\boldsymbol{u}_t.\omega\to 0} \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.x + 2\frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega}\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta + \frac{\boldsymbol{u}_t.\omega\Delta t}{2}\right)\sin\left(\frac{\boldsymbol{u}_t.\omega\Delta t}{2}\right) \\
&= \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.x + 2\boldsymbol{u}_t.v\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)\lim_{\boldsymbol{u}_t.\omega\to 0}\frac{\sin\left(\frac{\boldsymbol{u}_t.\omega\Delta t}{2}\right)}{\boldsymbol{u}_t.\omega} \\
&= \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.x + 2\boldsymbol{u}_t.v\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)\lim_{\boldsymbol{u}_t.\omega\to 0}\frac{\frac{\Delta t}{2}\cos\left(\frac{\boldsymbol{u}_t.\omega\Delta t}{2}\right)}{1} \\
&= \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.x + \boldsymbol{u}_t.v\Delta t\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)
\end{aligned}$$

$$\begin{aligned}
\bar{\boldsymbol{\mu}}_{\mathbf{x}_{t-1}}.y &= \lim_{\boldsymbol{u}_t.\omega\to 0} \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.y + \frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega}\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) - \frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega}\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta + \boldsymbol{u}_t.\omega\Delta t\right) \\
&= \lim_{\boldsymbol{u}_t.\omega\to 0} \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.y + 2\frac{\boldsymbol{u}_t.v}{\boldsymbol{u}_t.\omega}\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta + \frac{\boldsymbol{u}_t.\omega\Delta t}{2}\right)\sin\left(\frac{\boldsymbol{u}_t.\omega\Delta t}{2}\right) \\
&= \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.y + 2\boldsymbol{u}_t.v\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)\lim_{\boldsymbol{u}_t.\omega\to 0}\frac{\sin\left(\frac{\boldsymbol{u}_t.\omega\Delta t}{2}\right)}{\boldsymbol{u}_t.\omega} \\
&= \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.y + 2\boldsymbol{u}_t.v\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)\lim_{\boldsymbol{u}_t.\omega\to 0}\frac{\frac{\Delta t}{2}\cos\left(\frac{\boldsymbol{u}_t.\omega\Delta t}{2}\right)}{1} \\
&= \boldsymbol{\mu}_{\mathbf{x}_{t-1}}.y + \boldsymbol{u}_t.v\Delta t\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right)
\end{aligned}$$

Notice: Both derivations use in their first line the trigonometric relations

$$\sin(\alpha) - \sin(\beta) = 2\cos\left(\frac{\alpha+\beta}{2}\right)\sin\left(\frac{\alpha-\beta}{2}\right)$$

and

$$\cos(\alpha) - \cos(\beta) = -2\sin\left(\frac{\alpha+\beta}{2}\right)\sin\left(\frac{\alpha-\beta}{2}\right)$$

respectively. Furthermore, L'Hospital's rule is both times applied in line 3. □

Similar as we did for $\boldsymbol{g}\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}, \boldsymbol{u}_t\right)$, one can derive

$$\boldsymbol{G}_{\boldsymbol{x}_{t-1}} = \begin{pmatrix} 1 & 0 & -\boldsymbol{u}_t.v\Delta t\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) \\ 0 & 1 & \boldsymbol{u}_t.v\Delta t\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) \\ 0 & 0 & 1 \end{pmatrix} \tag{4.20}$$

and

$$\boldsymbol{G}_{\boldsymbol{u}_t} = \begin{pmatrix} \Delta t\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) & -\frac{1}{2}\Delta t^2\boldsymbol{u}_t.v\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) \\ \Delta t\sin\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) & \frac{1}{2}\Delta t^2\boldsymbol{u}_t.v\cos\left(\boldsymbol{\mu}_{\mathbf{x}_{t-1}}.\theta\right) \\ 0 & \Delta t \end{pmatrix} \tag{4.21}$$

for the Jacobian matrix $\boldsymbol{G}_t = \left(\boldsymbol{G}_{\boldsymbol{x}_{t-1}}\ \boldsymbol{G}_{\boldsymbol{u}_t}\right)$ with $\omega \to 0$.

Finally, let us have a look at the control noise expressed by $\boldsymbol{\Sigma}_{\mathbf{c}_t}$ and applied in line 2 of algorithm 4.1. For our purpose, we are modeling the control noise by

$$\boldsymbol{\Sigma}_{\mathbf{c}_t} = \begin{pmatrix} \alpha_1\boldsymbol{u}_t.v^2 + \alpha_2\boldsymbol{u}_t.\omega^2 & 0 \\ 0 & \alpha_3\boldsymbol{u}_t.v^2 + \alpha_4\boldsymbol{u}_t.\omega^2 \end{pmatrix} \tag{4.22}$$

with the reconfigurable parameters $\alpha_i,\ i \in [1,4]$ as introduced in [1].

Integrating these results into algorithm 4.1 appropriately yields the prediction part of the EKF-SLAM cycle implemented in the ROS package related to this thesis.

4.3 Data Association

The actual algorithm for the data association as well as the algorithms for the different NNSF approaches introduced in chapter 3 stay the same for the implementation. But algorithm 3.9 calculating the Mahalanobis distance is slightly adapted in order to support a central approach for the measurement model.

Algorithm 4.2 depicts this adaption. In particular, line 1 introduces a method implementing the measurement model. This method returns the Jacobian $\boldsymbol{H}_{t,j}$ of the measurement probability $\boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right)$ and $\boldsymbol{v}_{t,ij}$ denoting the difference $\boldsymbol{z}_{t,i} - \bar{\boldsymbol{\mu}}_{\mathbf{z}_{t,i}}$ as already introduced in the original algorithm 3.9 for the Mahalanobis distance. The remaining algorithm stays the same.

Algorithm 4.2: Mahalanobis Distance (implemented)

input : $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t$, i, j
output : $d_{t,ij}$

1 $(\boldsymbol{v}_{t,ij}, \boldsymbol{H}_{t,j}) =$ `Measurement` $(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \boldsymbol{z}_t, i, j)$;
2 $\boldsymbol{\Sigma}_{\mathbf{z}_{t,i}} = \boldsymbol{H}_{t,j} \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t} \boldsymbol{H}_{t,j}^{\top} + \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}}$;
3 $d_{t,ij} = \boldsymbol{v}_{t,ij}^{\top} \boldsymbol{\Sigma}_{\mathbf{z}_{t,i}}^{-1} \boldsymbol{v}_{t,ij}$;

Let us now have a closer look at the measurement probability described by $\boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right)$ introduced in equation 3.31 in chapter 3. As already mentioned, this function predicts the map feature $\mathbf{m}_j$ retained in the map frame M into the sensor frame S in order to compare it with an observed feature $\boldsymbol{z}_{t,i}$, which in turn is expressed by $\boldsymbol{v}_{t,ij}$.

Since measurements are obtained in spherical coordinates, an easy way in doing this is to express in a first step the current sensor pose $\boldsymbol{x}_S$ in Cartesian coordinates of the map frame M. Because the sensor pose is usually stated in the robot frame, this corresponds to a transformation from the robot frame R into map frame M determined by the currently expected vehicle's pose $\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}$:

$$\boldsymbol{h}_{j,1}\left(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\right) = {}^{M}\boldsymbol{T}^{R}_{\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}} \cdot \underline{\boldsymbol{x}}^{R}_{S} = \underline{\boldsymbol{x}}^{M}_{S} \tag{4.23}$$

This transformation yields the homogeneous sensor pose $\underline{\boldsymbol{x}}^{M}_{S}$ in the map frame. By means of figure 4.5 and simple trigonometry, we can now deduce a prediction of the map feature $\mathbf{m}_j$ in spherical coordinates in respect to the sensor frame in a next step:

$$\boldsymbol{h}_{j,2}\left(\underline{\boldsymbol{x}}^{M}_{S}, \bar{\boldsymbol{\mu}}_{\mathbf{m}_j}\right) = \begin{pmatrix} r_j^S \\ \phi_j^S \\ \theta_j^S \end{pmatrix} = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}\left(\delta_y, \delta_x\right) - \underline{\boldsymbol{x}}^{M}_{S}.\theta \\ \bar{\boldsymbol{\mu}}_{\mathbf{m}_j}.\theta - \underline{\boldsymbol{x}}^{M}_{S}.\theta \end{pmatrix} \tag{4.24}$$

Referring to [1], we have

$$q = \boldsymbol{\delta}^{\top}\boldsymbol{\delta} \tag{4.25}$$

with

$$\boldsymbol{\delta} = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\boldsymbol{\mu}}_{\mathbf{m}_j}.x - \underline{\boldsymbol{x}}^{M}_{S}.x \\ \bar{\boldsymbol{\mu}}_{\mathbf{m}_j}.y - \underline{\boldsymbol{x}}^{M}_{S}.y \end{pmatrix} \tag{4.26}$$

but instead of the robot pose the sensor pose in the map coordinate system is used. Besides, the common extended version

$$\operatorname{atan2}\left(y, x\right) = \begin{cases} \operatorname{atan}\left(\frac{y}{x}\right) & x > 0 \\ \operatorname{atan}\left(\frac{y}{x}\right) + \pi & x < 0, y \geq 0 \\ \operatorname{atan}\left(\frac{y}{x}\right) - \pi & x < 0, y < 0 \\ \frac{\pi}{2} & x = 0, y > 0 \\ -\frac{\pi}{2} & x = 0, y < 0 \\ 0 & x = 0, y = 0 \end{cases} \tag{4.27}$$

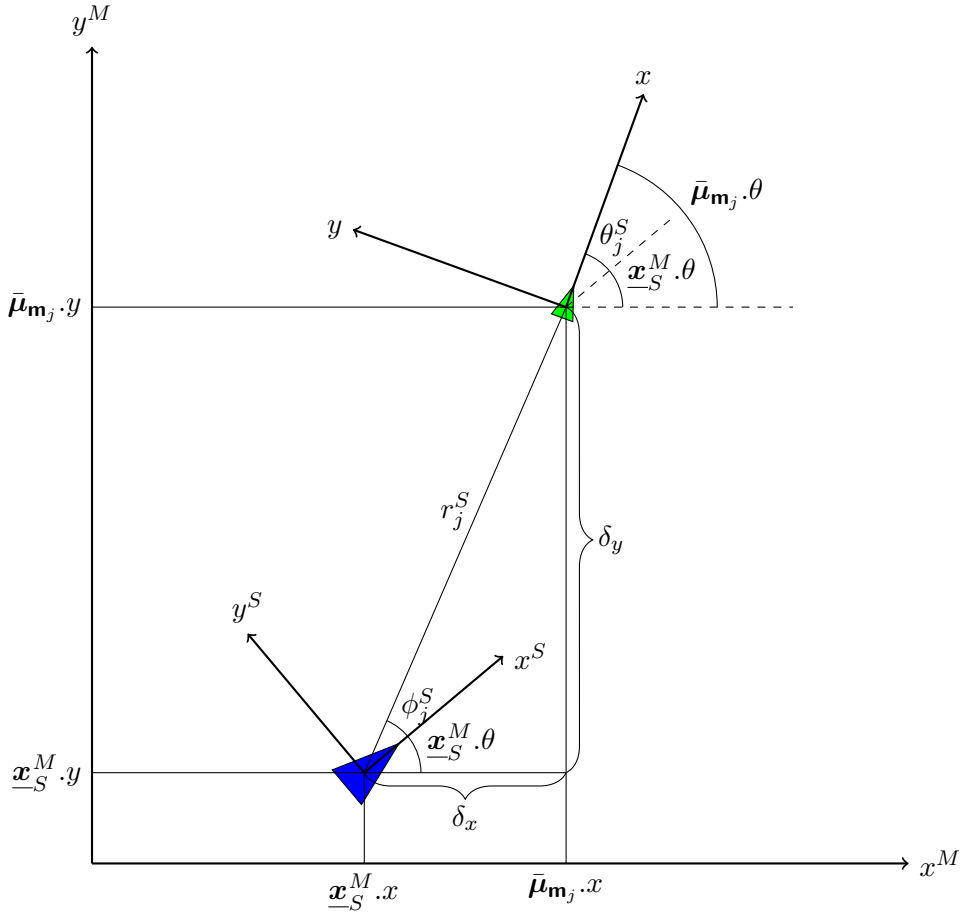


Figure 4.5: Measurement model

of the *inverse tangent* is used for the conversion from Cartesian coordinates into spherical coordinates. This function is supported by most modern programming languages and in contrast to the original inverse tangent the resulting angles are directly assigned to the right quadrants.

Thus, the composition of these two steps gives us the function describing the measurement probability:

$$\boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right) = \boldsymbol{h}_{j,2}\left(\boldsymbol{h}_{j,1}\left(\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\right), \bar{\boldsymbol{\mu}}_{\mathbf{m}_j}\right) \tag{4.28}$$

Furthermore, the Jacobian matrix of this function can be indicated by

$$\boldsymbol{H}_{t,j} = \begin{pmatrix} -\frac{\delta_x}{\sqrt{q}} & -\frac{\delta_y}{\sqrt{q}} & \frac{\delta_x\delta'_x+\delta_y\delta'_y}{\sqrt{q}} & \underbrace{0 \cdots 0}_{3(j-1)} & \frac{\delta_x}{\sqrt{q}} & \frac{\delta_y}{\sqrt{q}} & 0 & \underbrace{0 \cdots 0}_{3(N-j)} \\ \frac{\delta_y}{q} & -\frac{\delta_x}{q} & \frac{\delta'_y\delta_x-\delta_y\delta'_x}{q} - 1 & 0 \cdots 0 & -\frac{\delta_y}{q} & \frac{\delta_x}{q} & 0 & 0 \cdots 0 \\ 0 & 0 & -1 & 0 \cdots 0 & 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix} \tag{4.29}$$

denoting a sparse matrix of size $3 \times (N+1)$ with

$$\delta'_x = \frac{\partial \delta_x}{\partial x_t.\theta} \tag{4.30}$$

and

$$\delta'_y = \frac{\partial \delta_y}{\partial x_t.\theta} \tag{4.31}$$

describing first partial derivatives.

Notice: The Jacobian matrix in equation 4.29 slightly differs from the one introduced in [1], because the sensor pose is used instead of the robot pose in equation 4.26.

From this we can directly derive algorithm 4.3 outlining the central approach of the measurement model used in the implementation corresponding to this thesis. Obviously, the algorithm can be accomplished requiring only constant complexity.

Algorithm 4.3: Measurement

input : $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t} = (\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}\ \bar{\boldsymbol{\mu}}_{\mathbf{m}_1} \dots \bar{\boldsymbol{\mu}}_{\mathbf{m}_N})^\top$, $\boldsymbol{z}_t = (\boldsymbol{z}_{t,1} \dots \boldsymbol{z}_{t,M})^\top$, i, j

output: $\boldsymbol{v}_{t,ij}$, $\boldsymbol{H}_{t,j}$

1 $\underline{\boldsymbol{x}}_S^M = {}^M\boldsymbol{T}_{\bar{\boldsymbol{\mu}}_{\mathbf{x}_t}}^R \cdot \underline{\boldsymbol{x}}_S^R$;

2 $\boldsymbol{\delta} = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\boldsymbol{\mu}}_{\mathbf{m}_j}.x - \underline{\boldsymbol{x}}_S^M.x \\ \bar{\boldsymbol{\mu}}_{\mathbf{m}_j}.y - \underline{\boldsymbol{x}}_S^M.y \end{pmatrix}$;

3 $q = \boldsymbol{\delta}^\top \boldsymbol{\delta}$;

4 $\hat{\boldsymbol{z}}_{t,i} = \boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right) = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}\left(\delta_y, \delta_x\right) - \underline{\boldsymbol{x}}_S^M.\theta \\ \bar{\boldsymbol{\mu}}_{\mathbf{m}_j}.\theta - \underline{\boldsymbol{x}}_S^M.\theta \end{pmatrix}$;

5 $\boldsymbol{v}_{t,ij} = \boldsymbol{z}_{t,i} - \hat{\boldsymbol{z}}_{t,i}$;

6 $\boldsymbol{H}_{t,j} = \begin{pmatrix} -\frac{\delta_x}{\sqrt{q}} & -\frac{\delta_y}{\sqrt{q}} & \frac{\delta_x\delta'_x + \delta_y\delta'_y}{\sqrt{q}} & 0 & \cdots & 0 & \frac{\delta_x}{\sqrt{q}} & \frac{\delta_y}{\sqrt{q}} & 0 & 0 & \cdots & 0 \\ \frac{\delta_y}{q} & -\frac{\delta_x}{q} & \frac{\delta'_y\delta_x - \delta_y\delta'_x}{q} - 1 & 0 & \cdots & 0 & -\frac{\delta_y}{q} & \frac{\delta_x}{q} & 0 & 0 & \cdots & 0 \\ 0 & 0 & -1 & 0 & \cdots & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$;

Notice: In order to ensure proper angle differences, in line 5 of algorithm 4.3, the following relation is exploited in the implementation:

$$\alpha - \beta \mathrel{\widehat{=}} \operatorname{atan2}\left(\sin\left(\alpha - \beta\right), \cos\left(\alpha - \beta\right)\right), \ \alpha, \beta \in (-\pi, \pi) \tag{4.32}$$

4.4 Update

For the update step in the EKF-SLAM cycle the implementation provides two different approaches. Both of them are based on algorithm 3.4 and exploit the independent measurements assumption introduced in equation 3.28, when investigating the Mahalanobis distance. Furthermore, they make use of the central measurement model approach presented in the previous section.

The first implementation of the update step is depicted in algorithm 4.4 and is based on the solution derived in [1]. The for-loop covering lines 3 to 10 incorporates each observation $\boldsymbol{z}_{t,i}$ into the current believed state on its own. That is, why the algorithm is labeled *single*. In doing so, the independent measurements assumption is implicitly utilized. Line 4 ensures that only observed features belonging to an already known map feature $\mathbf{m}_j$ are further processed. In line 4 the measurement model is applied, calculating the deviation $\boldsymbol{v}_{t,ij}$ of the observation $\boldsymbol{z}_{t,i}$ and the measurement prediction $\hat{\boldsymbol{z}}_{t,i}$ of the associated map feature $\mathbf{m}_j$. Furthermore, the corresponding Jacobian matrix $\boldsymbol{H}_{t,j}$ is returned. Lines 7 to 9 depict the already familiar correction step of Gaussians filters. The covariance matrix $\boldsymbol{\Sigma}_{\mathbf{d}_{t,i}}$ in line 7 represents the measurement noise of a single observation, which will be examined more precisely in chapter 5. To anticipate at this point, it is a matrix of size 3×3, since the measurements $\boldsymbol{z}_{t,i}$ are of dimension 3.

Algorithm 4.4: Update (single)

input : $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t$, $\boldsymbol{c}_t = (c_1 \dots c_M)^\top$
output: $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}$, $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$

1 $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t} = \bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$;
2 $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} = \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$;
3 **for** $i = 1$ **to** M **do**
4 **if** $c_i \notin \mathbb{Z}$ **then continue**;// no associated map feature
5 $j = c_i$;
6 $(\boldsymbol{v}_{t,ij}, \boldsymbol{H}_{t,j}) =$ Measurement $(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}, \boldsymbol{z}_t, i, j)$;
7 $\boldsymbol{K}_{t,j} = \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} \boldsymbol{H}_{t,j}^\top (\boldsymbol{H}_{t,j} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} \boldsymbol{H}_{t,j}^\top + \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}})^{-1}$;
8 $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t} = \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t} + \boldsymbol{K}_{t,j} \boldsymbol{v}_{t,ij}$;
9 $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} = (\boldsymbol{I} - \boldsymbol{K}_{t,j} \boldsymbol{H}_{t,j}) \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$;
10 **end**

The computational complexity of algorithm 4.4 is determined on the one hand by the surrounding for-loop and on the other hand once again by the matrix operations inside. In particular, the matrix multiplications are crucial, because the matrix inversion in line 7 treats only a matrix of constant size 3×3. Consequently, we can denote the complexity by $\mathcal{O}(M \cdot N^2)$.

The second implementation of the update step - depicted in algorithm 4.5 - avoids the for-loop to cover matrix multiplications. Instead the vector $\boldsymbol{v}_t$ and the matrices $\boldsymbol{H}_t$

and $\boldsymbol{\Sigma}_{\mathbf{d}_t}$ are composed in advance in the for-loop covering lines 4 to 11 according to equations 3.31 and 3.32. This enables us to do the update at once as suggested in the original algorithm 3.4 for the update step. Thus, algorithm 4.5 is labeled *combined*. Again, the independent measurements assumption introduced in equation 3.28 is implicitly exploited. Similar to line 4 in algorithm 4.4, line 5 ensures that only observations associated with map features are further processed at this point. As soon as all relevant measurements are covered, they are incorporated into the current believed state at once in lines 12 to 19.

Algorithm 4.5: Update (combined)

input : $\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t$, $\boldsymbol{c}_t = (c_1 \dots c_M)^\top$
output: $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}$, $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$

1 $\boldsymbol{v}_t = \emptyset$;
2 $\boldsymbol{H}_t = \emptyset$;
3 $\boldsymbol{\Sigma}_{\mathbf{d}_t} = \emptyset$;
4 **for** $i = 1$ **to** M **do**
5 | **if** $c_i \notin \mathbb{Z}$ **then continue**;`// no associated map feature`
6 | $j = c_i$;
7 | $(\boldsymbol{v}_{t,ij}, \boldsymbol{H}_{t,j}) =$ `Measurement(`$\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t$, i, j`);`
8 | $\boldsymbol{v}_t = \begin{pmatrix} \boldsymbol{v}_t \\ \boldsymbol{v}_{t,ij} \end{pmatrix}$;
9 | $\boldsymbol{H}_t = \begin{pmatrix} \boldsymbol{H}_t \\ \boldsymbol{H}_{t,j} \end{pmatrix}$;
10 | $\boldsymbol{\Sigma}_{\mathbf{d}_t} = \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{d}_t} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \end{pmatrix}$;
11 **end**
12 **if** $\boldsymbol{H}_t = \emptyset$ **and** $\boldsymbol{v}_t = \emptyset$ **and** $\boldsymbol{\Sigma}_{\boldsymbol{d}_t} = \emptyset$ **then**
13 | $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t} = \bar{\boldsymbol{\mu}}_{\mathbf{y}_t}$;
14 | $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} = \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$;
15 **else**
16 | $\boldsymbol{K}_t = \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t} \boldsymbol{H}_t^\top (\boldsymbol{H}_t \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t} \boldsymbol{H}_t^\top + \boldsymbol{\Sigma}_{\mathbf{d}_t})^{-1}$;
17 | $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t} = \bar{\boldsymbol{\mu}}_{\mathbf{y}_t} + \boldsymbol{K}_t \boldsymbol{v}_t$;
18 | $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t) \bar{\boldsymbol{\Sigma}}_{\mathbf{y}_t}$;
19 **end**

In doing so, the computational complexity of algorithm 4.5 is a composition of the complexities of the for-loop and the correction itself. This yields $\mathcal{O}(M + N^2)$. Assuming again the number M of measurements to be at least one magnitude smaller than the overall number N of map features, this complexity can be simplified to $\mathcal{O}(N^2)$. Thus, the second update version performs better in general.

4.5 Integration

In algorithm 3.5 for the integration step in the EKF-SLAM cycle the function $\bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right)$ in line 5 integrates newly found map features one by one into the believed state $\overline{\overline{\text{bel}}}\left(\mathbf{y}_t\right)$ assumed after the update step. Since integrating a new map feature into the current map does not affect the actual pose $\mathbf{x}_t$ and the already established map features $\mathbf{m}_j,\ j \in [1, N]$, one can rewrite $\bar{\boldsymbol{p}}\left(\bar{\bar{\mathbf{y}}}_t, \bar{\mathbf{z}}_{t,i}\right)$ introduced in equation 3.20 as

$$\mathbf{y}_t = \begin{pmatrix} \mathbf{x}_t \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \\ \mathbf{m}_{N+1} \end{pmatrix} = \bar{\boldsymbol{p}}\left(\bar{\bar{\mathbf{y}}}_t, \bar{\mathbf{z}}_{t,i}\right) = \begin{pmatrix} \bar{\bar{\mathbf{x}}}_t \\ \bar{\bar{\mathbf{m}}}_1 \\ \vdots \\ \bar{\bar{\mathbf{m}}}_N \\ \boldsymbol{p}\left(\bar{\bar{\mathbf{x}}}_t, \bar{\mathbf{z}}_{t,i}\right) \end{pmatrix} \tag{4.33}$$

with the in general non-linear function

$$\mathbf{m}_{N+1} = \boldsymbol{p}\left(\bar{\bar{\mathbf{x}}}_t, \bar{\mathbf{z}}_{t,i}\right) = \boldsymbol{p}\left(\bar{\bar{\mathbf{x}}}_t, \boldsymbol{z}_{t,i} + \mathbf{d}_{t,i}\right) \tag{4.34}$$

estimating the probability of the new map feature $\mathbf{m}_{N+1}$. This is called *linear-state augmentation* in [28] and is expressed more in detail in [17] and [18]. Thus, by equation 3.24 one can immediately derive

$$\boldsymbol{\mu}_{\mathbf{y}_t} = \bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right) = \begin{pmatrix} \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t} \\ \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{m}_1} \\ \vdots \\ \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{m}_N} \\ \boldsymbol{p}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}, \boldsymbol{z}_{t,i}\right) \end{pmatrix} \tag{4.35}$$

for the expectation of the extended state after the integration of one new map feature.

The corresponding covariance matrix was derived in equation 3.25 and is restated at this place for the sake of convenience:

$$\boldsymbol{\Sigma}_{\mathbf{y}_t} = \bar{\boldsymbol{P}}_{\boldsymbol{y}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} \bar{\boldsymbol{P}}_{\boldsymbol{y}_t}^{\top} + \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}}^{\top} \tag{4.36}$$

By means of equations 4.33 and 4.34, one can now specify more precisely

$$\bar{\boldsymbol{P}}_{\boldsymbol{y}_t} = \frac{d\bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right)}{d\bar{\bar{\mathbf{y}}}_t} = \begin{pmatrix} \boldsymbol{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{I} \\ \boldsymbol{P}_{\boldsymbol{x}_t} & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix} \tag{4.37}$$

and

$$\bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}} = \frac{d\bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}, \boldsymbol{z}_{t,i}\right)}{d\bar{\mathbf{z}}_{t,i}} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{P}_{\boldsymbol{z}_{t,i}} \end{pmatrix} \tag{4.38}$$

with

$$\boldsymbol{P}_{\boldsymbol{x}_t} = \frac{d\boldsymbol{p}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}, \boldsymbol{z}_{t,i}\right)}{d\bar{\bar{\mathbf{x}}}_t} \tag{4.39}$$

and

$$\boldsymbol{P}_{\boldsymbol{z}_{t,i}} = \frac{d\bar{\boldsymbol{p}}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}, \boldsymbol{z}_{t,i}\right)}{d\bar{\mathbf{z}}_{t,i}} \tag{4.40}$$

denoting the Jacobian matrices of the function $\boldsymbol{p}\left(\bar{\bar{\mathbf{x}}}_t, \bar{\boldsymbol{z}}_{t,i}\right)$ of sizes 3×3 parameterized at the most probable values of the arguments. This allows us now to expand the covariance matrix $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$ of the current believed state depicted in equation 3.10 in a way such that

$$\boldsymbol{\Sigma}_{\mathbf{y}_t} = \begin{pmatrix} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t\mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t\mathbf{m}_N} & \boldsymbol{\Sigma}_{\mathbf{x}_t\mathbf{m}_{N+1}} \\ \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1\mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1\mathbf{m}_N} & \boldsymbol{\Sigma}_{\mathbf{m}_1\mathbf{m}_{N+1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N\mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N\mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N} & \boldsymbol{\Sigma}_{\mathbf{m}_N\mathbf{m}_{N+1}} \\ \boldsymbol{\Sigma}_{\mathbf{m}_{N+1}\mathbf{x}_t} & \boldsymbol{\Sigma}_{\mathbf{m}_{N+1}\mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_{N+1}\mathbf{m}_N} & \boldsymbol{\Sigma}_{\mathbf{m}_{N+1}} \end{pmatrix} \tag{4.41}$$

describes the covariance matrix after the integration of one new map feature with

$$\boldsymbol{\Sigma}_{\mathbf{m}_{N+1}} = \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top + \boldsymbol{P}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \boldsymbol{P}_{\boldsymbol{z}_{t,i}}^\top, \tag{4.42}$$

$$\boldsymbol{\Sigma}_{\mathbf{m}_{N+1}\mathbf{x}_t} = \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} \tag{4.43}$$

and

$$\boldsymbol{\Sigma}_{\mathbf{m}_{N+1}\mathbf{m}_j} = \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t\mathbf{m}_j}, \;\; j \in [1, N] \tag{4.44}$$

whereas

$$\boldsymbol{\Sigma}_{\mathbf{x}_t\mathbf{m}_{N+1}} = \boldsymbol{\Sigma}_{\mathbf{m}_{N+1}\mathbf{x}_t}^\top \text{ and } \boldsymbol{\Sigma}_{\mathbf{m}_j\mathbf{m}_{N+1}} = \boldsymbol{\Sigma}_{\mathbf{m}_{N+1}\mathbf{m}_j}^\top \tag{4.45}$$

because of the covariance symmetries:

$$\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t\mathbf{m}_{N+1}} = \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_{N+1}\mathbf{x}_t}^\top \text{ and } \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_j\mathbf{m}_{N+1}} = \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_{N+1}\mathbf{m}_j}^\top \tag{4.46}$$

Proof. In order to show equations 4.41 to 4.44, equation 4.36 is first split into its two additive terms $\bar{\boldsymbol{P}}_{\boldsymbol{y}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} \bar{\boldsymbol{P}}_{\boldsymbol{y}_t}^\top$ and $\bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}}^\top$. After inspecting these two terms separately their results are combined obtaining the desired statements. The proof itself was inspired by [17] and [18].

The first term denotes a big matrix multiplication yielding a matrix of size $3(N+2) \times 3(N+2)$. The affected columns and rows are highlighted during evaluation:

$$
\begin{aligned}
&\bar{\boldsymbol{P}}_{\boldsymbol{y}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} \bar{\boldsymbol{P}}_{\boldsymbol{y}_t}^\top = \\
&= \begin{pmatrix} \boldsymbol{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{I} \\ \boldsymbol{P}_{\boldsymbol{x}_t} & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix} \begin{pmatrix} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_N} \\ \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{m}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\ \mathbf{0} & \boldsymbol{I} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{I} & \mathbf{0} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{I} \\ \boldsymbol{P}_{\boldsymbol{x}_t} & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix} \begin{pmatrix} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\ \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \end{pmatrix} \\
&= \begin{pmatrix} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\ \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\ \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_1} & \cdots & \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_N} & \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \end{pmatrix}
\end{aligned}
$$

The second additive is also simply expanded, yielding a sparse matrix of size $3(N+2) \times 3(N+2)$ with a single entry:

$$
\begin{aligned}
\bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}}^\top &= \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{P}_{\boldsymbol{z}_{t,i}} \end{pmatrix} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \begin{pmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{P}_{\boldsymbol{z}_{t,i}}^\top \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{P}_{\boldsymbol{z}_{t,i}} \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \boldsymbol{P}_{\boldsymbol{z}_{t,i}}^\top \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{P}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \boldsymbol{P}_{\boldsymbol{z}_{t,i}}^\top \end{pmatrix}
\end{aligned}
$$

Now both terms are added together in order to achieve the desired results. Again, for better readability the affected columns and rows are highlighted:

$$
\begin{aligned}
\boldsymbol{\Sigma}_{\mathbf{y}_t} &= \bar{\boldsymbol{P}}_{\boldsymbol{y}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t} \bar{\boldsymbol{P}}_{\boldsymbol{y}_t}^\top + \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \bar{\boldsymbol{P}}_{\boldsymbol{z}_{t,i}}^\top \\
&= \begin{pmatrix}
\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\
\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\
\boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_1} & \cdots & \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_N} & \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top
\end{pmatrix} \\
&+ \begin{pmatrix}
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{P}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \boldsymbol{P}_{\boldsymbol{z}_{t,i}}^\top
\end{pmatrix} \\
&= \begin{pmatrix}
\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\
\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top \\
\boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_1} & \cdots & \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_N} & \boldsymbol{P}_{\boldsymbol{x}_t} \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top + \boldsymbol{P}_{\boldsymbol{z}_{t,i}} \boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} \boldsymbol{P}_{\boldsymbol{z}_{t,i}}^\top
\end{pmatrix} \\
&= \begin{pmatrix}
\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{x}_t \mathbf{m}_N} & \boldsymbol{\Sigma}_{\mathbf{x}_t \mathbf{m}_{N+1}} \\
\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_1 \mathbf{m}_N} & \boldsymbol{\Sigma}_{\mathbf{m}_1 \mathbf{m}_{N+1}} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{x}_t} & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N \mathbf{m}_1} & \cdots & \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{m}_N} & \boldsymbol{\Sigma}_{\mathbf{m}_N \mathbf{m}_{N+1}} \\
\boldsymbol{\Sigma}_{\mathbf{m}_{N+1} \mathbf{x}_t} & \boldsymbol{\Sigma}_{\mathbf{m}_{N+1} \mathbf{m}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{m}_{N+1} \mathbf{m}_N} & \boldsymbol{\Sigma}_{\mathbf{m}_{N+1}}
\end{pmatrix}
\end{aligned}
$$

Notice: Equation 4.42 for the covariance of the new map feature can also be derived analogously to equation 3.25 by means of equation 4.34. □

By applying equations 4.41 to 4.44 to the integration step of the EKF-SLAM cycle depicted in algorithm 3.5 one can directly derive algorithm 4.6 used within the implementation. Since the remaining matrices are of constant size the stated complexity in subsection 3.3.4 of chapter 3 can be reduced to $\mathcal{O}(M)$, determined by the surrounding for-loop.

Let us now have a closer look at $\boldsymbol{p}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}, \boldsymbol{z}_{t,i}\right)$ used in line 5 of algorithm 4.6 for obtaining the expectation $\boldsymbol{\mu}_{\mathbf{m}_{N+1}}$ of a newly found map feature $\mathbf{m}_{N+1}$ based on the observation $\boldsymbol{z}_{t,i}$ and the expected pose $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}$ at this time. Basically, this function does the opposite of $\boldsymbol{h}_j\left(\bar{\boldsymbol{\mu}}_{\mathbf{y}_t}\right)$ described in section 4.3. It transforms the measurement $\boldsymbol{z}_{t,i}$ received in spherical coordinates in the sensor frame S into Cartesian coordinates in the map frame M by

Algorithm 4.6: Integration (implemented)

input : $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t} = (\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}\ \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{m}_1} \dots \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{m}_N})^\top$, $\bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$, $\boldsymbol{z}_t = (\boldsymbol{z}_{t,1} \dots \boldsymbol{z}_{t,M})^\top$, $\boldsymbol{c}_t = (c_1 \dots c_M)^\top$
output: $\boldsymbol{\mu}_{\mathbf{y}_t}$, $\boldsymbol{\Sigma}_{\mathbf{y}_t}$

$\boldsymbol{\mu}_{\mathbf{y}_t} = \bar{\bar{\boldsymbol{\mu}}}_{\mathbf{y}_t}$;
$\boldsymbol{\Sigma}_{\mathbf{y}_t} = \bar{\bar{\boldsymbol{\Sigma}}}_{\mathbf{y}_t}$;
for $i = 1$ **to** M **do**
  if $c_i \neq$ new **then continue**;
  $\boldsymbol{\mu}_{\mathbf{m}_{N+1}} = \boldsymbol{p}\left(\boldsymbol{\mu}_{\mathbf{x}_t}, \boldsymbol{z}_{t,i}\right)$;
  $\boldsymbol{\Sigma}_{\mathbf{m}_{N+1}} = \boldsymbol{P}_{\boldsymbol{x}_t} \boldsymbol{\Sigma}_{\mathbf{x}_t} \boldsymbol{P}_{\boldsymbol{x}_t}^\top + \boldsymbol{P}_{\boldsymbol{z}_t} \boldsymbol{\Sigma}_{\mathbf{d}_t} \boldsymbol{P}_{\boldsymbol{z}_t}^\top$;
  $\boldsymbol{\Sigma}_{\mathbf{x}_t \mathbf{m}_{N+1}} = \boldsymbol{P}_{\boldsymbol{x}_t} \boldsymbol{\Sigma}_{\mathbf{x}_t}$;
  for $j = 1$ **to** N **do**
    $\boldsymbol{\Sigma}_{\mathbf{m}_{N+1} \mathbf{m}_j} = \boldsymbol{P}_{\boldsymbol{x}_t} \boldsymbol{\Sigma}_{\mathbf{x}_t \mathbf{m}_j}$;
  end
  $N = N + 1$;
end

means of the currently expected vehicle's pose $\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}$:

$$\boldsymbol{p}\left(\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}, \boldsymbol{z}_{t,i}\right) = \underline{\boldsymbol{I}} \cdot {}^{M}\boldsymbol{T}^{R}_{\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}} \cdot {}^{R}\boldsymbol{T}^{S} \cdot \underline{\boldsymbol{q}\left(\boldsymbol{z}_{t,i}\right)} \tag{4.47}$$

The function

$$\boldsymbol{q}\left(\boldsymbol{z}_{t,i}\right) = \begin{pmatrix} \boldsymbol{z}_{t,i}.r \cos\left(\boldsymbol{z}_{t,i}.\psi\right) \\ \boldsymbol{z}_{t,i}.r \sin\left(\boldsymbol{z}_{t,i}.\psi\right) \\ \boldsymbol{z}_{t,i}.\theta \end{pmatrix} \tag{4.48}$$

denotes the conversion of a pose in spherical coordinates into a pose in Cartesian coordinates. The underline emphasizes again the extension to a homogeneous vector needed for the subsequent transformations at this point. That is why the altered identity matrix

$$\underline{\boldsymbol{I}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.49}$$

is put on the left re-establishing a non-homogeneous pose for $\boldsymbol{\mu}_{\mathbf{m}_{N+1}}$.

Based on this definition, we can now specify the Jacobian matrices $\boldsymbol{P}_{\boldsymbol{x}_t}$ and $\boldsymbol{P}_{\boldsymbol{z}_{t,i}}$ in detail. In particular, we have

$$\boldsymbol{P}_{\boldsymbol{x}_t} = \underline{\boldsymbol{I}} \begin{pmatrix} \frac{\partial {}^{M}\boldsymbol{T}^{R}_{\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}}}{\partial \bar{\bar{\mathbf{x}}}_t.x} & \frac{\partial {}^{M}\boldsymbol{T}^{R}_{\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}}}{\partial \bar{\bar{\mathbf{x}}}_t.y} & \frac{\partial {}^{M}\boldsymbol{T}^{R}_{\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}}}{\partial \bar{\bar{\mathbf{x}}}_t.\theta} \end{pmatrix} \begin{pmatrix} {}^{R}\boldsymbol{T}^{S} \cdot \underline{\boldsymbol{q}\left(\boldsymbol{z}_{t,i}\right)} \\ {}^{R}\boldsymbol{T}^{S} \cdot \underline{\boldsymbol{q}\left(\boldsymbol{z}_{t,i}\right)} \\ {}^{R}\boldsymbol{T}^{S} \cdot \underline{\boldsymbol{q}\left(\boldsymbol{z}_{t,i}\right)} \end{pmatrix} \tag{4.50}$$

and

$$\boldsymbol{P}_{\boldsymbol{z}_{t,i}} = \underline{\boldsymbol{I}} \cdot {}^{M}\boldsymbol{T}^{R}_{\bar{\bar{\boldsymbol{\mu}}}_{\mathbf{x}_t}} \cdot {}^{R}\boldsymbol{T}^{S} \cdot \underline{\boldsymbol{Q}_{\boldsymbol{z}_{t,i}}} \tag{4.51}$$

with

$$\underline{\boldsymbol{Q}_{\boldsymbol{z}_{t,i}}} = \frac{d\boldsymbol{q}\left(\boldsymbol{z}_{t,i}\right)}{d\bar{\mathbf{z}}_{t,i}} \tag{4.52}$$

denoting the Jacobian of the transformation function $\boldsymbol{q}\left(\boldsymbol{z}_{t,i}\right)$ extended to homogeneous coordinates, which yields a matrix of size 4×3.

Notice: Equation 4.50 describes a sparse matrix of the form:

$$\begin{pmatrix} 1 & 0 & \\ 0 & 1 & \\ 0 & 0 & 1 \end{pmatrix} \tag{4.53}$$

Integrating these results into algorithm 4.6 appropriately yields the integration part of the EKF-SLAM cycle implemented in the ROS package related to this thesis.

4.6 ROS Package

As already mentioned, the algorithms accomplished in this chapter have been put into practice by providing a ROS package named `tuw_marker_filter`. The package itself is divided into the three sub-packages

- `tuw_marker_noise`
- `tuw_marker_server`
- `tuw_marker_slam`

discussed in this section. Furthermore, the package is made available to the public online in [3] under the open source BSD license. This shall ensure broad access and the provided documentation there can be used for more detailed information.

4.6.1 tuw_marker_noise

This sub-package serves on the one hand for recording marker measurements and deducing parameters for the measurement noise model discussed in the next chapter. On the other hand it reproduces this measurement noise model for simulation purposes. It is organized as follows:

- `tuw_record.py`
  In order to obtain the parameters for the measurement noise model one needs first to record samples. Thus, the ROS node `tuw_record.py` stores the poses of observed markers in combination with their expected pose in a separate output file *record.csv* for later comparison.

- `variance.py`
  Based on the observed and expected measurements given in the record file a statistical variance can be calculated. For a finer grained resolution `variance.py` separates the measurements into boxes of given precision determined by the expected pose. The calculated variances are written out again with their corresponding expected measurements into a file *variance.csv*. Then `parameter.m` (cp. next point) is called with this CSV-file as input. Finally, the result of this call is stored in another CSV-file named *parameter.csv*.

- `parameter.m`
  Here are the actual parameters of the underlying measurement noise model calculated, using a least square estimation based on the expected poses of the markers and their measured variance.

- `tuw_marker_noise.py`
  This ROS node implements the parameterized measurement noise model for simulation purposes. It receives (perfect) measurements of markers and puts noise on them based on the underlying measurement noise model. The noised poses are then re-sent, illustrating more realistic measurement results.

4.6.2 tuw_marker_server

The purpose of this sub-package is saving maps of explored markers and afterwards providing these marker maps for further localization tasks. It is a composition of following two Python nodes:

- `tuw_marker_saver.py`
  This node saves marker maps to disk using YAML files. A marker map consists of several marker poses augmented with their uncertainty indicated by covariance matrices. Additionally, each marker offers multiple possible IDs and their probabilities.

- `tuw_marker_server.py`
  This node reads such a YAML file including a marker map from disk and periodically publishes it. The so provided map is composed of the found marker poses with their covariance matrices and the markers' possible IDs supplemented by their probabilities.

4.6.3 tuw_marker_slam

This sub-package provides a C++ framework for the SLAM problem with visual markers. In particular, the EKF-SLAM approach discussed in this thesis is implemented there. The package itself consists of following components:

- `tuw_marker_slam_node.cpp`
  This is the actual ROS node executed when performing SLAM. After initialization, the node listens to incoming settings, control commands $\boldsymbol{u}_t$ and measurements $\boldsymbol{z}_t$ as well as it publishes the results, i.e. the believed state $\text{bel}(\mathbf{y}_t)$ expressed by $\boldsymbol{\mu}_{\mathbf{y}_t}$ and $\boldsymbol{\Sigma}_{\mathbf{y}_t}$.

- `tuw_marker_slam.cpp`
  This component represents the interface between the SLAM problem in general and the particular techniques solving it, e.g. EKF-SLAM. Thus, it stores SLAM specific settings and invokes the SLAM cycle.

- `slam_technique.cpp`
  From this component the particular SLAM techniques are inferred. This enables the support of different implementations solving the SLAM problem.

- `ekf_slam.cpp`
  This component implements the actual EKF-SLAM approach discussed in this thesis. This means, the essentials of the algorithms elaborated in this chapter can be found here. Furthermore, it offers EKF-SLAM specific settings, e.g. which version of the update step should be used.

- `measurement_marker.cpp`
  This component illustrates a measurement $\boldsymbol{z}_t$ at time t. Beside the representation introduced in chapter 3 some more properties relevant for practical application are included.

- `munkre.cpp`
  At this place a version of Munkres' algorithm [26] is implemented, solving the minimum assignment problem for rectangular matrices [27]. This is needed to find a global optimum for the data association problem when applying the appropriate NNSF approach to unknown correspondences.

In the next chapter, we will derive a model for the measurement noise expressed by $\boldsymbol{\Sigma}_{\mathbf{d}_{t,i}}$ when working with visual markers. By means of this, we are then ready to discuss the application of the implementation and its results.

CHAPTER 5

Measurement Noise Model and Application

The measurement noise model derived in this chapter outlines the scientific contribution of this thesis. The measurement noise model is important for the application of EKF-SLAM, because it indicates how meaningful the received measurements are and, thus, it determines their impact in the update step of the EKF-SLAM cycle. Wrong assumptions on this noise may have catastrophic consequences. Consider a real bad observation of a visual marker, which is assumed to be correct. Recognizing the same visual marker again, but with a complete different pose, will probably lead to incorrect results. For this reason, we statistically evaluated the measurement quality of the used visual marker detection in oder to derive an appropriate noise model.

Thus, section 5.1 introduces first a general model for the noise. Then, in order to obtain a more precise model, measurements of detected visual markers are recorded and investigated. Based on this insight, a specific noise model for visual marker detection is derived in section 5.2. Finally, section 5.3 discusses the application of the elaborated ROS package with the evolved noise model.

5.1 Proceeding and Measurement Results

As already mentioned in chapter 1, the implementation was first tested in the plain 2D simulation environment Stage. Since this simulation environment provided only perfect measurements, a simplified noise was added for more realistic first tests (cp. `tuw_marker_noise`). The model behind this noise was then reused in the implementation and followed the control noise introduced in chapter 4. In particular, we had for

the measurement noise

$$\boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} = \begin{pmatrix} \sigma_{\mathsf{r}}^2 & 0 & 0 \\ 0 & \sigma_{\varphi}^2 & 0 \\ 0 & 0 & \sigma_{\theta}^2 \end{pmatrix} \tag{5.1}$$

with adjustable $\sigma_{\mathsf{r}}^2 \in [0, \infty)$ and $\sigma_{\varphi}^2, \sigma_{\theta}^2 \in [0, \pi^2)$.

Notice: Because of the zero-covariances in $\boldsymbol{\Sigma}_{\mathbf{d}_{t,i}}$, independence for the noise between the distance r, the angle ϕ and the orientation θ is implicitly assumed.

After successful functionality tests in the 2D simulation environment Stage regarding EKF-SLAM, the implementation was ready to be tested in the more realistic 3D simulation environment Gazebo regarding visual markers. The reason for this was, that Gazebo supports in contrast to Stage the simulation of camera images, which is essential for visual marker detection. Thus, the setting was enhanced by a visual marker detection provided by Markus Bader and adapted by Lukas Pfeifhofer. Unfortunately, tests in Gazebo with this visual marker detection revealed that the simple measurement noise model depicted in equation 5.1 was not sufficient in practice.

Hence, in order to get a feeling for the measurement error of visual markers, a simulation run of a vehicle equipped with a visual marker detection was recorded in Gazebo by calling

```
roslaunch tuw_marker_noise record.launch
```

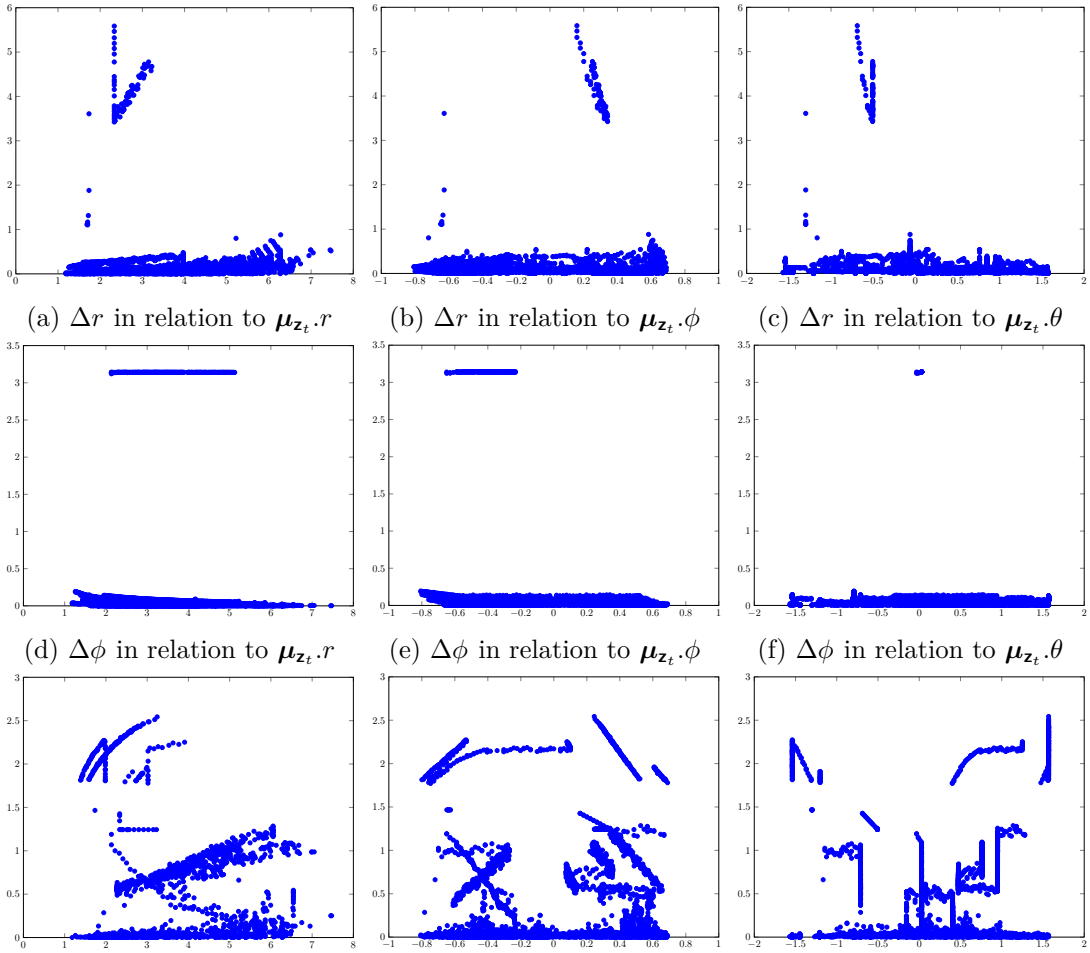
with the vehicles control commands $\boldsymbol{u}_t.v \in [-0.2\,\mathrm{m/s}, 0.2\,\mathrm{m/s}]$ and $\boldsymbol{u}_t.\omega \in [-0.2\,\mathrm{rad/s}, 0.2\,\mathrm{rad/s}]$.

The result of this run was stored in *record.csv* in the output directory of the ROS package `tuw_marker_noise` and is illustrated in figure 5.1. In particular, figures 5.1a to 5.1c show the absolute difference Δr_i between the measured distance $\boldsymbol{z}_{t,i}.r$ and the expected distance $\boldsymbol{\mu}_{\mathbf{z}_{t,i}}.r$, in relation to the expected pose $\boldsymbol{\mu}_{\mathbf{z}_{t,i}}$ of the visual marker relative to the vehicle. The expected pose $\boldsymbol{\mu}_{\mathbf{z}_{t,i}}$ itself is once again obtained by transforming the corresponding true visual marker pose in the map into the sensor frame. Thus, the procedure is similar as we discussed in chapter 4 in the context of the measurement function $\boldsymbol{h}$. But instead of the estimated pose of the vehicle, its real pose provided by the simulation environment is used this time. The same applies to the figures 5.1d to 5.1f and 5.1g to 5.1i but for the absolute differences $\Delta\phi_i$ and $\Delta\theta_i$, respectively.

Summed up, the following relationship is depicted in figure 5.1:

$$\begin{pmatrix} \Delta r_i \\ \Delta\phi_i \\ \Delta\theta_i \end{pmatrix} = |\boldsymbol{z}_{t,i} - \boldsymbol{\mu}_{\mathbf{z}_{t,i}}| \tag{5.2}$$

Notice: In order to ensure proper angle differences, the relation described in equation 4.32 in chapter 4 is again exploited.



(a) Δr in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (b) Δr in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (c) Δr in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

(d) $\Delta\phi$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (e) $\Delta\phi$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (f) $\Delta\phi$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

(g) $\Delta\theta$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (h) $\Delta\theta$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (i) $\Delta\theta$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

Figure 5.1: Measurement error

Based on this insight, we wanted to estimate the variances σ_{r}^2, σ_{φ}^2 and σ_{θ}^2 referred to

$$\hat{\sigma}_{\mathsf{x}}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_{\mathsf{x}})^2 \tag{5.3}$$

for n evidences x_i of a random variable x with the expectation μ_{x} [14].

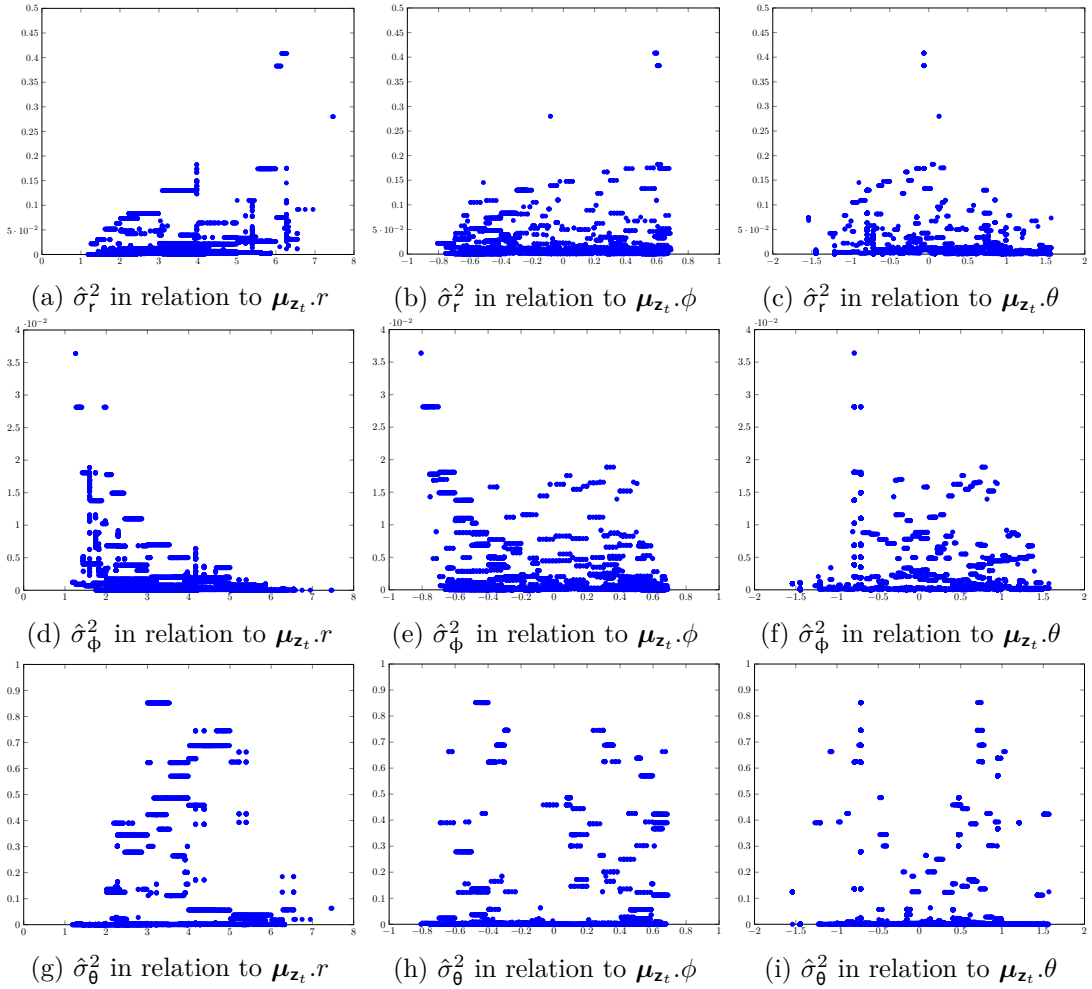
This was achieved by calling

```
./variance.py -r ../output/record.csv -p 1.0
```

inside the source directory of the `tuw_marker_noise` ROS package.

In doing so, the recorded measurements $\boldsymbol{z}_{t,i}$ are filtered and divided into boxes according to their expected pose $\boldsymbol{\mu}_{\mathbf{z}_{t,i}}$ and a precision of 1 m for the distance and 0.1 rad for the angle and orientation. The precision ensures on the one hand enough measurements per box for a meaningful variance estimation, but on the other hand, also a high number of boxes for a sophisticated view. For each box j the different variances $\sigma^2_{\mathsf{r},j}$, $\sigma^2_{\varphi,j}$ and $\sigma^2_{\theta,j}$ can now be estimated yielding $\hat{\sigma}^2_{\mathsf{r},j}$, $\hat{\sigma}^2_{\varphi,j}$ and $\hat{\sigma}^2_{\theta,j}$, where n denotes the number of measurements $\boldsymbol{z}_{t,i}$ inside the actual box. The so obtained variances are stored in the file *variance.csv* in the same directory as *record.csv* before.

Analogous to the measurement error, figure 5.2 depicts the estimated variances $\hat{\sigma}^2_{\mathsf{r},j}$, $\hat{\sigma}^2_{\varphi,j}$ and $\hat{\sigma}^2_{\theta,j}$ in relation to the corresponding expected poses $\boldsymbol{\mu}_{\mathbf{z}_{t,i}}$. Based on this, we can now derive a measurement noise model in the next section.



(a) $\hat{\sigma}^2_{\mathsf{r}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (b) $\hat{\sigma}^2_{\mathsf{r}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (c) $\hat{\sigma}^2_{\mathsf{r}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

(d) $\hat{\sigma}^2_{\phi}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (e) $\hat{\sigma}^2_{\phi}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (f) $\hat{\sigma}^2_{\phi}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

(g) $\hat{\sigma}^2_{\theta}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (h) $\hat{\sigma}^2_{\theta}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (i) $\hat{\sigma}^2_{\theta}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

Figure 5.2: Variance estimation

5.2 Visual Marker Noise Model

The measurement noise model for visual markers is now obtained by considering each variance σ_{r}^2, σ_{ϕ}^2 and σ_{θ}^2 in relation to the individual expectations $\boldsymbol{\mu}_{\mathbf{z}_t}.r$, $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ and $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$ separately, as suggested in figure 5.2.

By investigating figures 5.2a to 5.2c, one can see that all three relations for σ_{r}^2 can be approximated by using a centered parable with vertical offset. Thus, we can write

$$\sigma_{\mathsf{r},\mathsf{r}}^2(r) = \max\left(\beta_1 r^2 + \beta_2,\, 0\right), \tag{5.4}$$

$$\sigma_{\mathsf{r},\phi}^2(\phi) = \max\Big(\min\left(\beta_3\phi^2 + \beta_4,\, \pi^2\right),\, 0\Big) \text{ and} \tag{5.5}$$

$$\sigma_{\mathsf{r},\theta}^2(\theta) = \max\Big(\min\left(\beta_5\theta^2 + \beta_6,\, \pi^2\right),\, 0\Big) \tag{5.6}$$

regarding each individual expectation and with $\beta_i,\ i \in [1, 6]$ denoting adjustable parameters.

Considering figures 5.2d to 5.2f, one can see that the later two relations for σ_{ϕ}^2 can be described as we did for σ_{r}^2. But in contrast, the variance σ_{ϕ}^2 in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ is approximated better by an inverted parable with vertical offset. This yields

$$\sigma_{\phi,\mathsf{r}}^2(r) = \max\left(\frac{\beta_7}{r^2} + \beta_8,\, 0\right), \tag{5.7}$$

$$\sigma_{\phi,\phi}^2(\phi) = \max\Big(\min\left(\beta_9\phi^2 + \beta_{10},\, \pi^2\right),\, 0\Big) \text{ and} \tag{5.8}$$

$$\sigma_{\phi,\theta}^2(\theta) = \max\Big(\min\left(\beta_{11}\theta^2 + \beta_{12},\, \pi^2\right),\, 0\Big) \tag{5.9}$$

regarding each individual expectation and with $\beta_i,\ i \in [7, 12]$ denoting adjustable parameters, again.

Finally, by inspecting figures 5.2g to 5.2i, we derive for the variance σ_{θ}^2 again parabolic approximations

$$\sigma_{\theta,\mathsf{r}}^2(r) = \max\left(\beta_{13} r^2 + \beta_{14},\, 0\right), \tag{5.10}$$

$$\sigma_{\theta,\phi}^2(\phi) = \max\Big(\min\left(\beta_{15}\phi^2 + \beta_{16},\, \pi^2\right),\, 0\Big) \text{ and} \tag{5.11}$$

$$\sigma_{\theta,\theta}^2(\theta) = \max\Big(\min\left(\beta_{17}\theta^2 + \beta_{18},\, \pi^2\right),\, 0\Big) \tag{5.12}$$

regarding each individual expectation. In turn, $\beta_i,\ i \in [13, 18]$ denote adjustable parameters.

Notice: The surrounding minimum and maximum functions ensure valid domains for the particular variances as specified in the context of equation 5.1.

In a next step, the parameters $\beta_i,\ i \in [1, 18]$ of the introduced approximation functions were determined by a least square estimation of the corresponding variances and individual expectations, respectively. In particular, by calling

```
parameter("../output/variance.csv")
```

in a MATLAB/Octave shell started inside the source directory of the `tuw_marker_noise` ROS package, the file *parameter.csv* is created in the output directory of the package. The so obtained values for the parameters are retained in the tables 5.1 to 5.3.

Notice: The file *parameter.csv* is already created during the execution of *variance.py* by means of a sub-call to *parameter.m*. Nevertheless, for the sake of completeness it is restated at this place.

i	β_i	β_{i+1}
1	0.00151087929622	0.00307593942987
3	0.0209484955631	0.0190882544679
5	-0.0087332784877	0.0281445351132

Table 5.1: Estimated parameters for σ_{r}^2

i	β_i	β_{i+1}
7	0.0200817239281	-0.000409312497586
9	0.00850526786062	0.00102221836963
11	-0.00073979880801	0.00286934645709

Table 5.2: Estimated parameters for σ_{ϕ}^2

i	β_i	β_{i+1}
13	0.00257637663063	0.0721725283535
15	-0.00283353322719	0.105663927557
17	-0.00812729757853	0.110508496545

Table 5.3: Estimated parameters for σ_{θ}^2

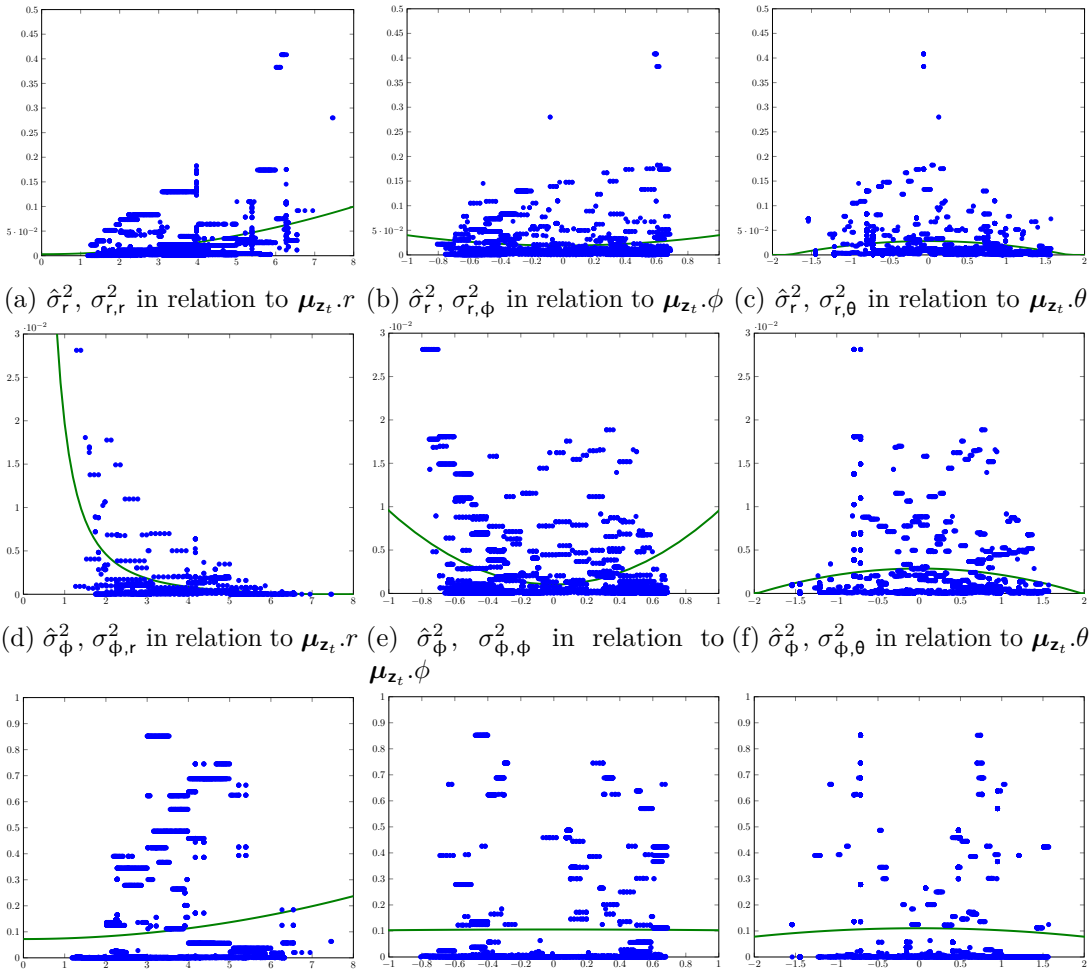
Furthermore, in figure 5.3 the specific approximation functions yielded by these parameters are compared with the corresponding estimated variances. The latter should be already familiar from figure 5.2. One can see, that the resulting approximation functions for the individual variances are fitting nicely to the previously estimated variances.

These individually approximated variances are now composed in order to derive single expressions for σ_{r}^2, σ_{ϕ}^2 and σ_{θ}^2 dependent on the actual measurement $\boldsymbol{z}_{t,i}$:

$$\sigma_{\mathsf{r}}^2\left(\boldsymbol{z}_{t,i}\right) = \sigma_{\mathsf{r},\mathsf{r}}^2\left(\boldsymbol{z}_{t,i}.r\right) + \sigma_{\mathsf{r},\phi}^2\left(\boldsymbol{z}_{t,i}.\phi\right) + \sigma_{\mathsf{r},\theta}^2\left(\boldsymbol{z}_{t,i}.\theta\right) \tag{5.13}$$

$$\sigma_{\phi}^2\left(\boldsymbol{z}_{t,i}\right) = \min\left(\sigma_{\phi,\mathsf{r}}^2\left(\boldsymbol{z}_{t,i}.r\right) + \sigma_{\phi,\phi}^2\left(\boldsymbol{z}_{t,i}.\phi\right) + \sigma_{\phi,\theta}^2\left(\boldsymbol{z}_{t,i}.\theta\right),\, \pi^2\right) \tag{5.14}$$

$$\sigma_{\theta}^2\left(\boldsymbol{z}_{t,i}\right) = \min\left(\sigma_{\theta,\mathsf{r}}^2\left(\boldsymbol{z}_{t,i}.r\right) + \sigma_{\theta,\phi}^2\left(\boldsymbol{z}_{t,i}.\phi\right) + \sigma_{\theta,\theta}^2\left(\boldsymbol{z}_{t,i}.\theta\right),\, \pi^2\right) \tag{5.15}$$



(a) $\hat{\sigma}^2_{\mathsf{r}}$, $\sigma^2_{\mathsf{r},\mathsf{r}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (b) $\hat{\sigma}^2_{\mathsf{r}}$, $\sigma^2_{\mathsf{r},\Phi}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (c) $\hat{\sigma}^2_{\mathsf{r}}$, $\sigma^2_{\mathsf{r},\theta}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

(d) $\hat{\sigma}^2_{\Phi}$, $\sigma^2_{\Phi,\mathsf{r}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (e) $\hat{\sigma}^2_{\Phi}$, $\sigma^2_{\Phi,\Phi}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (f) $\hat{\sigma}^2_{\Phi}$, $\sigma^2_{\Phi,\theta}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

(g) $\hat{\sigma}^2_{\theta}$, $\sigma^2_{\theta,\mathsf{r}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (h) $\hat{\sigma}^2_{\theta}$, $\sigma^2_{\theta,\Phi}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (i) $\hat{\sigma}^2_{\theta}$, $\sigma^2_{\theta,\theta}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

Figure 5.3: Estimated variance vs. approximation function

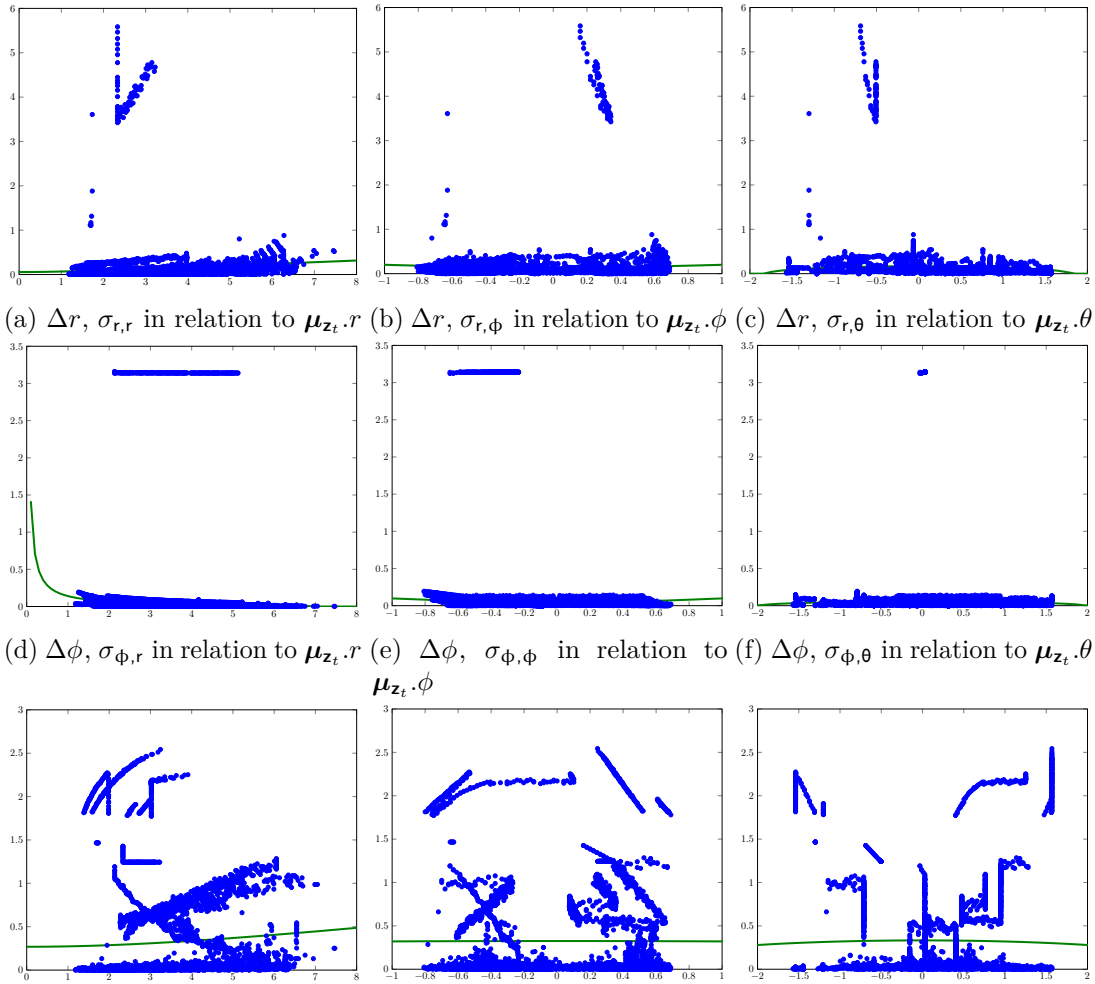
A side effect of this composition is, that it makes the system more robust against measurement spikes by implicitly increasing the total variance. Furthermore, the surrounding minimum function in the case of the angle and the orientation ensures again valid domains for the corresponding variances.

By means of this, we can restate the simple noise model introduced in equation 5.1 by the final measurement noise model for visual markers used within the implementation:

$$\boldsymbol{\Sigma}_{\mathbf{d}_{t,i}} = \begin{pmatrix} \sigma^2_{\mathsf{r}}(\boldsymbol{z}_{t,i}) & 0 & 0 \\ 0 & \sigma^2_{\Phi}(\boldsymbol{z}_{t,i}) & 0 \\ 0 & 0 & \sigma^2_{\theta}(\boldsymbol{z}_{t,i}) \end{pmatrix} \tag{5.16}$$

In contrast to the original one, it is more situation accurate by making it dependent on the actual measurement $\boldsymbol{z}_{t,i}$.

Finally, figure 5.4 compares the measurement error with the particular standard deviations obtained by the corresponding approximation functions. Again, one can see that the approximation is quite usable. Unfortunately, there are spikes in the measurements, which might be problematic in larger numbers.



(a) Δr, $\sigma_{\mathtt{r,r}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (b) Δr, $\sigma_{\mathtt{r,\phi}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (c) Δr, $\sigma_{\mathtt{r,\theta}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

(d) $\Delta\phi$, $\sigma_{\mathtt{\phi,r}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (e) $\Delta\phi$, $\sigma_{\mathtt{\phi,\phi}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (f) $\Delta\phi$, $\sigma_{\mathtt{\phi,\theta}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

(g) $\Delta\theta$, $\sigma_{\mathtt{\theta,r}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.r$ (h) $\Delta\theta$, $\sigma_{\mathtt{\theta,\phi}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\phi$ (i) $\Delta\theta$, $\sigma_{\mathtt{\theta,\theta}}$ in relation to $\boldsymbol{\mu}_{\mathbf{z}_t}.\theta$

Figure 5.4: Measurement error vs. approximation function

Nevertheless, we will see in the next section that the introduced visual marker noise model has indeed proven to be applicable.

5.3 Application and Results

In this section, the results of the implementation corresponding to this thesis in conjunction with the measurement noise model for visual markers are discussed. Although both simulation environments - Stage and Gazebo - are used for this purpose, they both have the visual marker noise model elaborated in the previous section in common. Furthermore, the visualization environment RViz enhanced by a plugin by Lukas Pfeifhofer for visual markers is used for illustrating the EKF-SLAM result as already mentioned at the beginning of this thesis. In particular, integrated visual markers are depicted by small coordinate systems indicating their estimated position as well as orientation. The corresponding uncertainty of the estimated pose is again expressed by a surrounding ellipse.

All the discussed results can easily be reproduced by either calling

```
roslaunch tuw_marker_slam slam_demo_gazebo.launch
```

using Gazebo for simulation or

```
roslaunch tuw_marker_slam slam_demo_stage.launch
```

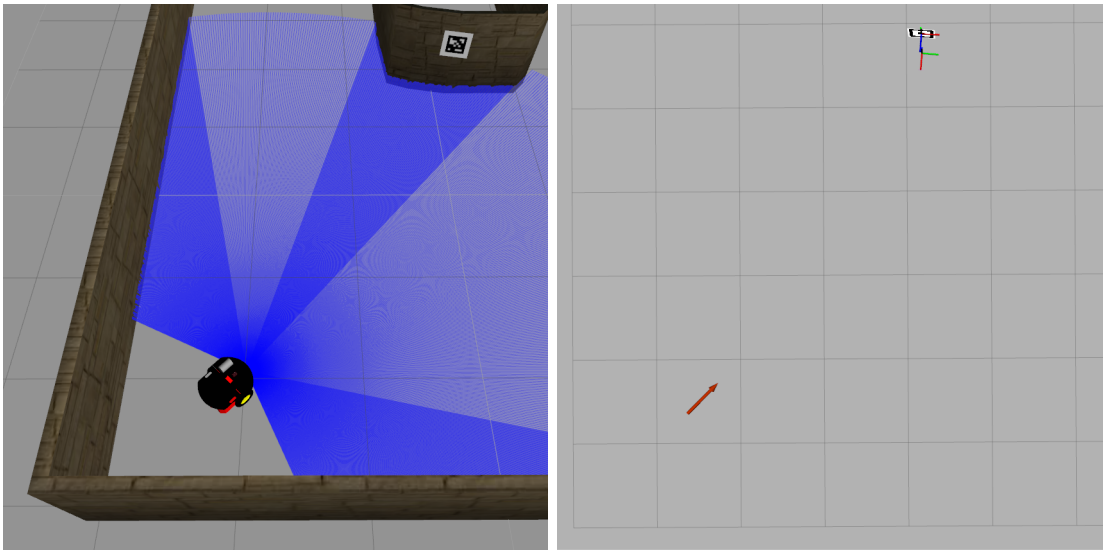
using Stage as simulation environment. For the navigation of the robot in Stage, the ROS package `tuw_teleop` provided by Markus Bader can be used by subsequently calling the command:

```
rosrun tuw_keyboard2twist tuw_keyboard2twist_node
```

When using Gazebo, `__ns:=r1` needs to be appended to the command.

First of all, in order to verify the correct reception of the detected visual markers and their integration into the map, the robot is placed close to a visual marker. Figure 5.5 depicts this setting in Gazebo and the corresponding EKF-SLAM result in RViz. In the bottom left corner the robot is positioned and in the upper right corner the detected visual marker.

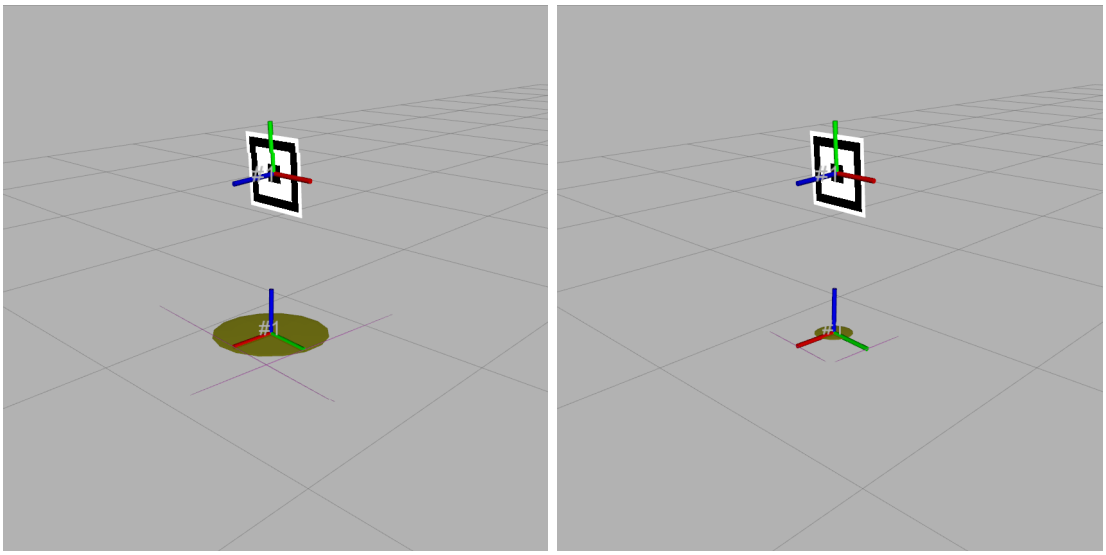
In figure 5.6, we have a closer look on the same visual marker. Since Gazebo is a 3D simulation environment, the received pose of the detected visual marker - illustrated by a plain visual marker - is in space. But the introduced EKF-SLAM implementation is designed for 2D. Thus, the received pose is projected into the underlying plane before integrating it into the map. Furthermore, figure 5.6 demonstrates the effect of the update step. The first time when the visual marker is observed, it is integrated into the map



(a) Gazebo (b) RViz

Figure 5.5: Testing visual marker detection and integration into the map

with a rather big uncertainty represented by the surrounding ellipse. But the more often the visual marker is re-observed the smaller gets this uncertainty.



(a) Variance at $t = 0$ (b) Variance after Δt

Figure 5.6: Detected visual marker and maintained marker in the map with decreasing variance

A well-known effect in EKF-SLAM is the *loop closure*. It means the moment, when an

already established visual marker is re-visited after a while. Between these two points in time the uncertainty in the vehicle's pose grows. But as soon as the relevant marker is observed again, the vehicle's uncertainty as well as the uncertainties of all the visual markers detected in the meantime immediately shrink. This effect is depicted in the figures 5.7 and 5.8, but this time using Stage as simulation environment.

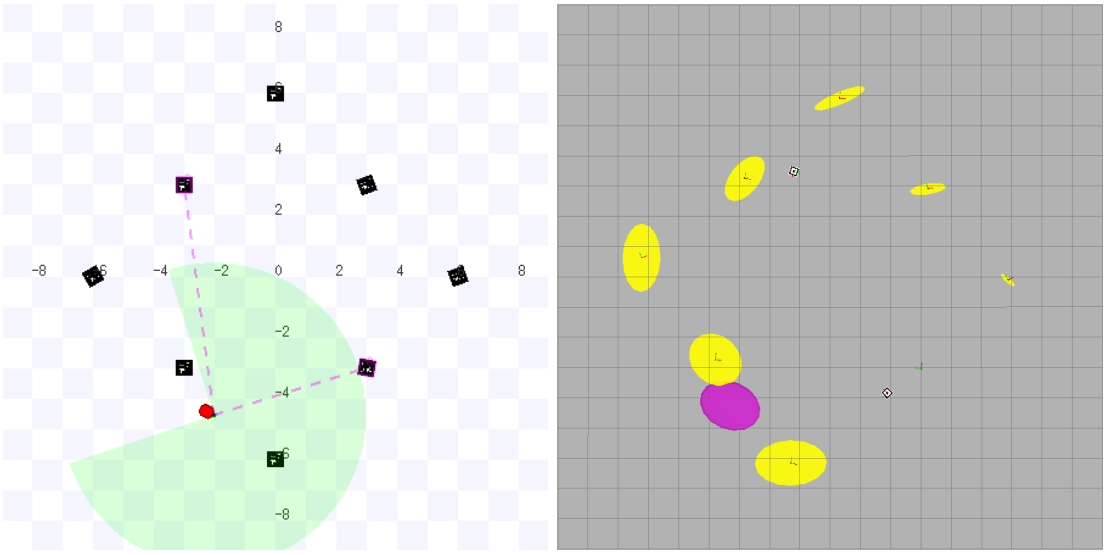
The reason why Stage is used here, is because of the geometric symmetry of the provided world emphasizing the effect. Furthermore, observations are distinguished by dashed lines in Stage and by small plain visual markers in RViz, again. The shaded segment of the circle around the vehicle in Stage denotes the area, in which visual markers and their IDs are detected. The remaining measurements, e.g. the upper dashed line in figure 5.7a, are neglected because we are only working with known correspondences at this point.

Coming back to the loop closure, both figures 5.7 and 5.8 show on the left the simulated world in Stage and on the right the current state of the EKF-SLAM visualized in RViz. The vehicle starts at the pose $(0\ 5\ 0)^\top$ and drives a circle with constant velocity $\boldsymbol{u}_t.v = 1.0\,\mathrm{m/s}$ and angular rate $\boldsymbol{u}_t.\omega = 0.2\,\mathrm{rad/s}$. At the beginning the vehicle is sure about its pose and, thus, the uncertainty of the detected marker at the position $(3\ -3)^\top$ is minimal. While moving the vehicle's uncertainty increases, which also affects the uncertainty of the observations during the vehicle's trip. In figure 5.7 the vehicle is right before its starting point with a grown uncertainty in its pose and the map. It already observes the first marker, but without ID and, thus, the measurement is refused yet. But a few moments later, when the first received marker appears inside the ID range, it immediately becomes sure about its pose again. Furthermore, at the same time the uncertainty of the map features shrinks too, because their observations become more reliable. Figure 5.8 reflects this occurrence.

For figure 5.9, the described scenario is replayed. But this time the NNSF approach described in chapter 3 is applied. Hence, not only measurements containing an ID but also measurements without ID are processed. Although the vehicle has not finished the first round yet, its pose and map estimation is much better than without NNSF at this point in time. The reason for this is a much earlier loop closure based on former unknown correspondences revealed by NNSF.

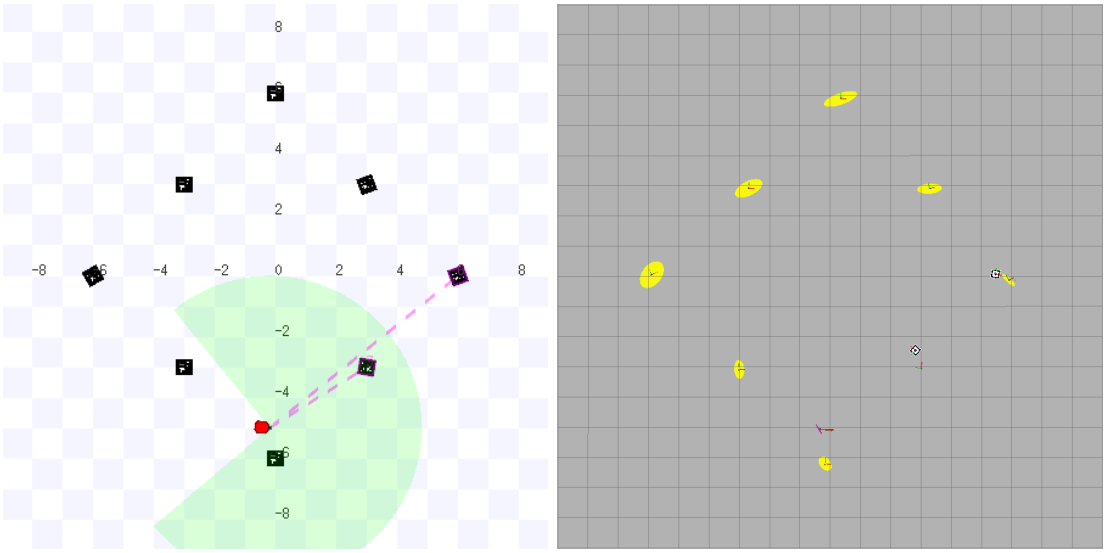
Although the NNSF approach enables us to take advantage of all observations, it is also a source of error. Consider figure 5.10, which depicts once again the vehicle on its round in conjunction with the estimated map. Compared to before the internal threshold, at which an observation without ID is still associated with a map feature, is higher. In figure 5.11, the vehicle detects an actually unknown visual marker. Since the received measurement contains no ID and NNSF is enabled, it tries to match the observation with an already known map feature. Although the measurement and the first of all found visual markers are far off, they get associated, because of the vehicles own uncertainty and the high threshold. Thus, the map is irreversibly distorted making further processing even more error-prone.

Another interesting case is depicted in figure 5.12. Until now there was always a visual



(a) Stage (b) RViz

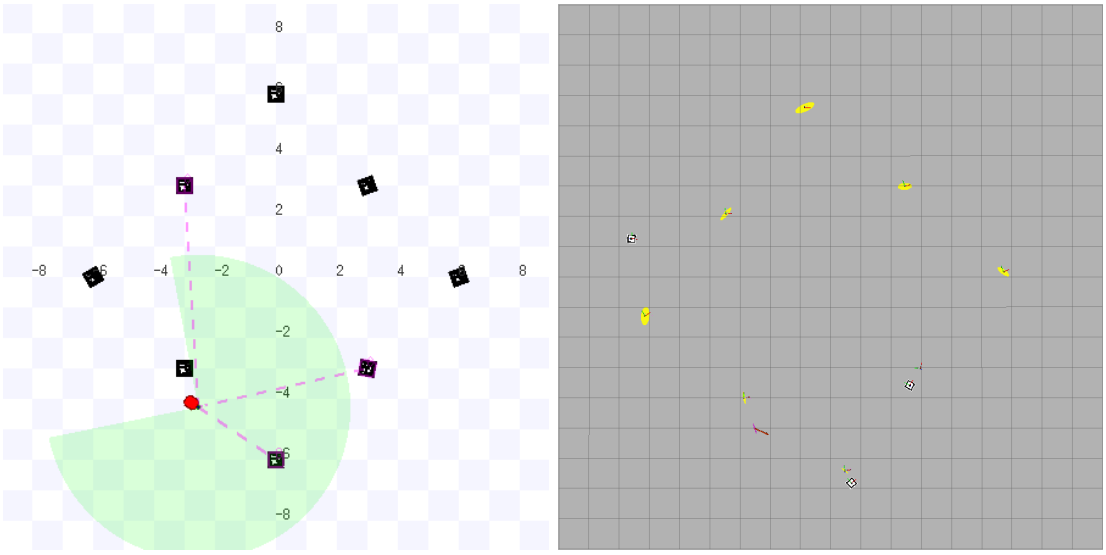
Figure 5.7: Before loop closure



(a) Stage (b) RViz

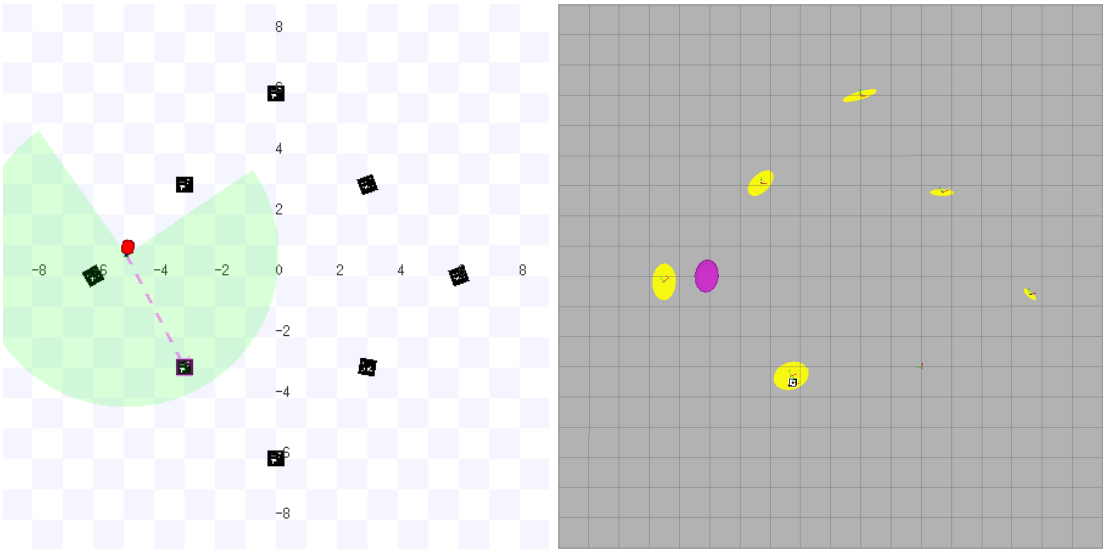
Figure 5.8: After loop closure

marker in the field of view at the beginning. This marker was used later on as a reference point re-establishing certainty about the robot's pose and the created map. This time there is no such initial reference point (cp. figure 5.12a). Instead the robot starts a journey into the unknown. Remember, although there is already an observation at the beginning, this observation is outside the ID range and, thus, it is not integrated into



(a) Stage (b) RViz

Figure 5.9: Loop closure with NNSF ($\alpha = 0.95$)



(a) Stage (b) RViz

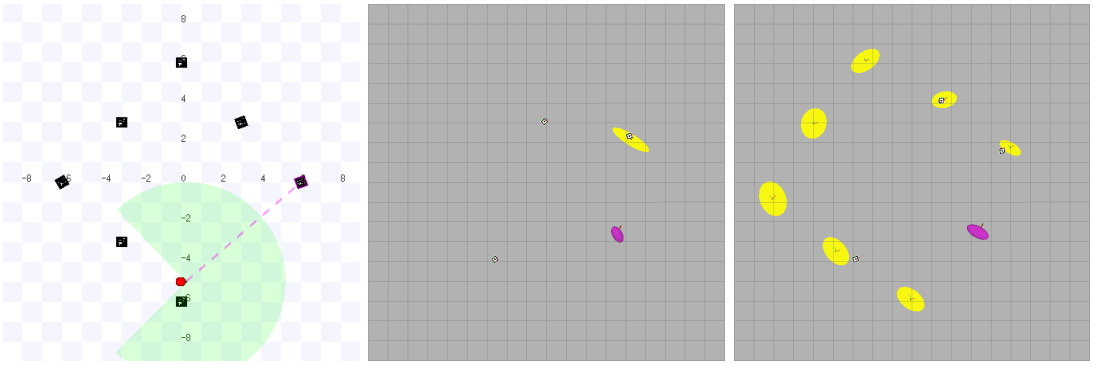
Figure 5.10: Before erroneous loop closure with NNSF ($\alpha = 0.995$)

the map yet. This is also true with enabled NNSF, since an ID needs to be stored when integrating a visual marker into the map for possible later associations based on IDs. But as soon as the robot comes close enough to recognize the ID, the visual marker gets incorporated into the map. Unfortunately, the robot is not sure anymore about its pose at this time. Consequently, the uncertainty of the obtained map feature is even

(a) Stage (b) RViz

Figure 5.11: After erroneous loop closure with NNSF ($\alpha = 0.995$)

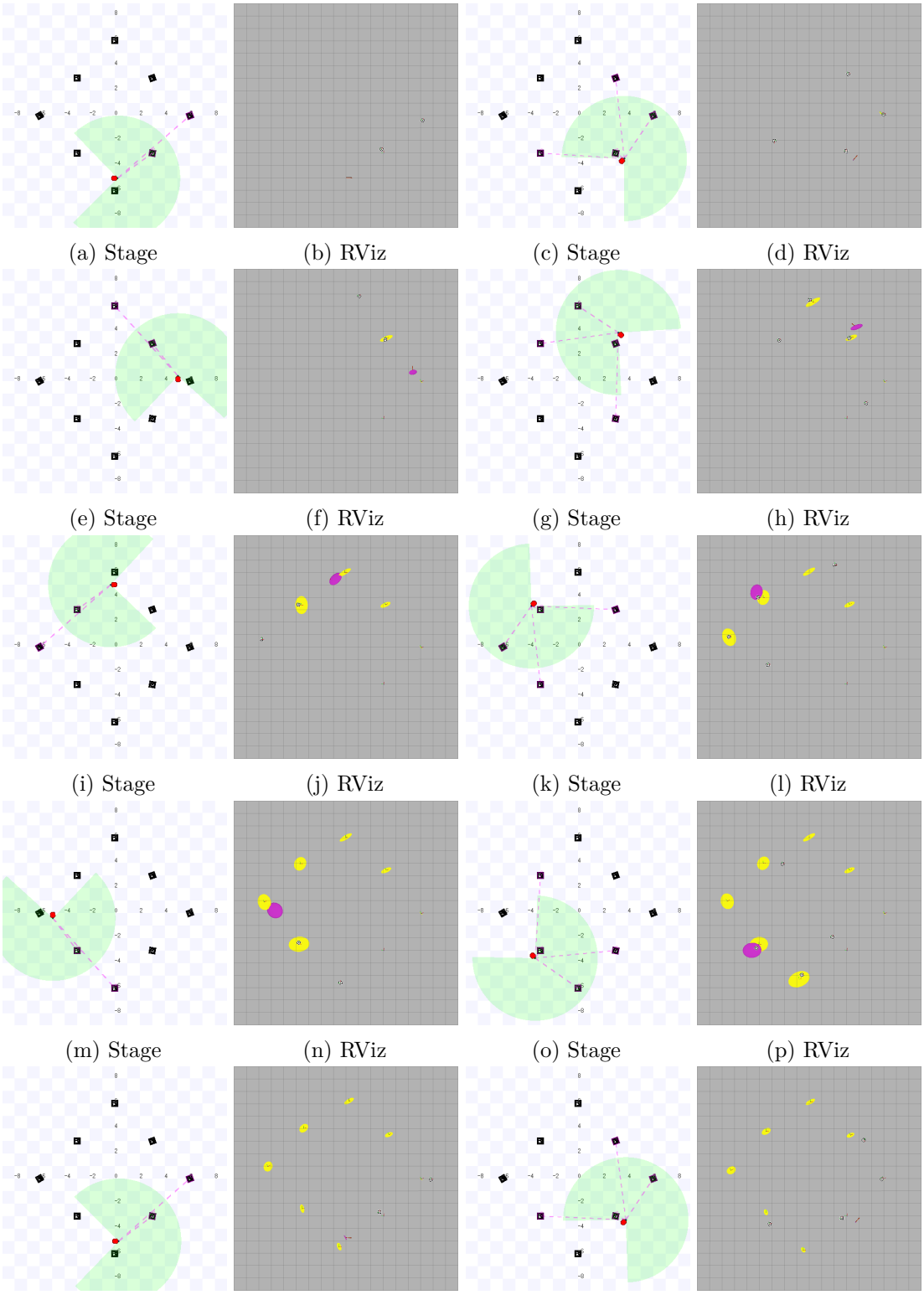
higher because of the additional measurement noise. Figure 5.12b illustrates this moment. Further, the robot continues its round discovering the other visual markers and coming back to the first visual marker. Until now, the robot became quite sure about its pose again at this point by the loop closure. This time the uncertainty is indeed reduced, but there still remains a significant portion of it. The reason is, the vehicle can never gain more certainty than it had initially, when detecting the first visual marker for reference. Although it is kind of natural, figure 5.12c confirms this fact.



(a) Stage (b) RViz (first round) (c) RViz (second round)

Figure 5.12: No initial reference point

For a better understanding, figure 5.13 recaps once again a full loop of the vehicle without using NNSF. The sub-figures are arranged pairwise from top left to bottom right.



(a) Stage (b) RViz (c) Stage (d) RViz

(e) Stage (f) RViz (g) Stage (h) RViz

(i) Stage (j) RViz (k) Stage (l) RViz

(m) Stage (n) RViz (o) Stage (p) RViz

(q) Stage (r) RViz (s) Stage (t) RViz

Figure 5.13: A full loop in the simulation environment Stage without NNSF

Finally, we come back to the simulation environment Gazebo. So far an artificial noise based on the measurement error, recorded in the previous section, was added to the observations in Stage. In contrast, Gazebo allows the application of a separate visual marker detection based on simulated camera images. Thus, the measurements are inherently biased and we can examine the noise model for visual markers introduced in the previous section. In doing so, the vehicle equipped with a camera and the visual marker detection is navigated through the simulated world depicted in figure 5.14 with $\boldsymbol{u}_t.v \in [-0.2\,\mathrm{m/s}, 0.2\,\mathrm{m/s}]$ and $\boldsymbol{u}_t.\omega \in [-0.2\,\mathrm{rad/s}, 0.2\,\mathrm{rad/s}]$. Furthermore, all visual markers placed in this world are distinguished by circles.

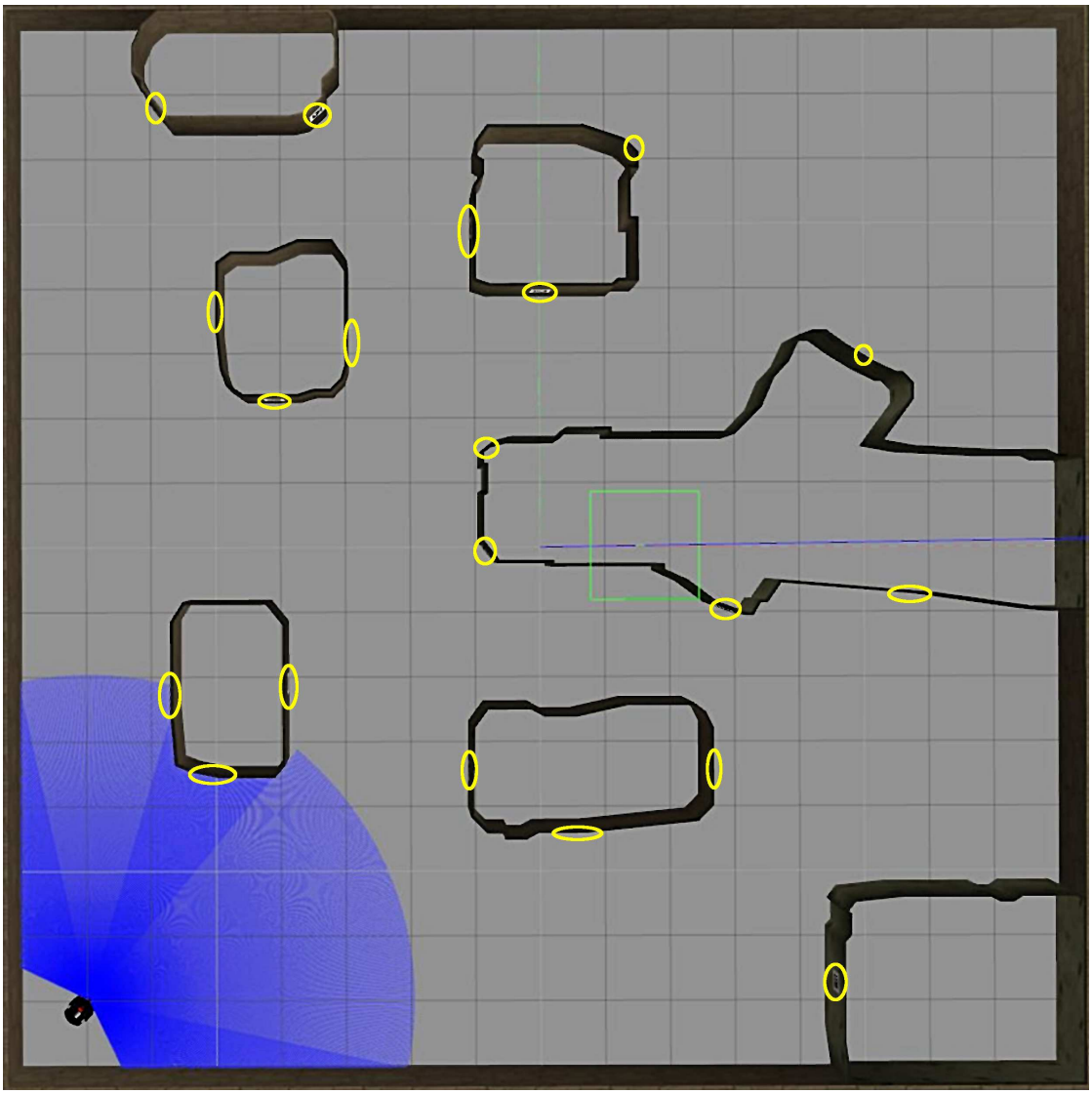


Figure 5.14: Simulated world in Gazebo with highlighted visual markers

This enables better comparability with the result of the EKF-SLAM depicted in figure 5.15. As we can see the recorded marker map is in fact close to the actual world and the visual markers in it. The uncertainty of the estimated visual marker poses increases the later they were observed. This is due to the increased uncertainty in the vehicle's estimated pose at the time of their integration into the map. Comparing figure 5.14 and figure 5.15 demonstrates the operational capability of the implementation as well as the elaborated measurement noise model under more realistic circumstances.

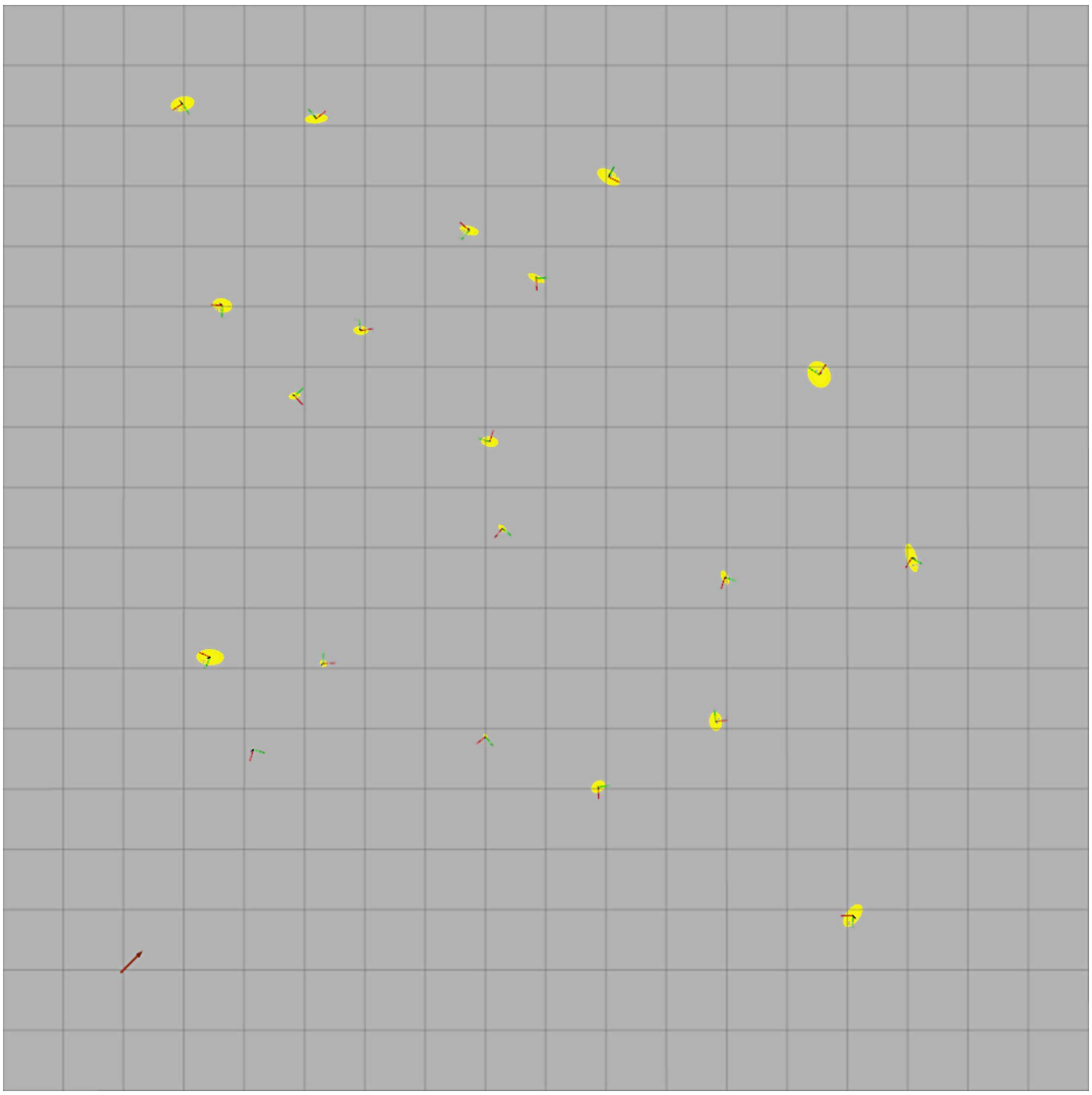


Figure 5.15: Result of EKF-SLAM using visual marker

In the last chapter, some challenges and drawbacks of the introduced implementation are revealed. These are then compared with the latest developments in the area of SLAM,

giving an insight into the state of the art. Finally, it concludes the thesis with a brief summary of the work and with future prospects of the implementation.

CHAPTER 6

Conclusion

The SLAM topic is well-studied. Yet, because of its computational complexity it could only be solved theoretically in the past. But hardware developments in the recent decades have made it possible to apply "*the solution in a number of different domains from indoor robots to outdoor, underwater, and airborne systems.*" [29] Due to its tremendous impact on mobile robotics, the topic became popular in the last decade. Furthermore, ROS evolved in the field of robotics in the last few years. Since there has not been any comparable EKF-SLAM implementation for ROS so far, we decided to enrich its functionality by an appropriate package. The scientific contribution to this package is the statistically evaluated measurement noise model for the visual marker detection used.

Although we provided a working solution, section 6.1 depicts some challenges and drawbacks of the implementation introduced and of EKF-SLAM in general. These challenges are then compared with state of the art developments in section 6.2. Section 6.3 concludes the thesis with a short summary and gives an outlook on further improvements as well as extensions of the presented implementation.

6.1 Challenges and Drawbacks

In the field of EKF-SLAM, considerations have to be taken into account regarding robustness. The EKF-SLAM approach introduced in chapter 3 using visual markers circumvents a few of the traditional problems in plain EKF-SLAM described in [1]. Nevertheless, some of these challenges and common drawbacks of EKF-SLAM are recapped in this section and complemented with problems specific to the presented implementation.

In general, EKF-SLAM is only applicable to the online SLAM problem. For global SLAM the state vector would have to be extended by the current pose estimation of each time step. This results in an infinite growth of the state vector and, thus, of the computational

effort too, regardless of how efficient the implementation is or how fast the processors used are. [1]

A common problem is the ideal number of features. An insufficient number of features is obviously bad for localization, but too many features are bad for the data association problem of unknown correspondences as well. As we have seen in the previous chapter, if visual markers are placed tightly together, the assignment of measurements with the correct map features is difficult. Thus, the probability of wrong correspondences rises. Unfortunately, decisions using NNSF for data association are hard and cannot be revoked later on. This makes further processing error-prone. Moreover, the number of map features obviously has a critical impact on the computational effort, because of the quadratic complexity of the EKF-SLAM cycle.

A main drawback of EKF-SLAM is the selection of appropriate features. Often lines, corners or other geometrical structures are filtered out of a video stream provided by a camera. The same holds for the visual marker approach used within this thesis. Beside the additional computational effort, one must also keep in mind the loss of information associated with it. This loss of information is increased once more in the presented solution for 2D. Although the measurements obtained from the visual marker detection are three-dimensional, they are projected into the plane for compatibility. This yields a reduction from 6 to 3 DOFs.

Furthermore, the orientation information received from the original visual marker detection was quiet poor. Thus, the corresponding variance in the measurement noise model was increased to a level at which the observed orientation was pretty much neglected. This loss of information had a noticeable negative impact on the EKF-SLAM result. The so created maps often began to turn while processing. This effect can be comprehended by imagining a pillar. No matter from which direction the pillar is seen, it always looks the same. Thus, we can never distinguish from which side we actually observe it. As a consequence, the orientation information has proven to be essential for meaningful EKF-SLAM results.

The original visual marker detection revealed another problem of the provided implementation. Sometimes it happened that the observed ID of a visual marker was not recognized correctly. As a consequence, the corresponding visual marker was added more than once to the established map. Since the wrong ID was detected rather infrequently, there remained a spurious map feature with high uncertainty. In the adapted visual marker detection used later on, this misbehavior was not noticed anymore during simulation runs.

Another drawback regarding angles, is the used *Euler angle* approach, which assumes angles between $-\pi$ and π. Unfortunately, the domain of the inverse tangent function is usually $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. This problem was circumvented by the common extension of the inverse tangent function illustrated in chapter 4. But further problems arise when performing operations with them, e.g. for the angle difference we exploited once again the extended inverse tangent function in combination with the sinus and cosinus function.

Moreover, we need to ensure proper angle values when transforming homogeneous pose vectors. A well-known approach avoiding these problems is the usage of *quaternions* instead of Euler angles.

Beside that, the presented transformations should be actually theoretical, because ROS internally provides a transformation system of its own. Using these transformations instead, would obviously avoid a possible source of error and make the provided solution more compatible to other settings. In particular, there has been built in a case distinction for Stage and Gazebo adjusting the received measurements appropriately. Nevertheless, the usage of the presented transformations has been preferred because it is facilitating the deviation of the corresponding Jacobian matrices. Furthermore, in the case of Gazebo the observations had to be projected into the plane anyway and in both cases they needed to be transformed from Cartesian coordinates back to spherical coordinates. The reason is that the visual marker detection stores the poses for further processing in Cartesian coordinates although the visual markers are measured in spherical coordinates.

Apart from that, the symmetry of covariance matrices is not utilized in the implementation by storing them as triangular matrices. This was suggested when elaborating the Kalman Filter in chapter 2 in order to reduce its complexity by exploiting the speed up of triangular matrix multiplications. Instead, floating point errors during multiplication corrupt the covariance symmetry, which needs then to be recovered by hand from time to time. Obviously, this increases the computational effort rather than decreasing it. But it is required, since non-conforming covariance matrices revealed a tremendous negative impact on the EKF-SLAM result. Besides that, neither discussed update step exploits entirely the sparse Jacobian matrix for the Kalman gain as it was done in the prediction and integration step. Thus, the computational complexity of the implementation is higher than specified.

Finally, the motion model used in the prediction step of the EKF-SLAM cycle is rather simple. Again, it uses Euler angles, for which reason the described case distinction in chapter 4 is required. This is called *gimbal lock* in [30]. For this purpose, a motion model exploiting the usage of quaternions has been developed. But exchanging the two motion models goes hand in hand with replacing the corresponding control noise with a more complex one. In [31], they successfully implemented such a motion model using quaternions for the independent steering of an autonomous system.

6.2 State of the Art and Comparison with related Work

Although the SLAM problem is well studied in theory, which is shortly recapped in [29], [28], in practice several problems still exist. These include firm real time demands on processing time and limited memory, e.g. on embedded systems.

For this reason, consider the data association problem between detected visual markers without IDs and features of the created map. A naive implementation as suggested in this thesis iterates over all possible visual markers resulting in linear complexity. In [32],

they introduced *octrees* as data structure fighting this problem. The limited memory problem is addressed in [33]. Their approach is to keep just the currently required extract of the map in memory and store the remaining part on hard-disc. Thus, the focus of this paper is mainly the update of the global map with the local one.

Beside the reduction of memory consumption, the usage of local maps has further advantages. In [34], the computational complexity of the EKF-SLAM cycle could be reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ by means of a *Divide and Conquer* approach using submaps. In addition, they claimed that working with smaller local maps lowers the angular error of measurements. This, in turn, keeps the linearization error small, which is introduced by the first order Taylor approximation used in EKFs.

Further problems regarding especially visual SLAM, i.e. SLAM using visual sensors like cameras, are the limitation of the sensors used as well as the use of low level features such as *salient points*. A possible solution to overcome these problems is presented in [35]. They extract e.g. points, line segments, lines, and planes from the underlying sensor data and use them as so called high level features.

Additional research is done in complementing different approaches. This includes the exchange of multiple vehicles performing SLAM [36] as well as combining observations from different sensor types [37]. The former describes a setting of multiple robots performing SLAM, which forward their local maps to a server. The server, in turn, merges the different local maps to a global one and returns it to the robots for further improvement. The second paper combines range-only measurements as discussed in [38] with visual markers. The reason is to augment very infrequent though precise and comprehensive visual marker observations with frequent but vague range-only measurements.

When discussing the application, we have seen in the previous chapter that the uncertainty in the estimated map decreases during the progress of time. In [39], they proved this empirical result by showing that the uncertainty indeed monotonically decreases for Kalman Filters. Furthermore, they confirmed theoretically the observations we made without a reference point. In particular, the accuracy of the estimated map is limited by the initial uncertainty in the vehicle's pose at the time the first feature is added to the map. Besides, the absolute alignment of the estimated map may differ from the true environment. But the relative alignment of the map features to each other converges correctly through the correlation of the particular map features depicted in the covariance matrix. Remember, we mentioned such a drift in relation with no reference point, too.

Consider now the spurious measurements observed with the original visual marker detection. One of the challenges is to avoid getting those measurements integrated into the vehicle's state vector, i.e. the estimated map, thus making further estimations of the map and the vehicle's pose in it less error-prone. One solution may be to manage an additional list with already observed features, but which are not considered in the update step yet. This additional list is known under the names *provisional* [1], *tentative* [17] or *potential landmark list* [39]. Consequently, features on this list do not affect the state correction. A simple way in doing this is to zero their deviation from further

measurements in line 5 of algorithm 4.3. As soon as the features have been observed frequently enough, they are fully integrated into the map.

Another challenge in this context is how to treat erroneously incorporated measurements. In [17], a timeout is suggested for this purpose. After incorporating a feature into the map, it needs to be re-observed often enough in a certain amount of time. If this is not the case, it is removed again. Another approach is to consider the ratio of the number of times a feature is observed and the number of times it is not although it should be. Based on this ratio, a feature is removed from the current state vector or not. But how to determine the fact a feature should be recognized? A way in doing this is the usage of probabilistic weights. This approach is named in [1] the *landmark existence probability*. In [39], [40], they call it the *landmark quality*.

Furthermore, in [40], they propose a *Multiple Hypotheses Tracking* (MHT) approach for the data association problem related to Kalman Filters. A well-known problem of the NNSF approach, used for unknown correspondences within this thesis and the corresponding implementation, is namely that wrong associations have a tremendous impact on further EKF-SLAM cycles. This is because they are irreversible. MHT as originally introduced in [41] fights this problem by maintaining multiple tracks of possible data associations during the progress in time. These different tracks are stored in a hypotheses matrix, which can be imagined as a tree. Obviously, a drawback of this approach is its increased demand in computational effort and memory size. Nevertheless, in [42], they introduce an implementation of the MHT algorithm, which is domain independent and, thus, reusable for different areas of applications as well as EKF-SLAM.

6.3 Summary and Outlook

After a recapitulation of basic probabilistic concepts, we derived a theoretical foundation for EKF-SLAM. Based on the underlying theory, algorithms were elaborated solving the problem of creating feature-based maps and simultaneously determining the vehicle's pose in them. These algorithms were then enriched with implementation specific details for EKF-SLAM using visual markers. Furthermore, a measurement noise model for visual markers was statistically evaluated by using an already existing visual marker detection. Finally, the implementation in conjunction with the noise model was shown to be applicable in simulation.

Consequently, in a next step, the provided solution needs to be examined in a real environment. In doing so, the measurement noise model introduced in the previous chapter may be adapted accordingly. Recent tests conducted by Lukas Pfeifhofer revealed that the visual marker detection works better for real markers than in simulation. A possible explanation is that the edges of simulated visual markers are blurred by anti-aliasing mechanism in Gazebo. Unfortunately, explicit edges are crucial for the visual marker detection and, thus, the detection is negatively affected.

Apart from that, the fact that the obtained measurements usually contain 6 DOFs should

be exploited. Thus, the provided solution should be extended from 2D to support 3D. In doing so, the implementation can be adapted at the same time in order to make use of the ROS internal transformation system for the measurement prediction and the integration function. This makes the solution more flexible and less error-prone concerning changes in the setting, e.g. replacing the used visual marker detection.

Finally, dynamic environments can be considered instead of the current static environment assumption. This means in effect, moving visual markers representing e.g. other vehicles performing their own task. This can be achieved by augmenting the present state vector with the first and second deviation of the pose denoting the velocity and acceleration. Thus, not only the pose but also the velocity and acceleration of visual markers are estimated.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Massachusetts: The MIT Press, 2006.

[2] Robot Operating System (ROS), Open Source Robotics Foundation (OSRF), Jul. 2016. [Online]. Available: http://www.ros.org/

[3] Robot Operating System (ROS) Wikipedia, Open Source Robotics Foundation (OSRF), Jul. 2016. [Online]. Available: http://wiki.ros.org/

[4] T. Moore and D. Stouch, "A Generalized Extended Kalman Filter Implementation for the Robot Operating System," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems*, ser. IAS'13. Springer International Publishing, Jul. 2014, pp. 335–348.

[5] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A Flexible and Scalable SLAM System with Full 3D Motion Estimation," in *Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics*, ser. SSRR. IEEE, Nov. 2011.

[6] M. Labbé and F. Michaud, "Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 2661–2666.

[7] ——, "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–745, 2013.

[8] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[9] ——, "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling," in *Proceedings of the IEEE International Conference on Robotics and Automation*, ser. ICRA'05, 2005, pp. 2443–2448.

[10] H. Kato, M. Billinghurst, and I. Poupyrev, *ARToolKit*, 2nd ed., Human Interface Technology Laboratory, University of Washington, Nov. 2000.

[11] D. Wagner and D. Schmalstieg, "ARToolKitPlus for Pose Tracking on Mobile Devices," in *Proceedings of 12th Computer Vision Winter Workshop*, ser. CVWW'07, 2007.

[12] S. Garrido-Jurado, R. M. noz Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, Jun. 2014.

[13] S. Garrido-Jurado, R. M. noz Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, no. C, pp. 481–491, Mar. 2016.

[14] G. Matz and F. Hlawatsch, "Lecture Notes in Signal Processing 2," Vienna University of Technology, 2015.

[15] S. Russel and P. Norvig, *Artificial Intelligence A Modern Approach*, 3rd ed. Upper Saddle River: Prentice Hall, 2003.

[16] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*, 3rd ed. New York: Academic Press, 1991.

[17] S. B. Williams, "Efficient Solutions to Autonomous Mapping and Navigation Problems," Ph.D. dissertation, University of Sydney, Australian Centre for Field Robotics, 2001.

[18] J. Solá, "Simultaneous localization and mapping with the extended Kalman filter," Oct. 2014.

[19] W. A. Bentley and W. J. Humphreys, *Snow Crystals*, 1st ed. New York: Dover Publications, Jun. 1962.

[20] K. G. Libbrecht, "The physics of snow crystals," *Reports on Progress in Physics*, vol. 68, no. 4, p. 855, 2005.

[21] R. E. Kálmán, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, pp. 35–40, 1960.

[22] P. Swerling, "First-Order Error Propagation in a Stagewise Smoothing Procedure for Satellite Observations," *Research Memoranda*, Jun. 1959.

[23] J. B. Fraleigh and R. A. Beauregard, *Linear Algebra*. Boston: Addison-Wesley Publishing Company, 1987.

[24] V. V. Williams, "Multiplying Matrices Faster Than Coppersmith-Winograd," in *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, ser. STOC'12. New York: ACM, 2012, pp. 887–898.

[25] Y. Bar-Shalom, F. Daum, and J. Huang, "The Probabilistic Data Association Filter," *IEEE Control Systems Magazine*, vol. 29, no. 6, pp. 82–100, Dec. 2009.

[26] J. Munkres, "Algorithms for Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, Mar. 1957.

[27] F. Bourgeois and J.-C. Lassalle, "An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices," *Communications of the ACM*, vol. 14, no. 12, pp. 802–804, Dec. 1971.

[28] T. Bailey and H. Durrant-Whyte, "Simultaneous Localization and Mapping: Part II," *IEEE Robotics and Automation Magazine*, pp. 108–117, Sep. 2006.

[29] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robotics and Automation Magazine*, pp. 99–108, Jun. 2006.

[30] R. G. Valenti, I. Dryanovski, and J. Xiao, "Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs," *Sensors*, vol. 15, no. 8, pp. 19 302–19 330, Aug. 2015.

[31] G. Todoran and M. Bader, "Expressive Navigation and Local Path-Planning of Independent Steering Autonomous Systems," in *Proceedings of the International Conference on Intelligent Robots and Systems*, ser. IROS'16, Oct. 2016.

[32] J. Elseberg, D. Borrmann, and A. Nüchter, "Efficient Processing of Large 3D Point Clouds," in *Proceedings of the XXIII International Symposium on Information, Communication and Automation Technologies*, ser. ICAT'11, Oct. 2011, pp. 1–7.

[33] Z. Alsayed, G. Bresson, F. Nashashibi, and A. Verroust-Blondet, "PML-SLAM: a solution for localization in large-scale urban environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, ser. IROS'15, Sep. 2015.

[34] L. M. Paz, J. D. Tardós, and J. Neira, "Divide and Conquer: EKF-SLAM in $\mathcal{O}(n)$," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 1107–1120, Sep. 2008.

[35] Y. Lu, D. Song, and J. Yi, "High Level Landmark-Based Visual Navigation Using Unsupervised Geometric Constraints in Local Bundle Adjustment," in *Proceedings of the IEEE International Conference on Robotics and Automation*, ser. ICRA'14, May 2014, pp. 1540–1545.

[36] N. Chebrolu, D. Marquez-Gamez, and P. Martinet, "Collaborative Visual SLAM Framework for a Multi-Robot System," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, ser. IROS'15, Sep. 2015.

[37] F. R. Fabresse, F. Caballero, I. Maza, and A. Ollero, "Localization and mapping for aerial manipulation based on range-only measurements and visual markers," in *IEEE International Conference on Robotics and Automation*, ser. ICRA'14, May 2014, pp. 2100–2106.

[38] E. Olson, J. J. Leonard, and S. Teller, "Robust Range-Only Beacon Localization," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 949–958, Oct. 2006.

[39] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun. 2001.

[40] D. Makarov and H. F. Durrant-Whyte, "Mobile vehicle navigation in unknown environments: a multiple hypothesis approach," *IEE Proceedings - Control Theory and Applications*, vol. 142, no. 4, pp. 385–400, Jul. 1995.

[41] D. B. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, Dec. 1979.

[42] D. M. Antunes, D. M. de Matos, and J. A. Gaspar, "A Library for Implementing the Multiple Hypothesis Tracking Algorithm," *CoRR*, vol. abs/1106.2263, Jun. 2011.