Rene Hölbling, BSc

Range-Only SLAM utilizing wireless sensor nodes

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr. Klaus Witrisal

Co-Supervisor

Dipl.-Ing. Stefan Josef Grebien, BSc

Institute of Signal Processing and Speech Communication
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Gernot Kubin

Graz, October 2019

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date

Signature

Abstract

Location awareness of mobile agents and the environment is a key component for many Internet of Things applications. This theses proposes a simultaneous localization and mapping algorithm for range-only measurements. The algorithm is designed to handle three-dimensional indoor environments where electronic shelf labels are used as features providing the range-only measurements. The approach is based on the FastSLAM algorithm with a particle filter representation for the initialization of the features. To reduce the computational cost of the particle filter representation methods like KLD-sampling and a likelihood-based approach are implemented and compared. For the evaluation of the algorithm, a simulation environment has been developed in MATLAB. The simulation environment is based on the structure of grocery stores and warehouses. Furthermore, the influences of the different parameters on the results are analyzed. The analysis is used to define requirements on the location system and the quality of the range-only measurements. The simulation results show the applicability and the performance of the algorithm. Using range-only measurements with a standard deviation of 1 m, the algorithm achieves a localization and mapping accuracy of 0.5 m and 0.8 m, respectively.

Kurzfassung

Für viele IoT-Anwendungen ist die Lokalisierung von mobilen Geräten und die Standortbestimmung der Umgebung ein zentraler Bestandteil. Diese Arbeit präsentiert einen Algorithmus für simultane Positionsbestimmung und Kartenerstellung (englisch simultaneous localization and mapping) unter Verwendung von Entfernungsmessungen zur Umgebung. Elektronische Regalplatzetiketten werden als Features verwendet, welche die Entfernungsmessungen zu den mobilen Geräten erzeugen. Das verwendete Konzept basiert auf dem FastSLAM-Algorithmus mit einem 'particle filter' als Lösung für die Initialisierung der Features. Um den Rechenaufwand eines 'particle filters' zu reduzieren, wurden verschiedene Methoden wie KLD-Sampling und ein wahrscheinlichkeitsbasierter Ansatz implementiert und in weiterer Folge verglichen. Zur Evaluierung des Algorithmus wurde eine Simulationsumgebung in MATLAB entwickelt. Die Simulationsumgebung wurde den Anforderungen von Lebensmittelgeschäften und Lagerhallen nachempfunden. Anhand dieser Simulationsumgebung werden die Einflüsse der verschiedenen Parameter auf die Ergebnisse analysiert. Aus dieser Analyse lassen sich Anforderungen an die Lokalisierung und die benötigte Qualität der Entfernungsmessungen definieren. Die Simulationsergebnisse spiegeln die Anwendbarkeit und Performance des Algorithmus wieder. Unter Verwendung von Entfernungsmessungen mit einer Standardabweichung von 1 m, erreicht der Algorithmus eine Genauigkeit von 0.5 m für die Lokalisierung und von 0.8 m für die Standortbestimmung der Features.

Contents

List of Figures

1 Introduction

Advances in modern technologies, like 5G communications systems or radio frequency identification (RFID) systems, make it possible to build high-accuracy indoor localization systems. These are key components [1] of future Internet of Things related applications. Such technologies are ideally suited for enabling improvements in the health care domain, robotics, retail, and many more fields [2].

A recent survey on ambient intelligence in healthcare [3] illustrates a wide range of application. In particular, applications for assisted living include behavioral monitoring to assess the physical and mental health of individuals, emergency detection to alert caretakers or emergency services, and even navigation assistance for visually impaired. For this field of applications a centimeter-accuracy indoor positioning system is needed.
An example for robotics is the European Ubiquitous Networking Robotics in Urban Settings project [4]. In this project, indoor localization is used for the evacuation in case of emergencies, like the outbreak of a fire. The robots lead the people to a secure area via safe pathways.

Another possible improvement is the integration of the technology into electronic shelf labels (ESLs). Consider a mobile agent, which could be a mobile robot or a human customer with a shopping cart, who uses some kind of map to navigate through an environment that contains ESLs. Such shelf labels allow navigation solutions inside a grocery store or a warehouse scenario. Any mobile agent can be navigated to the products they are interested in. In addition, logged customer paths can be used to optimize product placement. This is made possible with the distance measurement between the static shelf labels and a mobile agent.
This thesis focuses on this application scenario and extends the basic idea of localization with simultaneous mapping of the ESL. A simultaneous localization and mapping (SLAM) algorithm is presented which can deal with the range-only (RO) measurements. Using synthetic data, the algorithm is able to achieve decimeter-accurate indoor localization and mapping.
Furthermore, a simulation environment is implemented in order to evaluate the algorithm and analyze the impact of several influencing factors, e.g., the quality of the measurements, the velocity of the mobile agent, or different paths of the mobile agent. This enables the possibility to identify requirements on the ESLs for the purpose of achieving a decimeter-precise localization and mapping.

1.1 Problem statement

In this thesis, grocery store and warehouse scenarios are examined. The basic idea in both scenarios is that an arbitrary mobile agent is able to localize itself and map the positions of the ESLs at the same time. The mobile agent could be a customer with a smartphone, a shopping cart, or a mobile robot. The ESLs provide noisy distance measurements between their positions and the mobile agent's position. These range measurements are essential for the mapping to prevent the localization from diverging. The mobile agent needs an inertial measurement unit (IMU) to keep track of its position, due to the nature of the RO measurements and their associated initialization difficulty for the position of the ESL. Unlike most similar problems, the measurement correspondences are known, i.e., the measurements include an identification which ESL provided the measurement.

The SLAM algorithm should be evaluated in the two different scenarios. A warehouse is used to store goods. Therefore, warehouses usually are large buildings with wide corridors, enabling cranes or forklifts to move goods. A common arrangement is to place the goods on pallets and load them into pallet racks. Hence, it is reasonable to assume that the distribution of shelf labels is not dense. The distance between shelf labels can be in the range of meters.
A grocery store, on the other hand, provides less space. The corridors are smaller because only customers with shopping carts or staff members have to pass through. In order to ensure an efficient use of space, the products are narrowly crammed, leading to a huge number of shelf labels deployed within a small area. The distance between shelf labels can be in the range of centimeters.

1.2 Structure of the thesis

This thesis is structured as follows. Subsequent to the introduction, the theoretical background for the understanding of the thesis is covered in Chapter 2. The focus is put on the probabilistic description of the overall problem. An explanation of the developed simulation environment is given in Chapter 3, including all relevant parameters. Chapter 4 outlines the development of the SLAM algorithm. Special attention is paid to the initialization of the position of the ESLs. The achieved results are presented in Chapter 5. Different influences on the outcome are compared and analyzed. The requirements on the localization system are derived based on this analysis. Finally, in Chapter 6 the findings of this work are summarized and an outlook on possible improvements and further developments is given.

2 Theoretical background

This chapter provides a brief introduction to relevant foundations which are necessary in order to understand the advanced topics discussed later on in this thesis. First of all, the problem statement is generalized to the well known SLAM problem. Then, Bayes filters are introduced to solve this problem. Furthermore, two realizations of this approach are presented, namely the Kalman filter (KF) and the particle filter (PF). Afterwards, the interaction of the IMU and the ESLs in terms of the motion model and the measurement model are outlined. Finally, the Cramér–Rao lower bound (CRLB) is introduced as an instrument to validate potential estimators and to get an impression for the quality of the expected distance measurements.

2.1 Simultaneous localization and mapping

Localization describes the task of determining where a mobile agent is located with respect to a known environment, represented by a map. In this context, mapping is understood as the ability of a mobile agent to construct maps of its environment when knowing its location. The simultaneous localization and mapping (SLAM) problem is the combination of those two problems. Therefore, SLAM asks if a mobile agent can determine its position within an unknown environment and simultaneously generate a consistent map of this environment.
In order to get a mathematical formulation of the SLAM problem, it is necessary to define several quantities. At time instance $t \in \mathbb{N}_0$ the following quantities are defined:

- $\mathbf{p}_t = [p_{\mathrm{x},t}, p_{\mathrm{y},t}, p_{\mathrm{z}}]^T$: the agent position describing the three-dimensional (3D) location. The height p_{z} is assumed to be known and time-invariant.
- $\mathbf{x}_t = [\mathbf{p}_t, v_{\mathrm{x},t}, v_{\mathrm{y},t}, a_{\mathrm{x},t}, a_{\mathrm{y},t}, \theta_t]^T$: the agent state vector consisting of the agent position, velocity, acceleration and yaw angle.
- $\mathbf{f}^{(j)} = [f_{\mathrm{x}}^{(j)}, f_{\mathrm{y}}^{(j)}, f_{\mathrm{z}}^{(j)}]^T$: a vector describing the true 3D position of the j-th feature. The position is assumed to be time invariant.
- $\mathbf{m} = [\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, ..., \mathbf{f}^{(J)}]^T$: the map consisting of all feature vectors.
- $\mathbf{u}_t$: the control vector describing the change of the agent pose.
- $\mathbf{z}_t$: the measurement vector consisting of the received (distance) observations from the features within communication range.

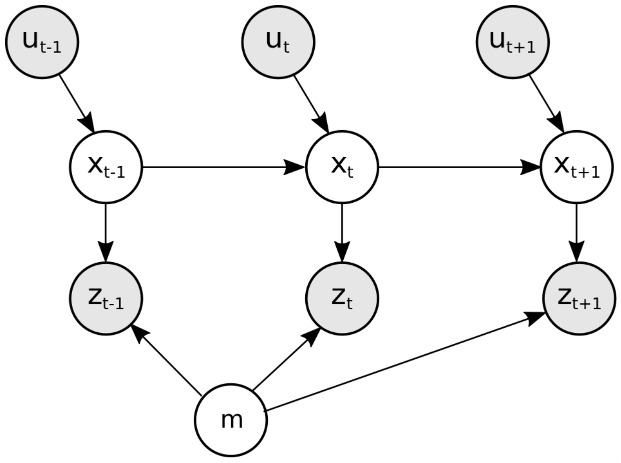


Figure 2.1: Graphical model of the SLAM problem. The shaded nodes are observable by the mobile agent. The white nodes are unobservable and have to be estimated.

With these definitions, the probability density function of the SLAM problem can be written as

$$p(\mathbf{x}_t, \mathbf{m}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{x}_0) \tag{2.1}$$

and has to be estimated for all times t. The graphical model of the SLAM problem is depicted in Figure 2.1.

This topic attracted a lot of attention in research over the last decades. Thus, solutions for many different SLAM variations have been proposed. They can be categorized along various properties. For instance, they can be distinguished by the type of sensors used to perceive the environment or in the way the map is represented. Another important criterion is the way the path is estimated. Full SLAM estimates the whole path in each time step. In contrast to that, the online SLAM estimates only the most recent pose.
Figure 2.2 gives an overview of the resulting taxonomy, without claiming to be thorough and complete. Further information on the taxonomy can be found in [5].

Although there are lots of paradigms, a vast majority of the proposed SLAM algorithms can be derived from three main paradigms. The historically oldest one is called EKF SLAM. It uses recursive Gaussian filter techniques. This paradigm became less popular due to its restrictions representing complex probability distributions with normal distributions. The second one uses graph-based or network-based representations. Graph-based SLAM algorithms use sparse nonlinear optimization to solve the full SLAM problem. An elaborated introduction on graph-based SLAM is given in [6]. The third and last paradigm applies nonparametric filter techniques known as PFs. It is frequently used for the online SLAM problem. One important characteristic of SLAM algorithms that made PF methods popular is the customization to available resources. Thus, the solution is always an approximation taking into consideration requirements of the application. PF methods enable an easy tradeoff between accuracy of the approximation and computational costs. More detailed explanations on the paradigms can be found in [7].

The class of RO online SLAM has a high relevance for this thesis because the problem statement is part of this class. The proposed SLAM algorithms in this class are mainly distinguished by the way they handle the initialization problem arising from the RO measurements. The initialization problem often determines the representation of the features. Most state-of-the-art algorithms are based on the EKF-SLAM.
One of the older approaches [8] extends the EKF-SLAM with a grid-based voting scheme for the localization of the features. Due to the underwater scenario the noisy range measurements include many outliers. For this reason, the voting scheme is constructed to filter outliers and cope with noisy measurements. The voting scheme needs a sufficient dead-reckoning to work. Compared to the grocery store and warehouse scenarios, the number of features is very small. Only four features were used.
In [9] the proposed approach uses a PF for the initialization of the features and then switches back to the EKF representation after the particles have converged. A Gaussian mixture model based on the polar parameterization is used in [10] to represent the features. As a result of the considered independence between all azimuth and elevation angle parameters, the EKF correction step is optimized.
In [11] a sum of Gaussian filters is used. Possible feature locations are represented with Gaussian distributions. The initialization is done according to a geodesic grid with a fixed number of equally spaced Gaussians. A recent approach is based on the sparse extended information filter [12]. This approach has efficiency and scalability advantages compared to the EKF. Additionally, inter-feature measurements are used, which improve map and robot localization accuracies. Furthermore, they are speeding up the feature initialization. Depending on the scenario and the quality of the range measurements, state-of-the-art RO SLAM algorithms achieve absolute mapping errors from 0.35 m to 2 m.
More detailed information on the SLAM problem and its history can be found in [13].

2.2 Bayes filter

Bayes filtering is an approach to estimate an unknown probability density function using measurements $\mathbf{z}_t$ and a process model. In case of the SLAM problem described in Section 2.1, the process model becomes a dynamic motion model with control input $\mathbf{u}_t$. The online SLAM problem requires to compute

$$p(\mathbf{x}_t, \mathbf{m} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{x}_0) \tag{2.2}$$

for all time instances t. Starting with the probabilistic definition, the recursive form of the Bayes filter can be derived. First the Bayes theorem is applied on (2.2), which then

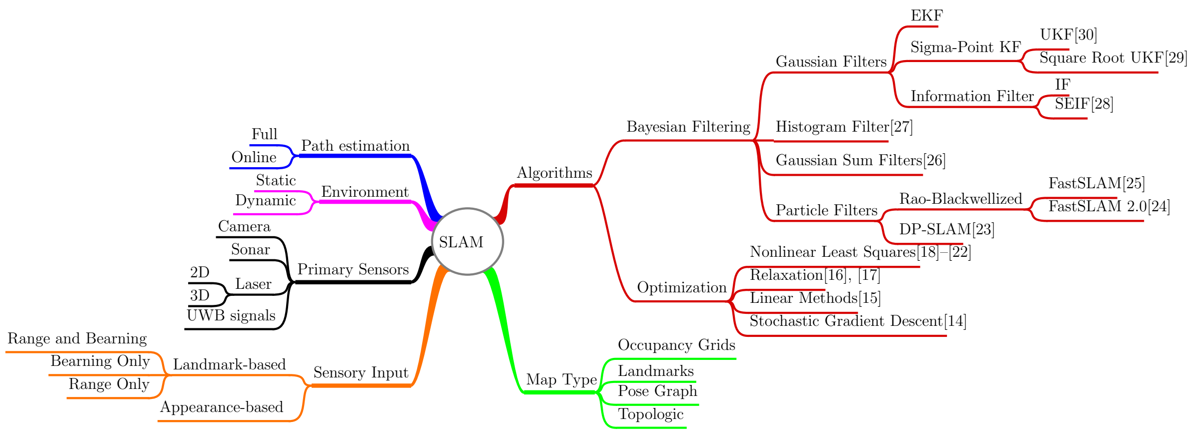


Figure 2.2: A taxonomy of the SLAM problem. Figure adapted from [5].

becomes

$$\frac{p(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m},\mathbf{u}_{1:t},\mathbf{z}_{1:t-1},\mathbf{x}_0)\ p(\mathbf{x}_t,\mathbf{m}|\mathbf{u}_{1:t},\mathbf{z}_{1:t-1},\mathbf{x}_0)}{p(\mathbf{z}_t|\mathbf{u}_{1:t},\mathbf{z}_{1:t-1},\mathbf{x}_0)}, \tag{2.3}$$

where $[p(\mathbf{z}_t|\mathbf{u}_{1:t},\mathbf{z}_{1:t-1},\mathbf{x}_0)]^{-1}$ is a normalization term and has no further influence. For convenience, the normalization term will be called η further on. Next, assuming the Markov property, the expression simplifies to

$$\eta\ p(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m})\ p(\mathbf{x}_t,\mathbf{m}|\mathbf{u}_{1:t},\mathbf{z}_{1:t-1},\mathbf{x}_0). \tag{2.4}$$

The term $p(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m})$ describes the measurement model. The last term is still difficult to interpret. Therefore, the law of total probability is applied to rewrite the expression (2.4) into

$$\eta\ p(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m})\ \int p(\mathbf{x}_t,\mathbf{m}|\mathbf{u}_{1:t},\mathbf{z}_{1:t-1},\mathbf{x}_0,\mathbf{x}_{t-1})p(\mathbf{x}_{t-1},\mathbf{m}|\mathbf{u}_{1:t},\mathbf{z}_{1:t-1},\mathbf{x}_0)d\mathbf{x}_{t-1}. \tag{2.5}$$

Using the Markov assumption one more time, the equation can be written as

$$\eta\ p(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m})\ \int p(\mathbf{x}_t,\mathbf{m}|\mathbf{u}_t,\mathbf{x}_{t-1})\ p(\mathbf{x}_{t-1},\mathbf{m}|\mathbf{u}_{1:t},\mathbf{z}_{1:t-1},\mathbf{x}_0)d\mathbf{x}_{t-1}. \tag{2.6}$$

The classical SLAM problem does not include path planning or active perception. Thus, the current control input $\mathbf{u}_t$ has no influence on $p(\mathbf{x}_{t-1},\mathbf{m}|\mathbf{u}_{1:t},\mathbf{z}_{1:t-1},\mathbf{x}_0)$. The final expression is formulated as

$$\eta\ p(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m})\ \int p(\mathbf{x}_t,\mathbf{m}|\mathbf{u}_t,\mathbf{x}_{t-1})\ p(\mathbf{x}_{t-1},\mathbf{m}|\mathbf{u}_{1:t-1},\mathbf{z}_{1:t-1},\mathbf{x}_0)d\mathbf{x}_{t-1}. \tag{2.7}$$

The term $p(\mathbf{x}_t,\mathbf{m}|\mathbf{u}_t,\mathbf{x}_{t-1})$ corresponds to the state transition probability and is given by the motion model. The term $p(\mathbf{x}_{t-1},\mathbf{m}|\mathbf{u}_{1:t-1},\mathbf{z}_{1:t-1},\mathbf{x}_0)$ corresponds to the posterior of the last time step. Expression (2.7) can be rewritten in order to show the two-step recursive formulation of the Bayes filter. The prediction step is given by

$$p(\mathbf{x}_t, \mathbf{m}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{x}_0) = \int p(\mathbf{x}_t, \mathbf{m}|\mathbf{u}_t, \mathbf{x}_{t-1})\ p(\mathbf{x}_{t-1}, \mathbf{m}|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{x}_0) d\mathbf{x}_{t-1}. \tag{2.8}$$

The correction step is given by

$$p(\mathbf{x}_t, \mathbf{m}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{x}_0) = \eta p(\mathbf{x}_t|\mathbf{z}_t, \mathbf{u}_t) p(\mathbf{x}_t, \mathbf{m}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{x}_0). \tag{2.9}$$

Bayes filters allow for recursive state estimation. The rest of the section is dedicated to realizations of these estimators. Three of them will be explained briefly.

2.2.1 Kalman Filter

The most famous realizations of Bayes filters are Kalman filters (KFs). Introduced in 1960 by Rudolph Kalman [31], the KF is the best studied and most used technique for implementing Bayes filters. The probability distributions are represented by multivariate normal distributions. The definition of a Gaussian normal distribution is stated as

$$p(\mathbf{x}) = \det(2\pi\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\big\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\big\}, \tag{2.10}$$

where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\Sigma}$ is the covariance matrix. One of the fundamental properties of Gaussian distributions is that they are unimodal. This means they possess a single maximum. A random variable with a Gaussian distribution can be parameterized at time instance t by its mean $\boldsymbol{\mu}_t$ and its covariance $\boldsymbol{\Sigma}_t$. The random variable is then denoted as $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. The KF assumes a linear state space model describing the problem. Hence, the state transition probability $p(\mathbf{x}_t, \mathbf{m}|\mathbf{u}_t, \mathbf{x}_{t-1})$ and the measurement probability $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m})$ have to be linear functions in their arguments. The state space model is then expressed by the following equations

$$\mathbf{x}_t = \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t \tag{2.11}$$

$$\mathbf{z}_t = \mathbf{C}_t\mathbf{x}_t + \mathbf{v}_t, \tag{2.12}$$

where $\mathbf{w}_t$ and $\mathbf{v}_t$ are two Gaussian random vectors which are used to model the uncertainties introduced by the state transition and the measurements. They are assumed to be independent of each other and can be denoted as

$$\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t) \tag{2.13}$$

$$\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t). \tag{2.14}$$

Considering all the models and assumptions the state transition probability is given by

$$\begin{aligned} p(\mathbf{x}_t, \mathbf{m}|\mathbf{u}_t, \mathbf{x}_{t-1}) = {} & \det(2\pi\mathbf{Q}_t)^{-\frac{1}{2}} \\ & \exp\big\{-\frac{1}{2}(\mathbf{x}_t - \mathbf{A}_t\mathbf{x}_{t-1} - \mathbf{B}_t\mathbf{u}_t)^T \mathbf{Q}_t^{-1}(\mathbf{x}_t - \mathbf{A}_t\mathbf{x}_{t-1} - \mathbf{B}_t\mathbf{u}_t)\big\}. \end{aligned} \tag{2.15}$$

In the same manner the measurement probability is expressed by the following equation

$$p(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}) = \det(2\pi\mathbf{R}_t)^{-\frac{1}{2}}\exp\Big\{-\frac{1}{2}(\mathbf{z}_t-\mathbf{C}_t\mathbf{x}_t)^T\mathbf{R}_t^{-1}(\mathbf{z}_t-\mathbf{C}_t\mathbf{x}_t)\Big\}. \tag{2.16}$$

Finally, the formulation of the the iterative steps of the KF algorithm at time t can be written as

$$\hat{\boldsymbol{\mu}}_t = \mathbf{A}_t\boldsymbol{\mu}_{t-1} + \mathbf{B}_t\mathbf{u}_t \tag{2.17}$$
$$\hat{\boldsymbol{\Sigma}}_t = \mathbf{A}_t\boldsymbol{\Sigma}_{t-1}\mathbf{A}_t^T + \mathbf{Q}_t \tag{2.18}$$
$$\mathbf{K}_t = \hat{\boldsymbol{\Sigma}}_t\mathbf{C}_t^T(\mathbf{C}_t\hat{\boldsymbol{\Sigma}}_t\mathbf{C}_t^T + \mathbf{R}_t)^{-1} \tag{2.19}$$
$$\boldsymbol{\mu}_t = \hat{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}_t\hat{\boldsymbol{\mu}}_t) \tag{2.20}$$
$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C}_t)\hat{\boldsymbol{\Sigma}}_t. \tag{2.21}$$

Equations (2.17) and (2.18) represent the prediction step mentioned in equation (2.8). Equation (2.17) predicts the state at the next time instance. Equation (2.18) is the propagation of uncertainty from the old estimate to the new predicted state. Equations (2.19), (2.20) and (2.21) correspond to the correction step. Equation (2.19) computes the so-called Kalman gain K_t. Equation (2.20) incorporates the measurement update. Equations (2.19) and (2.20) combined are equivalent to the maximum a posteriori estimator for linear models with Gaussian prior. Equation (2.21) computes the covariance matrix of the estimate.
At time instance t the state is represented by the mean $\boldsymbol{\mu}_t$ and the covariance matrix $\boldsymbol{\Sigma}_t$. The covariance matrix $\boldsymbol{\Sigma}_t$ expresses the uncertainty of the state. KFs work quite well if the problem is linear, roughly Gaussian distributed and unimodal. Further details and references can be found in [32]. Detailed mathematical derivations are given in [33].

2.2.2 Extended Kalman Filter

One limitation of the KF is the linearity of the state transition and the measurement function. This is rarely the case in real world applications. Therefore, a concept is needed that can deal with non-linear functions. This concept is an extension to the KF and is called the extended Kalman filter (EKF).
The state space model of the EKF can be written as

$$\mathbf{x}_t = \mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) + \mathbf{w}_t \tag{2.22}$$
$$\mathbf{z}_t = \mathbf{h}(\mathbf{x_t}) + \mathbf{v}_t, \tag{2.23}$$

with the non-linear state transition function $\mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1})$ and the non-linear measurement function $\mathbf{h}(\mathbf{x_t})$. These non-linear functions are linearized utilizing a first-order Taylor expansion. The general Taylor expansion can be written as

$$\mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) = \sum_{n=0}^{\infty} \frac{\mathbf{g}^{(n)}(\mathbf{u}_t, \mathbf{x}_{t-1})}{n!}\Big|_{\mathbf{x}_{t-1}=\boldsymbol{\mu}_{t-1}} (\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})^n, \tag{2.24}$$

where the nth derivation of $\mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1})$ yields

$$\mathbf{g}^{(n)}(\mathbf{u}_t, \mathbf{x}_{t-1}) = \frac{\partial^{(n)}\mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1})}{\partial \mathbf{x}_{t-1}^{(n)}}. \tag{2.25}$$

For the application of the KF a linear function is required. Therefore, all the higher order terms are neglected. The linearizion is the approximation of a nonlinear function by a linear function that is the tangent hyper-plane to the original function at the mean of the Gaussian. This approximation results in the following approximation

$$\mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) \approx \mathbf{g}(\mathbf{u}_t, \boldsymbol{\mu}_{t-1}) + \underbrace{\mathbf{g}'(\mathbf{u}_t, \boldsymbol{\mu}_{t-1})}_{\mathbf{G}_t}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}). \tag{2.26}$$

$\mathbf{G}_t$ is a matrix of size $m \times n$ with n denoting the dimension of the state and m denoting the dimension of $\mathbf{g}$. The matrix $\mathbf{G}_t$ can be calculated as

$$\mathbf{G}_t = \begin{bmatrix} \left.\frac{\partial g_1(\mathbf{u}_t,\mathbf{x}_{t-1})}{\partial x_1}\right|_{\mathbf{x}_{t-1}=\boldsymbol{\mu}_{t-1}} & \cdots & \left.\frac{\partial g_1(\mathbf{u}_t,\mathbf{x}_{t-1})}{\partial x_n}\right|_{\mathbf{x}_{t-1}=\boldsymbol{\mu}_{t-1}} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial g_m(\mathbf{u}_t,\mathbf{x}_{t-1})}{\partial x_1}\right|_{\mathbf{x}_{t-1}=\boldsymbol{\mu}_{t-1}} & \cdots & \left.\frac{\partial g_m(\mathbf{u}_t,\mathbf{x}_{t-1})}{\partial x_n}\right|_{\mathbf{x}_{t-1}=\boldsymbol{\mu}_{t-1}} \end{bmatrix}. \tag{2.27}$$

$\mathbf{G}_t$ is called the Jacobian matrix of $\mathbf{g}$. If the state transition is linear, $\mathbf{G}_t$ equals $\mathbf{A}_t$. The Jacobian $\mathbf{H}_t$ of the measurement functions is given by

$$\mathbf{H}_t = \begin{bmatrix} \left.\frac{\partial h_1(\mathbf{x}_t)}{\partial x_1}\right|_{\mathbf{x}_t=\boldsymbol{\mu}_t} & \cdots & \left.\frac{\partial h_1(\mathbf{x}_t)}{\partial x_n}\right|_{\mathbf{x}_t=\boldsymbol{\mu}_t} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial h_m(\mathbf{x}_t)}{\partial x_1}\right|_{\mathbf{x}_t=\boldsymbol{\mu}_t} & \cdots & \left.\frac{\partial h_m(\mathbf{x}_t)}{\partial x_n}\right|_{\mathbf{x}_t=\boldsymbol{\mu}_t} \end{bmatrix}. \tag{2.28}$$

Analogous to $\mathbf{G}_t$, the Jacobian $\mathbf{H}_t$ equals $\mathbf{C}_t$ if the measurement function is linear. After the linearizion of the state model, the EKF algorithm can be formulated as

$$\hat{\boldsymbol{\mu}}_t = \mathbf{g}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) \tag{2.29}$$

$$\hat{\boldsymbol{\Sigma}}_t = \mathbf{G}_t\boldsymbol{\Sigma}_{t-1}\mathbf{G}_t^T + \mathbf{Q}_t \tag{2.30}$$

$$\mathbf{K}_t = \hat{\boldsymbol{\Sigma}}_t\mathbf{H}_t^T(\mathbf{H}_t\hat{\boldsymbol{\Sigma}}_t\mathbf{H}_t^T + \mathbf{R}_t)^{-1} \tag{2.31}$$

$$\boldsymbol{\mu}_t = \hat{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{h}(\hat{\boldsymbol{\mu}}_t)) \tag{2.32}$$

$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\hat{\boldsymbol{\Sigma}}_t. \tag{2.33}$$

The equations (2.29) to (2.33) have the same meaning as in the last section. The only difference is the linearized state space model.

2.2.3 Particle Filter

Another limitation of the KF is the restriction to Gaussian distributions. Most real world problems are non-linear and non-Gaussian. Therefore, a different approach may be advantageous for the state representation. Thus, a particle filter (PF) as a sample-based Bayes filter is used. The posterior is represented as a finite number of samples. The big advantage of this approach is that every arbitrary probability distribution can be modeled if enough samples are used. In the case of PFs, the samples of a posterior distribution are called particles and are denoted as

$$\mathcal{X}_t = \{\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, ..., \mathbf{x}_t^{(N)}\} \tag{2.34}$$

The number of particles N that should be used to model a posterior distribution is mainly depending on the complexity of the posterior distribution. Therefore, finding the best number of particles is always a trade-off between accuracy and computational effort. A larger number of particles leads to a more accurate representation but also to higher computational costs.
The basic version has an intuitive explanation. Every particle $\mathbf{x}_{t-1}^{(n)}$ corresponds to a specific realization of the state $\mathbf{x}_{t-1}$. The state if each particle of the next time instance is updated with the state transition function. Afterwards, the particles are rated regarding the measurement model, utilizing the real measurements. In the end, they are resampled according to this rating.

Algorithm 1 Particle filter algorithm

```
1:  procedure PARTICLE_STEP($\mathcal{X}_{t-1}, \mathbf{u}_t, \mathbf{z}_t$)
2:      $\hat{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:      for n = 1 to N do
4:          get $\mathbf{x}_{t-1}^{(n)}$ from $\mathcal{X}_{t-1}$
5:          sample $\mathbf{x}_t^{(n)} \sim p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1}^{(n)})$
6:          $w_t^{(n)} = p(\mathbf{z}_t|\mathbf{x}_t^{(n)})$
7:          add $\mathbf{x}_t^{(n)}$ and $w_t^{(n)}$ to $\hat{\mathcal{X}}_t$
8:      for n = 1 to N do
9:          normalize $w_t^{(n)}$ in $\hat{\mathcal{X}}_t$
10:     for n = 1 to N do
11:         draw $\mathbf{x}_t^{(i)}$ from $\hat{\mathcal{X}}_t$ with probability $\propto w_t^{(n)}$
12:         add $\mathbf{x}_t^{(i)}$ to $\mathcal{X}_t$
13:     return $\mathcal{X}_t$
```

The pseudo code of the most basic version of the PF can be seen in Algorithm 1. The steps of the intuitive explanation are directly seen in the pseudo code. The first for-loop (line 3-7) applies the motion model to each particle. Furthermore, a weight w_t^m is calculated

from the measurement model. The weight rates the particles according to the measurement. The second for-loop (line 8-9) normalizes all the weights. This has to be done to get a valid probability distribution with a probability mass of 1. The last for-loop (line 10-12) describes the resample procedure.
One has to be aware that only the basic version of the PF is discussed in this section. There have been many improvements and extensions to this algorithm over the years. Most of them try to make the PF computationally more efficient.
Good tutorials on the PF are [34] and [35]. These tutorials give a more elaborated instruction and further references.

2.3 Inertial measurement unit

Next, a deeper look is taken into the motion model. Regarding the problem statement, no specifications have been made for the mobile agent. As a consequence, no prior knowledge can be incorporated into the motion model. Therefore, an electronic sensor system is required that measures all the physical quantities needed to estimate the motion of the mobile agent. Such a sensor system is called inertial measurement unit (IMU). An IMU typically consists of accelerometers and gyroscopes to measure the acceleration and the angular velocity.
Some IMUs additionally include magnetometers to determine the magnetic field strength in a given direction. The magnetic field strength in the x- and y-direction are used to calculate the orientation. This gives an absolute orientation with respect to the Earth's magnetic North. This is especially useful for navigation problems. Hence, this thesis simulates an IMU with included magnetometer.

IMUs exist in two system configurations. The first configuration are stable platform type systems. Stable platform type systems are mounted isolated from the external rotational motion. This can be done using gimbals and torque motors. The gyroscopes signals are fed back to the torque motors which rotate the gimbals to keep the platform always in the same orientation. The orientation measurement can be read from the angles between adjacent gimbals using angle pick-offs. As a result, the acceleration is in the global frame and can be integrated twice to get the position. The stable platform inertial navigation algorithm is visualized in Figure 2.3.
The second configuration are strapdown type systems. In this kind of systems, the inertial sensors are mounted rigidly on the mobile agent. Thus, all the physical quantities are measured in the agent frame and not in the global frame. Hence, to keep track of the orientation, the gyroscope signals are integrated. A magnetometer helps with the initialization of the orientation and the correction of the gyroscope signals. Using the known orientation, the accelerometer signals can be transformed into the global frame. The global acceleration signals are then integrated in the same manner as in the stable platform algorithm. Strapdown type systems are computationally more demanding but mechanically

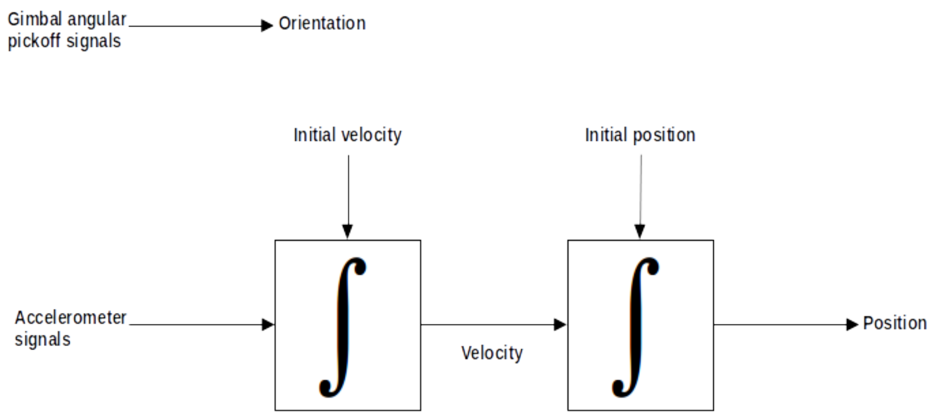


Figure 2.3: Stable platform inertial navigation algorithm. Figure adapted from [36].

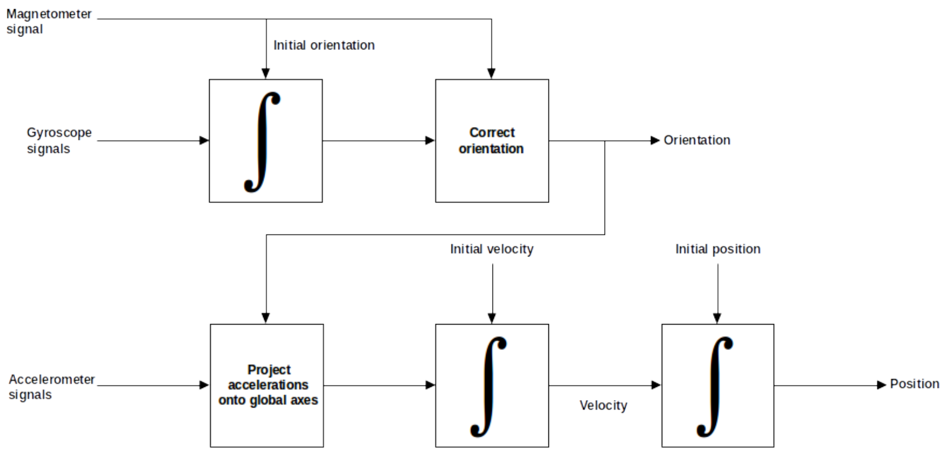


Figure 2.4: Strapdown inertial navigation algorithm. Figure adapted from [36].

less complex. Thus, they tend to be physically smaller and cheaper than stable platform type systems. For these reasons, strapdown type systems are more feasible for an arbitrary mobile agent. Therefore, all further IMUs are considered to be strapdown type system. The strapdown inertial navigation algorithm is shown in Figure 2.4.

Unfortunately, measurements from IMUs are error prone. The main error sources for the accelerometers and gyroscopes are identified as misalignment, scale factor, noise, and bias. The bias itself consists of a static and a drifting bias. With a calibration procedure, the misalignment, the scale factor, and the deterministic part of the bias can be corrected. Assuming a calibrated IMU, only the noise and the drifting bias are considered as error sources. The biggest issue with magnetometers is that they are adversely affected by other magnetic fields. As this is difficult to simulate and not within the scope of this work, only noise is considered for the magnetometer.

Table 2.1: IMU grades. Reprinted from [37].

	Perfomance Parameters	Consumer	Automotive	Tactical	Navigation I	Navigation II
Accelerometer	Bias stability (μg)	2400	1200	200-500	50-100	5-10
	Scale Factor (PPM)	-	>1000	400-1000	100-200	10-20
	Noise density VRW($\mu g/\sqrt{\text{Hz}}$)	1000	1000	200-400	50	5-10
Gyroscope	Bias stability ($°/h$)	>200	10-200	1-10	0.1-1.0	<0.01
	Scale Factor (PPM)	-	>500	200-500	100-200	5-50
	Noise density ARW ($°/\sqrt{h}$) ($°/h/\sqrt{\text{Hz}}$)	3 180	3 180	0.2-0.5 12-30	0.05-0.2 3-12	0.002-0.005 0.12-0.3

The most common classification of IMUs is by their performance, not by their technology. For the classification only accelerometers and gyroscopes are looked at. An interesting fact regarding the classification of IMUs is that in many datasheets the performance of the gyroscope is prioritized over the performance of the accelerometer. In Table 2.1, the classification is presented with respect to typical associated performance parameters. The presented classification should give a feeling for the capabilities of IMUs associated with a certain grade term. For further information about IMUs see [36].

2.4 Smart electronic shelf labels

The emergence of online grocery shopping comes with a lot of advantages for the customers like low prices and a wide product range. Hence, stationary retailers are required to adapt their retail strategies. More and more grocery stores migrate from paper labels or plastic labels to electronic shelf labels (ESLs). A recent study [38] shows that customers perceive ESLs easy to use but are mostly unaware of their benefits. This creates the interest to improve the functionality and make it clearer to recognize its benefits for the customer. Hence, one of the next steps to make ESLs even smarter is to incorporate indoor localization technology. Such ESLs do not only display the price of the product and useful information but also know their position and help a mobile agent with its localization.
Indoor localization witnessed a lot of interest lately, due to the services it can provide for the Internet of Things (IoT). Therefore, many different techniques such as Angle of Arrival,

Time of Flight, Return Time of Flight, and Received Signal Strength based on technologies such as WiFi, RFID, ultra wideband (UWB), and Bluetooth have been developed. A good summary on indoor localization in general can be found in [39]. Furthermore, a lot of systems utilizing these techniques and technologies have been proposed. A comparison of different state-of-the-art systems is given in [40]. Although these systems are developed for localization, the same technology could be used with small adaptations to solve the SLAM problem. Because of the high numbers of ESLs in a grocery store, there are special requirements on the system. It needs to be easily deployable, cheap and with a low energy consumption. Thus, the most applicable technologies for ESLs are either UWB or RFID. For convenience and to be consistent with the relevant literature, the ESLs will be called features further on in the context of the SLAM problem.
One approach for such a system is to extract position-related parameters from the measured signals and estimate the position in a second step. In our case the position-related parameter is the distance from the feature to the mobile agent. The focus in the thesis is on the second step. Hence, it does not matter which localization techniques or technologies are used. The quality of the estimator specifies the probabilistic description of the measurement model of the mobile agent.

2.5 Cramér–Rao lower bound

Before the quality of estimators can be discussed, the concept of estimators has to be introduced. In general an estimator is a function that maps a set of observations to a set of estimates. Henceforth, an estimator will be denoted as

$$\hat{\zeta} = g(\mathbf{z}). \tag{2.35}$$

Two famous examples of estimators are the maximum likelihood and the maximum a posteriori estimator. They can be formulated as

$$\hat{\zeta}_{ML} = \arg \max_{\zeta}(p(\mathbf{z};\zeta)) \tag{2.36}$$

$$\hat{\zeta}_{MAP} = \arg \max_{\zeta}(p(\zeta|\mathbf{z})) = \arg \max_{\zeta} \left(\frac{p(\mathbf{z}|\zeta)p(\zeta)}{p(\mathbf{z})} \right). \tag{2.37}$$

An estimator determines the unknown parameters ζ using the observations z. The unknown parameters ζ could be the distance, the observations $\mathbf{z}$ could be either one of the techniques such as a Time of Flight or Received Signal Strength measurements. As two possible measures for the performance of estimators, two statistical properties are used: the bias and the variance. The bias and the variance are given by

$$\text{bias}(\hat{\zeta}) = E(\hat{\zeta}) - \zeta \overset{unbiased}{=} 0 \tag{2.38}$$

$$\text{var}(\hat{\zeta}) = E((\hat{\zeta} - E(\hat{\zeta}))^2). \tag{2.39}$$

The bias is defined as $E(\hat{\zeta}) - \zeta$. The bias can be understood as the deviation of the expected value of the estimator $\hat{\zeta}$ from the real value of the parameter ζ. An estimator is called unbiased if the bias is zero. The variance is an indicator of how far, on average, the produced estimates are from the expected value of the estimates. For an unbiased estimator, the variance equals the mean squared error.

The preferable estimator is called minimum-variance unbiased (MVU) estimator. It is unbiased and has the smallest possible variance of all unbiased estimator.
Even if the MVU estimator does not exist for a certain problem, a lower bound on its variance can be found. A possibility to derive such a lower bound is the Cramér–Rao lower bound (CRLB). This is often useful as a benchmark to rate the results of an estimator. Additionally, the theory of CRLB is capable to determine if an estimator exists that attains this limit.
The CRLB for scalar estimator is given by

$$\operatorname{var}(\hat{\zeta}) \geq \frac{1}{-E\left(\frac{\partial^2 \ln(p(\mathbf{z};\zeta))}{\partial \zeta^2}\right)}, \tag{2.40}$$

where the derivative is evaluated at the true value of the parameter ζ and the expected value is taken with respect to the likelihood function. Because the CRLB holds for any unbiased estimator $\hat{\zeta}$, the probability density function $p(\mathbf{z};\zeta)$ has to satisfy the regularity condition

$$E\left(\frac{\partial \ln(p(\mathbf{z};\zeta))}{\partial \zeta}\right) = 0, \ \forall \zeta. \tag{2.41}$$

An often mentioned term related to the CRLB is the Fisher information (FI). The FI is the inverse of the CRLB. The FI is explicitly given by

$$I(\zeta) = -E\left(\frac{\partial^2 \ln(p(\mathbf{z};\zeta))}{\partial \zeta^2}\right) = -\int \frac{\partial^2 \ln(p(\mathbf{z};\zeta))}{\partial \zeta^2} p(\mathbf{z};\zeta) d\mathbf{z}. \tag{2.42}$$

The FI is used to conveniently express the condition for the MVU estimator as

$$\frac{\partial \ln(p(\mathbf{z};\zeta))}{\partial \zeta} = I(\zeta)(\hat{\zeta} - \zeta). \tag{2.43}$$

If and only if this condition holds for all ζ the estimator $\hat{\zeta} = g(\mathbf{z})$ is an MVU estimator.
For the extension to the vector case, also the FI is used. The vector parameter CRLB is defined as the elements of the main diagonal of the inverse FI matrix

$$var(\hat{\zeta}_i) \geq [\mathbf{I}^{-1}(\hat{\boldsymbol{\zeta}})]_{ii}. \tag{2.44}$$

The FI matrix is defined as

$$[\mathbf{I}(\hat{\boldsymbol{\zeta}})]_{ij} = -E\left(\frac{\partial^2 \ln(p(\mathbf{z};\hat{\boldsymbol{\zeta}}))}{\partial \zeta_i \partial \zeta_j}\right). \tag{2.45}$$

A detailed description of the CRLB can be found in [41] including many examples.

3 Simulation environment

In the last chapter the basis for understanding the thesis were covered. Most topics of the thesis will be based on this theoretical background. This chapter deals with the simulation environment. The simulation environment is used to test the developed SLAM algorithm in a setting close to real-world conditions. All the implementations were done in the programming language MATLAB.

3.1 Mobile agent

The mobile agent is deliberately formulated in a broad way. The SLAM algorithm can be applied to various applications. The mobile agent can take many different forms ranging from a mobile robot with a sophisticated localization sensor system to a shopping cart with a simplistic localization sensor system. The use case for the mobile robot is a decent initialization of the map within a mostly unknown environment. In comparison, a shopping cart as a mobile agent makes sense only with a map that is at least partially initialized. The main focus in this case will be on the localization and the mapping will be fine-tuned.

Nonetheless, all mobile agents have in common that they have to be equipped with a communication system that can receive signals from the features. To model which features are in reach, a parameter called detection radius r_{det} is defined. This parameter determines in which radius, regarding its current position, the mobile agent can receive signals from the features. Typically, the position of the features is unknown to the mobile agent but for some of the features, the position is known beforehand. This kind of features are called anchors. These features with known position are important to the location part of the SLAM algorithm.

The most important characteristic of the mobile agent is the odometry. Odometry can be understood as the use of data from motion sensors to estimate the change in position over time. Hence, it has a direct influence on the performance of the whole SLAM algorithm. There are various ways to determine the odometry, for instance a rotary encoder on a robot wheel. A rotary encoder can measure how far the wheels have rotated. With the combination of the circumference of its wheels and the distance between the wheels the relative motion can be computed. To keep the mobile agent as versatile as possible and

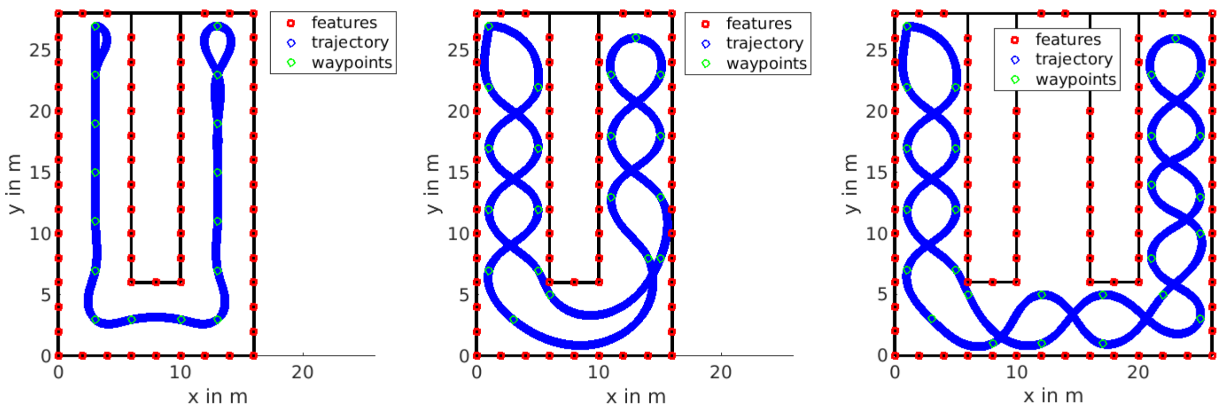


Figure 3.1: Example trajectories for the warehouse scenario.

not rely on specific mobile sensors, the only requirement for the mobile agent is to have an IMU.

3.1.1 Calculation of the trajectory

For the description of the mobile agent pose, the position and orientation is sufficient. For the simulation including the IMU, more than just the covered path is necessary. The full description consists of the position, orientation, velocity, acceleration, and angular velocity of the mobile agent at all times t.
The actual calculation of the trajectory uses the MATLAB-function `waypointTrajectory`. The function uses a set of 3D waypoints, the times at which these waypoints should be reached, and the sample frequency to calculate the necessary quantities for the full description. As already mentioned in the problem statement, the trajectories are restricted to two-dimensional (2D) movement. Therefore the z-component of the 3D waypoints is fixed. Examples of trajectories can be seen in Figure 3.1.

3.1.2 Modeling of the inertial measurement unit

In Section 2.3 the main error sources of IMUs were explained. A matching error model is found in [37] and can be formulated as

$$\mathbf{a}_{\mathrm{m},t} = \mathbf{a}_t + \delta\mathbf{a} + \mathbf{a}_{\mathrm{vrw}} \tag{3.1}$$

$$\omega_{\mathrm{m},t} = \omega_{\mathrm{yaw},t} + \delta\omega + \omega_{\mathrm{arw}}, \tag{3.2}$$

where $\mathbf{a}_t$ and $\omega_{\mathrm{yaw},t}$ denote the physical quantities of acceleration and the angle velocity in yaw direction at time instance t. The error sources which have to be simulated are the drifting biases $\delta\mathbf{a}$ and $\delta\omega$ and the noises $\mathbf{a}_{\mathrm{vrw}}$ and ω_{arw}. The drifting bias for the accelerometer and gyroscope are modeled as a first-order Gauss-Markov process. The same model is used in [42] and is defined as

$$\delta\mathbf{a}' = -\frac{1}{\tau_{\mathrm{a}}}\delta\mathbf{a} + \mathbf{w}_{\mathrm{a}_{\mathrm{bias}}} \tag{3.3}$$

$$\mathbf{w}_{\mathrm{a}_{\mathrm{bias}}} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathrm{a}_{\mathrm{bias}}}) \tag{3.4}$$

$$\delta\omega' = -\frac{1}{\tau_{\omega}}\delta\omega + w_{\omega_{\mathrm{bias}}} \tag{3.5}$$

$$w_{\omega_{\mathrm{bias}}} \sim \mathcal{N}(0, \sigma^2_{\omega_{\mathrm{bias}}}). \tag{3.6}$$

The values for the variances can be found in the datasheet of the IMU under the name of bias stability or bias instability. If there is no such entry, the variances can be calculated using Allan deviation [43]. The noise is assumed to be normally distributed with

$$\mathbf{a}_{\mathrm{vrw}} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathrm{vrw}}) \tag{3.7}$$

$$\omega_{\mathrm{arw}} \sim \mathcal{N}(0, \sigma^2_{\mathrm{arw}}). \tag{3.8}$$

In terms of the accelerometer this noise is called velocity random walk and for the gyroscope the noise is called angle random walk. These two parameter can also be found in the datasheet. Regarding the magnetometer only the noise is simulated as

$$\mathbf{h}_{\mathrm{m},t} = \mathbf{h}_{\mathbf{t}} + \mathbf{h}_{\mathrm{noise}}. \tag{3.9}$$

Again a normal distribution is assumed for the noise with

$$\mathbf{h}_{\mathrm{noise}} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{h_{\mathrm{noise}}}). \tag{3.10}$$

Knowing the horizontal component of the Earth's magnetic field and the local declination D [44], the orientation can be derived as follows

$$\theta_{\mathrm{m},t} = \tan^{-1}\left(\frac{h_{m,\mathrm{x},t}}{h_{m,\mathrm{y},t}}\right) \pm D. \tag{3.11}$$

In Table 3.1 the parameters for the simulation of the IMUs can be seen. In comparison to that, the inertial sensor of the iPhone 4 are listed in Table 3.2. With the cheaper inertial sensors the drifting bias gets problematic. While the noise is in the same range, the bias error is higher. The same magnetometer is used for all the IMUs .

3.2 Scenario setting

As mentioned in Section 1.1 this thesis will focus on two scenarios: a grocery store and a warehouse facility scenario. This section will take a deeper look at the two different settings and the way they are modeled. Many requirements which were already mentioned in the

Table 3.1: Parameters of the used IMUs.

	IMU used in [42]		MTi 1 series [45]		
Parameter	VTI SCA3000	ADI ADXRS150	Acc.	Gyro.	Mag.
Bias stability	100 μg	36 °/h	30 μg	10 °/h	-
Noise density	450 $\mu g/\sqrt{\text{Hz}}$	180 °/h/$\sqrt{\text{Hz}}$	120 $\mu g/\sqrt{\text{Hz}}$	25 °/h/$\sqrt{\text{Hz}}$	0.5 $\frac{\text{mG}}{\sqrt{\text{Hz}}}$

Table 3.2: Inertial sensors used in the iPhone 4 [46].

	STMicroelectronics	
Parameter	L3G4200D	L1S331DLH
Bias stability	75 °/s	20 mg
Noise density	108 °/h/$\sqrt{\text{Hz}}$	218 $\mu g/\sqrt{\text{Hz}}$

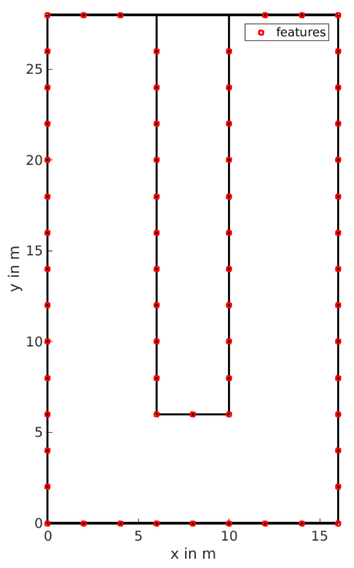
problem statement will be revisited.
The two scenarios can be seen in Figure 3.2. To keep the simulations comparable, both scenarios have similar layouts with two shelves. There are two basic parameters that distinguish the scenarios: the spatial size of the room with their shelves and the density of the features. A higher density means that the distance between two features is smaller and vice versa. The features are equally spaced. In addition, the different features have three possible heights, namely $z_1 = 0.5\,\text{m}$, $z_2 = 1\,\text{m}$, or $z_3 = 1.5\,\text{m}$. In the generation of the scenario, they are placed randomly at one of those heights.
The warehouse facility is characterized by corridors with a width of 5 m, a distance between the features of 2 m. In contrast, the grocery store scenario has smaller corridors of 2 m but a shorter distance between the features of 0.3 m. A real grocery store has an even higher density of features. A shorter distance between features of 0.1 m would be realistic. The chosen distance can be explained with the computability of the SLAM algorithm and the possibility of multiple runs using just a subset of all the features.

3.2.1 Simulation of the shelf labels

The idea of the distance estimator is that the mobile agent receives a signal from the feature in reach. To calculate the distance from this received signal, the signal has to be modeled as a convolution of the transmit pulse $s(t)$ with the channel. Similar models as well as the resulting CRLBs of the distance estimator are found in [41], [47]. The CRLB of the distance estimator is expressed by

$$\text{var}(d) \geq \frac{c^2}{8\pi^2\ \text{SNR}\ \beta^2}, \tag{3.12}$$

where $\beta^2 = \frac{\int_f f^2 |S(f)|^2 df}{\int_f |S(f)|^2 df}$ is the mean square bandwidth of the Fourier transform $S(f)$ of the transmitted pulse $s(t)$. For a block spectrum $|S(f)|^2 = \frac{1}{B}$ for $|f| \leq \frac{B}{2}$ with the bandwidth



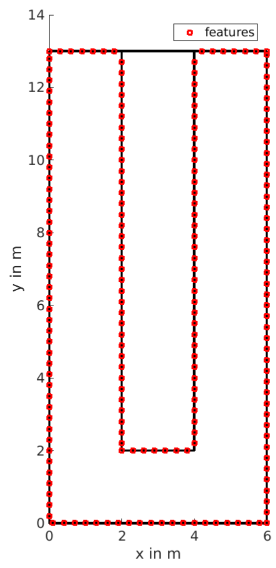


Figure 3.2: Visualization of the warehouse facility and grocery store scenario.

B, the mean square bandwidth can be calculated as

$$\beta^2 = \frac{\int_f f^2 |S(f)|^2 df}{\int_f |S(f)|^2 df} = \frac{\displaystyle\int_{-\frac{B}{2}}^{\frac{B}{2}} \frac{1}{B} f^2 df}{\displaystyle\int_{-\frac{B}{2}}^{\frac{B}{2}} \frac{1}{B} df} = \frac{B^2}{12}. \tag{3.13}$$

To get an assessment for the quality of the distance measurement, two bandwidths $B_1 = 20\,\mathrm{MHz}$ and $B_2 = 200\,\mathrm{MHz}$ are considered to calculate the CRLBs. Two bandwidths are used to analyze the effect of different qualities of measurements. With an assumed signal-to-noise ratio (SNR) of 30 dB the CRLBs can be calculated as

$$\mathrm{var}(d_1) \geq \frac{c^2}{8\pi^2\ \mathrm{SNR}\ \beta_1^2} \approx (0.58\,\mathrm{m})^2 \tag{3.14}$$

$$\mathrm{var}(d_2) \geq \frac{c^2}{8\pi^2\ \mathrm{SNR}\ \beta_2^2} \approx (0.058\,\mathrm{m})^2. \tag{3.15}$$

For the two discussed cases, minimal variances of approximately $(0.58\,\mathrm{m})^2$ and $(0.058\,\mathrm{m})^2$ are achieved. This is the optimal case for the MVU estimator. It is very unlikely to achieve this results, thus a more realistic value for the variances of $(1\,\mathrm{m})^2$ and $(0.1\,\mathrm{m})^2$ is expected. With this estimations the simulated measurements can be formulated. The distance for the ith feature will be calculated according to the following equation

$$d(\mathbf{p}_t, \mathbf{f}^{(j)}) = ||\mathbf{p}_t - \mathbf{f}^{(j)}||_2 = \sqrt{(p_{\mathrm{x},t} - f_{\mathrm{x}}^{(j)})^2 + (p_{\mathrm{y},t} - f_{\mathrm{y}}^{(j)})^2 + (p_{\mathrm{z}} - f_{\mathrm{z}}^{(j)})^2}. \tag{3.16}$$

The final measurement will be generated as

$$z_t^{(j)} = \begin{cases} \mathcal{N}(d(\mathbf{p}_t, \mathbf{f}^{(j)}), \mathrm{var}(d)), & \text{if } d(\mathbf{p}_t, \mathbf{f}^{(j)}) \leq r_{\mathrm{det}} \\ \emptyset, & \text{otherwise} \end{cases}, \tag{3.17}$$

with $\emptyset$ as the empty set.
For the simulation of the measurements, additive white Gaussian noise with the two estimated variances is used. This measurement model may not model precisely the real-world measurements. For example, a possible bias of the observations due to multipath is not modeled at all. If a communication system is built, the assumed theoretical measurement model has to be replaced with the measured model.

3.3 Simulation of the passage of time

The IMU as well as the features have a number of measurements they provide within a certain time interval. The frequency parameters f_{IMU} and f_{features} are introduced to model this behavior. Even under the assumption that the mobile agent is synchronized with the features and the features are synchronized with themselves it is still unlikely that they operate on the same frequency. Hence, it can be assumed that the features provide measurements at a different frequency than the IMU. Due to the energy efficiency of the features, it is more likely that the IMU runs on a higher frequency than the features. Therefore, the minimal time step Δt is defined as

$$\Delta t = \frac{1}{f_{\mathrm{IMU}}}. \tag{3.18}$$

This implies the time t in the simulation can only advance in this discrete time steps and have to be a whole multiple of it

$$t = n \cdot \Delta t, \ n \in \mathbb{N}. \tag{3.19}$$

To ensure that in every time step an IMU measurement is available the frequencies need to satisfy

$$\mathrm{mod}\,(f_{\mathrm{IMU}}, f_{\mathrm{features}}) = 0. \tag{3.20}$$

By combining this requirements with the assumption that

$$f_{\mathrm{IMU}} \gg f_{\mathrm{features}}, \tag{3.21}$$

the feature measurements can be written as

$$z_t = \begin{cases} z_t^{(j)}, & \text{if } z_t^{(j)} \neq \emptyset \wedge \mathrm{mod}\left(t, \frac{1}{f_{\mathrm{features}}}\right) = 0 \\ \emptyset, & \text{otherwise} \end{cases} \tag{3.22}$$

$$\mathbf{z}_t = \{z_t^{(j)} | j \in \mathbb{N} \wedge j \leq J\}. \tag{3.23}$$

4 Algorithm

The previous chapter was dedicated to introduce the simulation environment in which the SLAM algorithm is tested. This chapter deals with the development of the proposed RO SLAM. A state of the art SLAM algorithm was implemented as a starting point. FastSLAM [25] was chosen since it is a widely used SLAM algorithm. The basic version of the FastSLAM has to be adapted to the simulation environment. Afterwards, the adapted version of FastSLAM is extended to handle the initialization problem arising from the RO measurements. To overcome the initialization problem, a PF is applied. Furthermore, methods to decrease the computing effort of the PF are added to the extended FastSLAM. Finally, a detection method is introduced for outliers.

4.1 Overview

The proposed SLAM algorithm is based on the FastSLAM algorithm, explained in Section 4.2. As a starting point, the FastSLAM algorithm is adapted to our environment. For this purpose, an EKF filter is designed for the motion model using the IMU measurements in Section 4.3.1. For the feature representation, an EKF filter is designed in Section 4.2.1. This EKF filter incorporates the measurement model. The naive adaptation of the FastSLAM leads to the initialization problem of the EKF with RO measurement. To overcome the initialization problem, a PF representation is introduced in Section 4.3.

An overview of the operation of the extended FastSLAM algorithm is given in form of a flowchart in Figure 4.1. With the assumption of a known start position the initialization of the agent position is straightforward. The first measurement for each feature initializes the associated feature PF. Section 4.3.2 gives a detailed description about the initialization procedure. For each measurement, the corresponding weights are calculated in the agent PF. Subsequently to the calculation of all the weights, the weights will be normalized within their affiliated PF. The normalized weights are then used to resample the agent and the feature PF. Further details on the resampling process are given in Section 4.3.3. The functionality of the resampling process is extended with two optional approaches to reduce the computational effort. The first approach employs methods to reduce the number of particles and is described in Section 4.3.5. The second approach switches back to the EKF representation after the feature PF has converged, hence the initialization problem has been solved. A detailed explanation is given in Section 4.3.6. Finally, the outlier detection

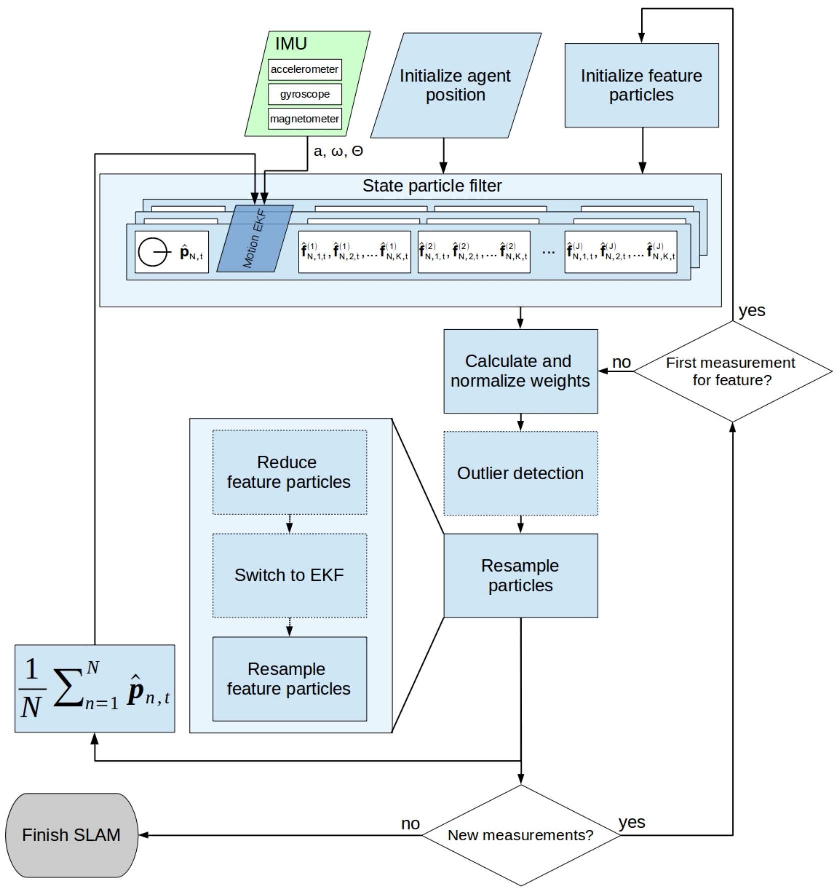


Figure 4.1: Flowchart of the extended FastSLAM algorithm. The agent position is denoted by $\hat{\mathbf{p}}_{n,t}$ for the n-th particle in the state particle filter. The algorithm represents each features with a PF. One feature particle is denoted as $\hat{\mathbf{f}}^{(j)}_{n,k,t}$ where n denotes the index of the agent particle, j denotes the index of the feature, and k denotes the index of the particle.

is described in Section 4.3.7. It can be used as an optional step between the normalization of the weights and the resampling procedure.

4.2 FastSLAM

The individual feature measurements are assumed to be independent of each other. For a mapping-only problem, the determination of the feature locations can be divided into independent estimation problems, one for each feature. Based on this idea, FastSLAM decomposes the SLAM problem into a mobile agent localization problem, and a collection of feature estimation problems that are conditioned on the agent pose estimate. FastSLAM uses a PF for estimating the agent pose over the path of the mobile agent. The PF is

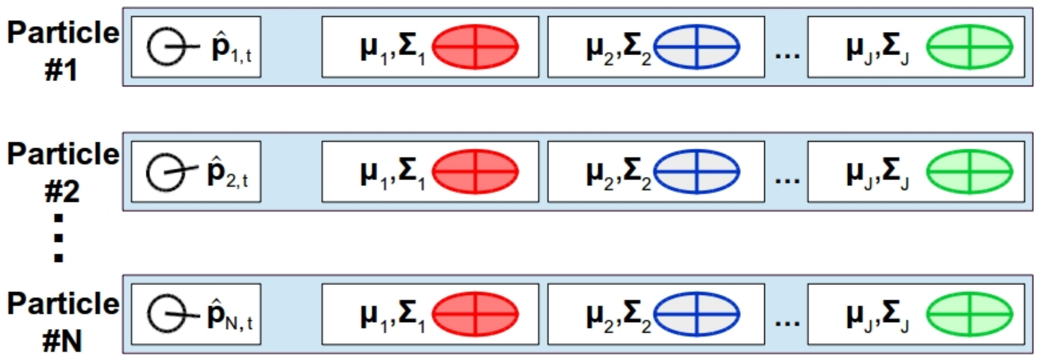

Particle #1
Particle #2
Particle #N
$\hat{\mathbf{p}}_{1,t}$
$\hat{\mathbf{p}}_{2,t}$
$\hat{\mathbf{p}}_{N,t}$
$\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1$
$\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2$
...
$\boldsymbol{\mu}_J,\boldsymbol{\Sigma}_J$

Figure 4.2: Representation of the individual particles in the FastSLAM algorithm.

composed of N different particles. Each particle uses J EKFs to estimate the feature locations, depending on the path estimate. Hence, each feature is represented by N EKFs with 3D state vectors. The representation of the FastSLAM algorithm can be seen in Figure 4.2.

4.2.1 Extended Kalman filter for the measurements

The representation of the FastSLAM algorithm requires an EKF with 3D state vectors for each feature. In this special case where the EKF is used to estimate the positions of static features, no transition function can be applied. Therefore, the prediction step is not needed. The measurement function was already stated in equation (3.16). For convenience the measurement function is restated here

$$h_t^{(j)}(\mathbf{p}_t, \mathbf{f}^{(j)}) = d(\mathbf{p}_t, \mathbf{f}^{(j)}) = \sqrt{(p_{\mathrm{x},t} - f_{\mathrm{x}}^{(j)})^2 + (p_{\mathrm{y},t} - f_{\mathrm{y}}^{(j)})^2 + (p_{\mathrm{z}} - f_{\mathrm{z}}^{(j)})^2}. \tag{4.1}$$

From the measurement function, the Jacobian matrix can be derived

$$\begin{aligned}\mathbf{H}_t^{(j)} &= \left[\frac{\partial h_t^{(j)}(\mathbf{p}_{n,t},\boldsymbol{\mu}_t^{(j)})}{\partial f_{\mathrm{x}}^{(j)}}, \quad \frac{\partial h_t^{(j)}(\mathbf{p}_{n,t},\boldsymbol{\mu}_t^{(j)})}{\partial f_{\mathrm{y}}^{(j)}}, \quad \frac{\partial h_t^{(j)}(\mathbf{p}_{n,t},\boldsymbol{\mu}_t^{(j)})}{\partial f_{\mathrm{z}}^{(j)}}\right] &(4.2)\\ &= \left[\frac{\mu_{\mathrm{x},t}^{(j)} - p_{\mathrm{x},n,t}}{d(\mathbf{p}_{n,t},\boldsymbol{\mu}_t^{(j)})}, \quad \frac{\mu_{\mathrm{y},t}^{(j)} - p_{\mathrm{y},n,t}}{d(\mathbf{p}_{n,t},\boldsymbol{\mu}_t^{(j)})}, \quad \frac{\mu_{\mathrm{z},t}^{(j)} - p_{\mathrm{z},n,t}}{d(\mathbf{p}_{n,t},\boldsymbol{\mu}_t^{(j)})}\right]. &(4.3)\end{aligned}$$

With the Jacobian matrix of the measurement function, the correction step for the used EKFs can be formulated as

$$\begin{aligned}\mathbf{K}_t^{(j)} &= \boldsymbol{\Sigma}_{t-1}^{(j)}(\mathbf{H}_t^{(j)})^T \left(\mathbf{H}_t^{(j)}\boldsymbol{\Sigma}_{t-1}^{(j)}(\mathbf{H}_t^{(j)})^T + \mathbf{R}_t^{(j)}\right)^{-1} &(4.4)\\ \boldsymbol{\mu}_t^{(j)} &= \boldsymbol{\mu}_t^{(j)} + \mathbf{K}_t^{(j)}\left(z_t^{(j)} - h_t^{(j)}(\mathbf{p}_{n,t}, \boldsymbol{\mu}_t^{(j)})\right) &(4.5)\\ \boldsymbol{\Sigma}_t^{(j)} &= \left(\mathbf{I} - \mathbf{K}_t^{(j)}\mathbf{H}_t^{(j)}\right)\boldsymbol{\Sigma}_{t-1}^{(j)}. &(4.6)\end{aligned}$$

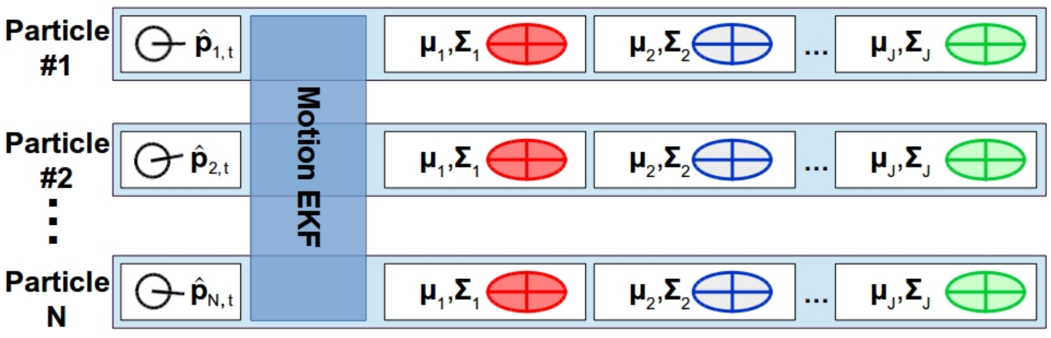


Figure 4.3: Representation of the individual particles in the adapted FastSLAM algorithm.

The question arises, how to initialize $\boldsymbol{\mu}_0^{(j)}$ and $\boldsymbol{\Sigma}_0^{(j)}$. Even after adapting the FastSLAM algorithm to the simulation environment, it cannot deal with the initialization of the features. The probability distribution of the initialization is spherical shaped. For this reason it cannot be well represented by a Gaussian distribution.

4.3 Extension of FastSLAM

The FastSLAM treats the mobile agent localization problem and the feature estimation problems separately. As a consequence, the motion model can be decoupled from the particles. Hence, in our adaption all particles share one EKF for the motion model. This adapted version of the FastSLAM algorithm can be seen in Figure 4.3. The EKF estimates the state vector $\mathbf{x}_t$ from the IMU measurements. Individual particles use the 2D change in position $\Delta\hat{\mathbf{p}}_t$ to update their instances of the agent position. Additionally, a small noise is added to account for the uncertainties of the IMU. The motion model can be written as

$$\hat{\mathbf{p}}_{n,t+1} = \hat{\mathbf{p}}_{n,t} + \Delta\hat{\mathbf{p}}_t + \mathcal{N}(\mathbf{0}, \Sigma_{\text{motion}}), \tag{4.7}$$

with Σ_{motion} depending on the maximum velocity of the mobile agent and the frequency of the IMU.

4.3.1 Extended Kalman filter for the motion

An EKF has to be designed to estimate the position, orientation, velocity, acceleration from the IMU measurements. The state vector $\mathbf{x}_t$ was already mentioned in Section 2.1 but is restated here

$$\mathbf{x}_t = \begin{bmatrix} p_{\text{x},t}, & p_{\text{y},t}, & p_{\text{z}}, & v_{\text{x},t}, & v_{\text{y},t}, & a_{\text{x},t}, & a_{\text{y},t}, & \theta_t \end{bmatrix}^T. \tag{4.8}$$

The equations of the EKF introduced in Section 2.2.2 have to be adapted to the intended use, since the prediction step is linear and the control input is scalar. The adapted formulation is given by

$$\hat{\boldsymbol{\mu}}_t = \mathbf{A}_\mathrm{m}\boldsymbol{\mu}_{t-1} + \mathbf{B}_\mathrm{m}u_t \tag{4.9}$$
$$\hat{\boldsymbol{\Sigma}}_t = \mathbf{A}_\mathrm{m}\boldsymbol{\Sigma}_{t-1}\mathbf{A}_\mathrm{m}^T + \mathbf{Q}_t \tag{4.10}$$
$$\mathbf{K}_t = \hat{\boldsymbol{\Sigma}}_t\mathbf{H}_{\mathrm{m},t}^T(\mathbf{H}_{\mathrm{m},t}\hat{\boldsymbol{\Sigma}}_t\mathbf{H}_{\mathrm{m},t}^T + \mathbf{R}_t)^{-1} \tag{4.11}$$
$$\boldsymbol{\mu}_t = \hat{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_{\mathrm{m},t} - \mathbf{h}_\mathrm{m}(\hat{\boldsymbol{\mu}}_t)) \tag{4.12}$$
$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_{\mathrm{m},t})\hat{\boldsymbol{\Sigma}}_t. \tag{4.13}$$

With the formulation of the motion EKF, the 2D change in position $\Delta\hat{\mathbf{p}}_t$ is calculated with

$$\Delta\hat{\mathbf{p}}_t = \begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\end{bmatrix}\boldsymbol{\mu}_t - \begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\end{bmatrix}\boldsymbol{\mu}_{t-1}. \tag{4.14}$$

The IMU provides measurements on acceleration, angular velocity and the orientation. All measurements directly related to the state vector $\mathbf{x}_t$ are considered in the correction step of the EKF. The rest is treated as control input for the prediction step. Thus, the control input consists of the angular velocity only

$$u_t = \omega_{\mathrm{m},t}. \tag{4.15}$$

The angular velocity for the yaw-rotation can be integrated to get the change in orientation. With this knowledge the control-input matrix $\mathbf{B}_\mathrm{m}$ is defined as

$$\mathbf{B}_\mathrm{m} = \begin{bmatrix}0&0&0&0&0&0&\Delta t\end{bmatrix}^T. \tag{4.16}$$

No information about the motion is known beforehand. Thus, no prior knowledge can be used to improve the motion model. The position, velocity, and acceleration are obtained through the following well-known relationships

$$\mathbf{p}_t = \mathbf{p}_0 + \int_0^t \mathbf{v}_t dt \tag{4.17}$$
$$\mathbf{v}_t = \mathbf{v}_0 + \int_0^t \mathbf{a}_t dt. \tag{4.18}$$

The discretized versions of these relationships result in the system matrix $\mathbf{A}_\mathrm{m}$ of the state transition function

$$\mathbf{A}_\mathrm{m} = \begin{bmatrix}1&0&\Delta t&0&\frac{\Delta t^2}{2}&0&0\\0&1&0&\Delta t&0&\frac{\Delta t^2}{2}&0\\0&0&1&0&\Delta t&0&0\\0&0&0&1&0&\Delta t&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{bmatrix}. \tag{4.19}$$

Due to the reason that a strapdown type IMU is used, the acceleration measurements $\mathbf{a}_{\mathrm{m},t}$ are in the local coordinate system of the IMU. To calculate the local acceleration measurements from the global accelerations in the state vector, the following equation are used for a left hand coordinate system

$$\mathbf{a}_{\mathrm{m},t} = \begin{bmatrix} \cos(\theta_t) & \sin(\theta_t) \\ -\sin(\theta_t) & \cos(\theta_t) \end{bmatrix} \mathbf{a}_t. \tag{4.20}$$

With this equation the non-linear measurement function can be expressed as

$$\mathbf{h}_{\mathrm{m}_1}(\mathbf{x}_t) = \begin{bmatrix} h_1(\mathbf{x}_t) \\ h_2(\mathbf{x}_t) \\ h_3(\mathbf{x}_t) \end{bmatrix} = \begin{bmatrix} \cos(\theta_t)a_{\mathrm{x},t} + \sin(\theta_t)a_{\mathrm{y},t} \\ -\sin(\theta_t)a_{\mathrm{x},t} + \cos(\theta_t)a_{\mathrm{y},t} \\ \theta_t \end{bmatrix}. \tag{4.21}$$

From the measurement function the Jacobian matrix is calculated

$$\mathbf{H}_{\mathrm{m}_1,t} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}_t)}{\partial p_{\mathrm{x},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial p_{\mathrm{y},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial v_{\mathrm{x},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial v_{\mathrm{y},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial a_{\mathrm{x},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial a_{\mathrm{y},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial \theta_t} \\ \frac{\partial h_2(\mathbf{x}_t)}{\partial p_{\mathrm{x},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial p_{\mathrm{y},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial v_{\mathrm{x},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial v_{\mathrm{y},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial a_{\mathrm{x},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial a_{\mathrm{y},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial \theta_t} \\ \frac{\partial h_3(\mathbf{x}_t)}{\partial p_{\mathrm{x},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial p_{\mathrm{y},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial v_{\mathrm{x},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial v_{\mathrm{y},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial a_{\mathrm{x},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial a_{\mathrm{y},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial \theta_t} \end{bmatrix} \tag{4.22}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & \cos(\theta_t) & \sin(\theta_t) & -\sin(\theta_t)a_{\mathrm{x},t} + \cos(\theta_t)a_{\mathrm{y},t} \\ 0 & 0 & 0 & 0 & -\sin(\theta_t) & \cos(\theta_t) & -\cos(\theta_t)a_{\mathrm{x},t} - \sin(\theta_t)a_{\mathrm{y},t} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.23}$$

The designed EKF is relying solely on the input of the IMU. Therefore, the EKF is at the mercy of the error sources of the IMU. Especially, the drifting bias tends to make the position estimate diverge over time. To correct this behavior, the mean position of the resampled particles is used as additional measurement in the correction step. The mean position is given by

$$\hat{\mathbf{p}}_t = \frac{1}{N}\sum_{n=1}^{N} \hat{\mathbf{p}}_{n,t}. \tag{4.24}$$

The new measurement function is given by

$$\mathbf{h}_{\mathrm{m}_2}(\mathbf{x}_t) = \begin{bmatrix} h_1(\mathbf{x}_t) \\ h_2(\mathbf{x}_t) \\ h_3(\mathbf{x}_t) \\ h_4(\mathbf{x}_t) \\ h_5(\mathbf{x}_t) \end{bmatrix} = \begin{bmatrix} \cos(\theta_t)a_{\mathrm{x},t} + \sin(\theta_t)a_{\mathrm{y},t} \\ -\sin(\theta_t)a_{\mathrm{x},t} + \cos(\theta_t)a_{\mathrm{y},t} \\ \theta_t \\ p_{\mathrm{x},t} \\ p_{\mathrm{y},t} \end{bmatrix}. \tag{4.25}$$

The corresponding Jacobian matrix is calculated in the same manner as before

$$\mathbf{H}_{\mathrm{m}_2,t} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}_t)}{\partial p_{\mathrm{x},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial p_{\mathrm{y},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial v_{\mathrm{x},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial v_{\mathrm{y},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial a_{\mathrm{x},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial a_{\mathrm{y},t}} & \frac{\partial h_1(\mathbf{x}_t)}{\partial \theta_t} \\ \frac{\partial h_2(\mathbf{x}_t)}{\partial p_{\mathrm{x},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial p_{\mathrm{y},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial v_{\mathrm{x},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial v_{\mathrm{y},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial a_{\mathrm{x},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial a_{\mathrm{y},t}} & \frac{\partial h_2(\mathbf{x}_t)}{\partial \theta_t} \\ \frac{\partial h_3(\mathbf{x}_t)}{\partial p_{\mathrm{x},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial p_{\mathrm{y},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial v_{\mathrm{x},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial v_{\mathrm{y},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial a_{\mathrm{x},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial a_{\mathrm{y},t}} & \frac{\partial h_3(\mathbf{x}_t)}{\partial \theta_t} \\ \frac{\partial h_4(\mathbf{x}_t)}{\partial p_{\mathrm{x},t}} & \frac{\partial h_4(\mathbf{x}_t)}{\partial p_{\mathrm{y},t}} & \frac{\partial h_4(\mathbf{x}_t)}{\partial v_{\mathrm{x},t}} & \frac{\partial h_4(\mathbf{x}_t)}{\partial v_{\mathrm{y},t}} & \frac{\partial h_4(\mathbf{x}_t)}{\partial a_{\mathrm{x},t}} & \frac{\partial h_4(\mathbf{x}_t)}{\partial a_{\mathrm{y},t}} & \frac{\partial h_4(\mathbf{x}_t)}{\partial \theta_t} \\ \frac{\partial h_5(\mathbf{x}_t)}{\partial p_{\mathrm{x},t}} & \frac{\partial h_5(\mathbf{x}_t)}{\partial p_{\mathrm{y},t}} & \frac{\partial h_5(\mathbf{x}_t)}{\partial v_{\mathrm{x},t}} & \frac{\partial h_5(\mathbf{x}_t)}{\partial v_{\mathrm{y},t}} & \frac{\partial h_5(\mathbf{x}_t)}{\partial a_{\mathrm{x},t}} & \frac{\partial h_5(\mathbf{x}_t)}{\partial a_{\mathrm{y},t}} & \frac{\partial h_5(\mathbf{x}_t)}{\partial \theta_t} \end{bmatrix} \tag{4.26}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & \cos(\theta_t) & \sin(\theta_t) & -\sin(\theta_t)a_{\mathrm{x},t} + \cos(\theta_t)a_{\mathrm{y},t} \\ 0 & 0 & 0 & 0 & -\sin(\theta_t) & \cos(\theta_t) & -\cos(\theta_t)a_{\mathrm{x},t} - \sin(\theta_t)a_{\mathrm{y},t} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.27}$$

The problem that the IMU and the features have different frequencies was already encountered in Section 3.3. Hence, the measurement function and the Jacobian matrix change depending if feature measurements are available. They are stated as

$$\mathbf{z}_{\mathrm{m},t} = \begin{cases} \begin{bmatrix} \mathbf{a}_{\mathrm{m},t}, & \theta_{\mathrm{m},t}, & \hat{\mathbf{p}}_t \end{bmatrix}^T, & \text{if } \bmod\left(t, \frac{1}{f_{\text{features}}}\right) = 0 \\ \begin{bmatrix} \mathbf{a}_{\mathrm{m},t}, & \theta_{\mathrm{m},t} \end{bmatrix}^T, & \text{otherwise} \end{cases} \tag{4.28}$$

$$\mathbf{h}_{\mathrm{m}}(\mathbf{x}_t) = \begin{cases} \mathbf{h}_{\mathrm{m}_2}(\mathbf{x}_t), & \text{if } \bmod\left(t, \frac{1}{f_{\text{features}}}\right) = 0 \\ \mathbf{h}_{\mathrm{m}_1}(\mathbf{x}_t), & \text{otherwise} \end{cases} \tag{4.29}$$

$$\mathbf{H}_{\mathrm{m},t} = \begin{cases} \mathbf{H}_{\mathrm{m}_2,t}, & \text{if } \bmod\left(t, \frac{1}{f_{\text{features}}}\right) = 0 \\ \mathbf{H}_{\mathrm{m}_1,t}, & \text{otherwise} \end{cases}. \tag{4.30}$$

4.3.2 Overcoming the initialization problem

The basic idea is to overcome the initialization problem arising from the RO measurements with an approach that can represent a spherical distribution. Thus, a PF is used to represent the features. The new representation can be seen in Figure 4.4. The feature are represented as a PF. The PF for the j-th feature position at time instance t is denoted by

$$\mathcal{F}_{n,t}^{(j)} = \{\hat{\mathbf{f}}_{n,1,t}^{(j)}, \hat{\mathbf{f}}_{n,2,t}^{(j)}, ..., \hat{\mathbf{f}}_{n,K,t}^{(j)}\}, \tag{4.31}$$

where n denotes the index of the agent particle. The state PF $\mathcal{P}_t$ at time instance t can then be denoted by

$$\mathcal{F}_{n,t} = \{\mathcal{F}_{n,t}^{(1)}, \mathcal{F}_{n,t}^{(2)}, ..., \mathcal{F}_{n,t}^{(J)}\} \tag{4.32}$$

$$\mathcal{P}_t = \{\{\hat{\mathbf{p}}_{1,t}, \mathcal{F}_{1,t}\}, \{\hat{\mathbf{p}}_{2,t}, \mathcal{F}_{2,t}\}, ..., \{\hat{\mathbf{p}}_{N,t}, \mathcal{F}_{N,t}\}\}, \tag{4.33}$$

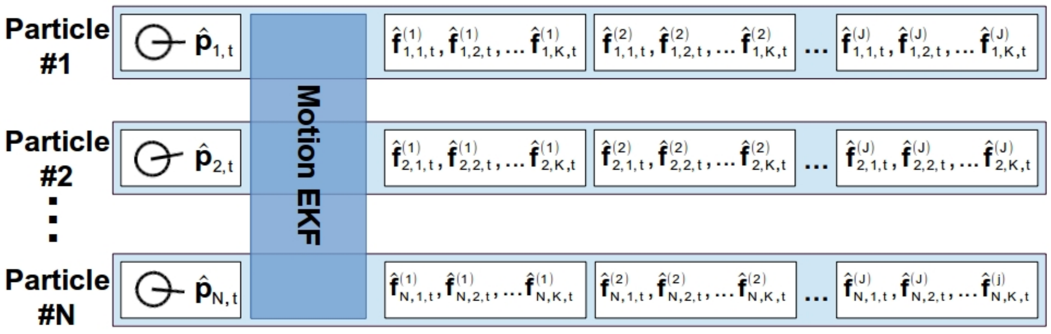

Particle #1
Particle #2
Particle #N
Motion EKF
$\hat{\mathbf{p}}_{1,t}$
$\hat{\mathbf{p}}_{2,t}$
$\hat{\mathbf{p}}_{N,t}$
$\hat{\mathbf{f}}^{(1)}_{1,1,t}, \hat{\mathbf{f}}^{(1)}_{1,2,t}, \dots \hat{\mathbf{f}}^{(1)}_{1,K,t}$
$\hat{\mathbf{f}}^{(2)}_{1,1,t}, \hat{\mathbf{f}}^{(2)}_{1,2,t}, \dots \hat{\mathbf{f}}^{(2)}_{1,K,t}$
$\hat{\mathbf{f}}^{(J)}_{1,1,t}, \hat{\mathbf{f}}^{(J)}_{1,2,t}, \dots \hat{\mathbf{f}}^{(J)}_{1,K,t}$
$\hat{\mathbf{f}}^{(1)}_{2,1,t}, \hat{\mathbf{f}}^{(1)}_{2,2,t}, \dots \hat{\mathbf{f}}^{(1)}_{2,K,t}$
$\hat{\mathbf{f}}^{(2)}_{2,1,t}, \hat{\mathbf{f}}^{(2)}_{2,2,t}, \dots \hat{\mathbf{f}}^{(2)}_{2,K,t}$
$\hat{\mathbf{f}}^{(J)}_{2,1,t}, \hat{\mathbf{f}}^{(J)}_{2,2,t}, \dots \hat{\mathbf{f}}^{(J)}_{2,K,t}$
$\hat{\mathbf{f}}^{(1)}_{N,1,t}, \hat{\mathbf{f}}^{(1)}_{N,2,t}, \dots \hat{\mathbf{f}}^{(1)}_{N,K,t}$
$\hat{\mathbf{f}}^{(2)}_{N,1,t}, \hat{\mathbf{f}}^{(2)}_{N,2,t}, \dots \hat{\mathbf{f}}^{(2)}_{N,K,t}$
$\hat{\mathbf{f}}^{(J)}_{N,1,t}, \hat{\mathbf{f}}^{(J)}_{N,2,t}, \dots \hat{\mathbf{f}}^{(j)}_{N,K,t}$

Figure 4.4: Representation of the individual particles in the extended FastSLAM algorithm.

Initialization of the feature particles

A PF can represent every arbitrary distribution if enough particles are used. The RO measurements give only information on the distance between the current position $\mathbf{p}_t$ of the mobile agent and the position of the feature $\mathbf{f}^{(j)}$. This means that the feature can be located anywhere on the surface of a sphere with the radius of the given distance measurement. Furthermore, with larger distance measurements the space that has to be covered with the particles increases quadratical due to the 3D nature of the problem. Therefore, a high number of particles is needed to cover the 3D space. A parameter K describing the number of particles for the features has to be introduced. This number has a big impact on the computing effort of the algorithm.

To initialize the particles, the spherical coordinate system is used. With this coordinate system any point on the sphere can be represented with the radius, the azimuthal angle θ, and the elevation angle φ. Figure 4.5 visualizes the spherical coordinate system for an arbitrary particle. The particles should be equally distributed over the surface area of the sphere. The naive approach, sampling the angles from a uniform distribution, will not work because the surface area decreases towards the poles.

To get equally distributed points over the surface area of the sphere, a probability density function (PDF) $p(\theta, \varphi)$ for the two angles has to be found. Additionally, the PDF $p(\theta, \varphi)$ has to ensure that both angles are equally likely over the whole surface area of the sphere. This PDF can be derived from the differential surface area element parameterized on the spherical coordinate system. A differential surface area element can be seen as the multiplication of two infinitesimal circle arcs. So, the differential surface area can be formulated as

$$d\mathrm{A} = r d\varphi \cdot L d\theta = r d\varphi \cdot r \sin(\varphi) d\theta = r^2 \sin(\varphi) d\varphi d\theta. \tag{4.34}$$

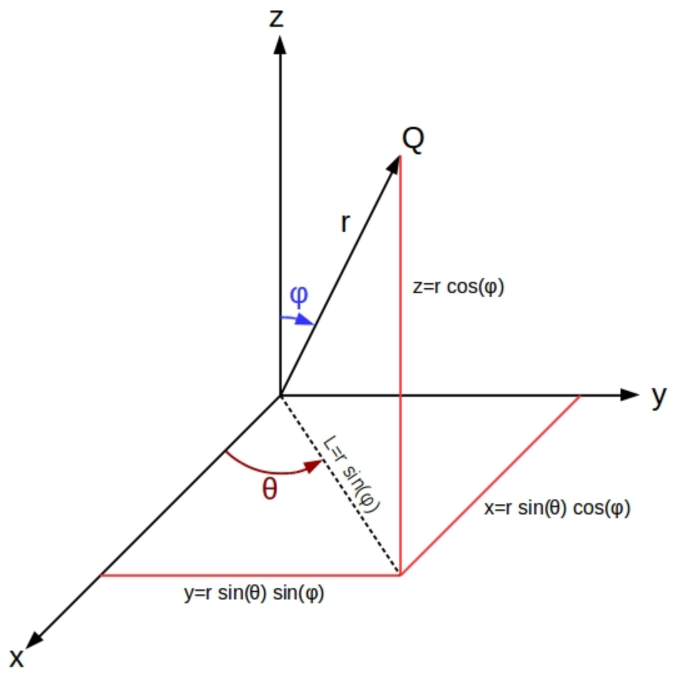


Figure 4.5: Relationship between the spherical polar coordinates and Cartesian coordinates.

With this equation, the probability that an arbitrary point Q lies in this infinitesimal surface area is given by

$$p(Q)d\mathrm{A}. \tag{4.35}$$

Now the PDF $p(Q)$ can be calculated as

$$\int_0^{\pi}\int_0^{2\pi} p(Q)dA \overset{!}{=} 1 \tag{4.36}$$

$$A = \int_0^{\pi}\int_0^{2\pi} dA = \int_0^{\pi}\int_0^{2\pi} r^2 sin(\varphi)d\theta d\varphi = r^2 4\pi \tag{4.37}$$

$$p(Q) = \frac{1}{r^2 4\pi}. \tag{4.38}$$

The point Q can be represented differently using the parametrization of the angles θ, φ with their corresponding PDF $p(\theta, \varphi)$. Therefore, the following equation must be applied

$$p(Q)d\mathrm{A} = p(\theta, \varphi)d\theta d\varphi. \tag{4.39}$$

From this equation the PDF $p(\theta, \varphi)$ can be calculated

$$p(\theta, \varphi) = \frac{1}{4\pi}\sin(\varphi). \tag{4.40}$$

The marginal PDFs are defined as

$$p(\theta) = \int_0^{\pi} p(\theta, \varphi) d\varphi = \frac{1}{2\pi} \tag{4.41}$$

$$p(\varphi) = \int_0^{2\pi} p(\theta, \varphi) d\theta = \frac{\sin(\varphi)}{2}. \tag{4.42}$$

The last step is to generate angles which are distributed like their corresponding PDFs. This can be achieved with the inversion method [48]. The main idea is to use samples from a uniform distribution $u_1, u_2 \sim \mathcal{U}(0, 1)$ to generate other probability distributions. This can be done using the following equations

$$P(\theta) = u_1 \tag{4.43}$$

$$P^{-1}(u) = \theta \tag{4.44}$$

$$P(\varphi) = u_2 \tag{4.45}$$

$$P^{-1}(u) = \varphi, \tag{4.46}$$

with $P(\theta)$ and $P(\varphi)$ being the cumulative distribution functions of the angles. The cumulative distribution functions of the angles are given by

$$P(\theta) = \int_0^{\theta} p(\hat{\theta}) d\hat{\theta} = \frac{\theta}{2\pi}, \ 0 \leq \theta < 2\pi \tag{4.47}$$

$$P(\varphi) = \int_0^{\varphi} p(\hat{\varphi}) d\hat{\varphi} = \frac{1 - \cos(\varphi)}{2}, \ 0 \leq \varphi \leq \pi. \tag{4.48}$$

$$\tag{4.49}$$

With this, the inversion method can be calculated as

$$\theta = 2\pi u_1 \tag{4.50}$$

$$\varphi = \arccos(1 - 2u_2). \tag{4.51}$$

In the next step, K particles are generated. The received distance measurement corresponds to the radius r of the sphere. This distance measurement contains noise. As a consequence, the initialization procedure has to reflect the measurement model. To meet this requirement the radius r is sampled from a normal distribution with the variance of the estimator. The equations used to generate the particles are written as

$$r = \mathcal{N}(z_t^{(j)}, \mathrm{var(d)}) \tag{4.52}$$

$$\theta = 2\pi u_1 \tag{4.53}$$

$$\varphi = \arccos(1 - 2u_2) \tag{4.54}$$

After calculating the spherical coordinates, they are converted back to the Cartesian coordinate system and the position of the mobile agent is added. The initialization for the

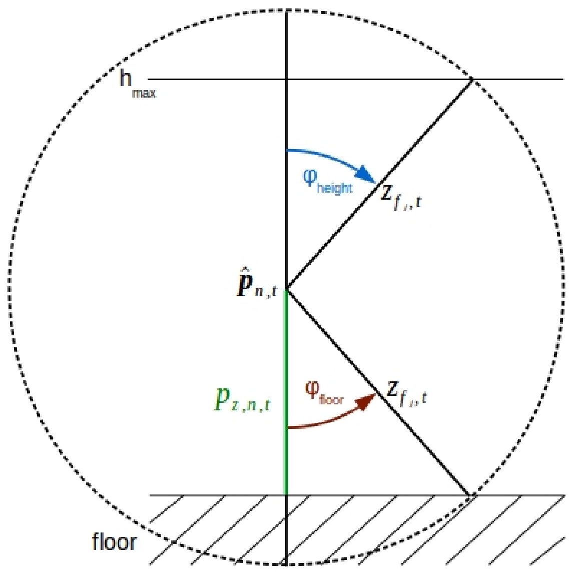


Figure 4.6: Visualization of the restriction on the elevation angle.

k-th feature particle in the n-th agent particle can be formulated as

$$\hat{\mathbf{f}}^{(j)}_{n,k,t} = \begin{bmatrix} r\sin(\theta)\cos(\varphi) \\ r\sin(\theta)\sin(\varphi) \\ r\sin(\varphi) \end{bmatrix} + \hat{\mathbf{p}}_{n,t}. \tag{4.55}$$

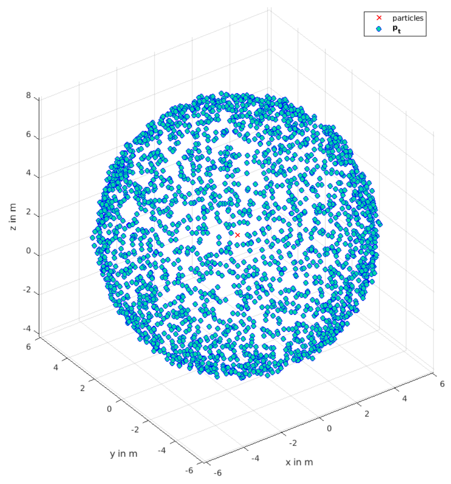
In real-world scenarios, the features are unlikely to be positioned above a certain height. Furthermore, the features cannot be positioned below the floor. Thus, the particles could be used more efficiently. To restrict the initialization to a reasonable area, a parameter for the maximum height $h_{\max}$ is introduced. All the simulations conducted in Chapter 5 use a maximum height of 3 m. A visualization of the resulting range of elevation angles can be seen in Figure 4.6.
The limiting angles are calculated as

$$\varphi_{\text{floor}} = \begin{cases} \arccos\left(\frac{p_{\text{z},n,t}}{z_t^{(j)}}\right) & \text{if } p_{\text{z},n,t} < z_t^{(j)} \\ 0 & \text{otherwise} \end{cases} \tag{4.56}$$

$$\varphi_{\text{height}} = \begin{cases} \arccos\left(\frac{h_{\max} - p_{\text{z},n,t}}{z_t^{(j)}}\right) & \text{if } h_{\max} - p_{\text{z},n,t} < z_t^{(j)} \\ 0 & \text{otherwise} \end{cases}. \tag{4.57}$$

The standard elevation angle is an element of the interval $[0, \pi]$. The updated elevation angle has a smaller interval of $[0 + \varphi_{\text{height}}, \pi - \varphi_{\text{floor}}]$. Thus, the limiting angles have to be

(a) (b)



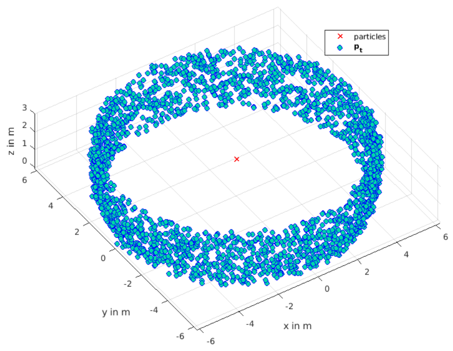


Figure 4.7: Comparison of the (a) spherical initialization and (b) annular initialization.

updated over the cumulative distribution function considering the new interval

$$\hat{\varphi}_{\text{floor}} = \frac{1-\cos(\pi - \varphi_{\text{floor}})}{2} \tag{4.58}$$

$$\hat{\varphi}_{\text{height}} = \frac{1-\cos(\varphi_{\text{height}})}{2}. \tag{4.59}$$

With the updated angles, the final elevation angle can be formulated as

$$u_{\text{new}} \sim \mathcal{U}(\hat{\varphi}_{\text{height}}, \hat{\varphi}_{\text{floor}}) \tag{4.60}$$

$$\varphi = \arccos(1 - 2u_{\text{new}}). \tag{4.61}$$

The restrictions on the elevation angle φ change the shape of the initialization from spheric to angular. An instance of the two different initializations can be seen in Figure 4.7. The initializations were generated with $z_{f_j,t} = 6\,\text{m}$, $\hat{\mathbf{p}}_{n,t} = \begin{bmatrix}0\,\text{m}, & 0\,\text{m}, & 2\,\text{m}\end{bmatrix}^T$, and $\text{var}(d) = 0.01\,\text{m}$.

4.3.3 Resampling

In our approach a PF within a PF is applied. Hence, two PF depending on each other have to be resampled. For the resampling algorithm all the particles have to be rated with a certain weight that corresponds to the measurement model. The weight of a feature particle can be calculated as

$$\hat{w}^{(j)}_{n,k,t} = \frac{1}{\sqrt{2\pi \mathrm{var(d)}}} \exp\left(-\frac{(||\hat{\mathbf{p}}_{n,t} - \hat{\mathbf{f}}^{(j)}_{n,k}||_2 - z^{(j)}_t)^2}{2\mathrm{var(d)}} \right) \cdot \hat{w}^{(j)}_{n,k,t-1}. \tag{4.62}$$

After the calculation of all the feature weights, they have to be normalized

$$w^{(j)}_{n,k,t} = \frac{\hat{w}^{(j)}_{n,k,t}}{\sum_{k=1}^{K} w^{(j)}_{n,k,t}}. \tag{4.63}$$

The weight for the agent particle can be calculated as

$$\hat{w}_{n,t} = \prod_{j=1}^{J} \left(\frac{1}{K} \sum_{k=1}^{K} \hat{w}^{(j)}_{n,k,t} \right) \cdot w_{n,t-1}. \tag{4.64}$$

Also the agent particle weight have to be normalized

$$w_{n,t} = \frac{\hat{w}_{n,t}}{\sum_{n=1}^{N} \hat{w}_{n,t}}. \tag{4.65}$$

From the different algorithms to resample a PF, systematic resampling was chosen because it has a linear computational complexity [49] and only one random number is drawn per resampling step. Furthermore, the systematic resampling algorithm achieves comparable results compared to other resampling algorithms [50] but is the simplest method to implement. Two slightly modified versions of the basic systematic resampling algorithm are used to resample the two PFs.

Algorithm 2 Systematic state resampling

```
1: procedure RESAMPLE_STATE
2:     $\mathcal{P}_t = \emptyset$
3:     calculate $w_{1,t-1}$
4:     $c_1 = w_{1,t-1}$
5:     for n = 2 to N do
6:         calculate $w_{n,t-1}$
7:         $c_n = c_{n-1} + w_{n,t-1}$
8:     $u_1 \sim \mathcal{U}(0, \frac{1}{N})$
9:     $i = 1$
10:    for l = 1 to N do
11:        while $u_l > c_i$ do
12:            $i = i + 1$
13:        $u_{l+1} = u_l + \frac{1}{N}$
14:        if $i$ changed OR $i$ is 1 then
15:            get $\hat{\mathbf{p}}_{i,t-1}$ from $\mathcal{P}_{t-1}$
16:            $\mathcal{F}_{i,t}$ = resample_features
17:        insert $\{\hat{\mathbf{p}}_{i,t-1}, \mathcal{F}_{i,t}\}$ into $\mathcal{P}_t$
18:    return $\mathcal{P}_t$
```

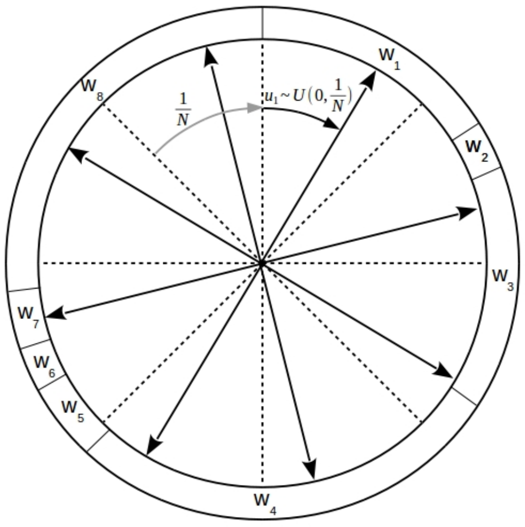


Figure 4.8: Visualization of the systematic resampling algorithm with a particle number $N = 8$.

The pseudo code of the modified systematic resampling algorithm for the agent particles can be seen in Algorithm 2. A good way to visualize the working principle is a wheel where the outer ring correspond to the cumulative distribution function of the weights. An example for such a wheel is given in Figure 4.8. A number of segments equivalent to the number of resampled particles are equally spaced across the wheel (see dashed line in Figure 4.8). An offset smaller than the angle between two segments is randomly drawn. All the segment boundaries are rotated according to this offset (arrows in Figure 4.8). All the particles where an arrow points on their weight in the outer ring are resampled.
The pseudo code mimics this behavior. First of all, the cumulative distribution function (outer ring of the wheel) is generated (line 3-7). The offset (first arrow) is drawn from a uniform distribution (line 8). The index i is used to determine which agent particles in $\mathcal{P}_{t-1}$ get resampled. The first particle has the index $i = 1$ (line 11). The outer for-loop ensures that exactly N agent particles are resampled (line 10). The inner while-loop is used to skip all the agent particles until the next offset u_j is reached (line 11-12). This corresponds to the wheel where all the agent particles are skipped if no arrow is pointing onto them. The offset is increased by $\frac{1}{N}$ (line 13). Afterwards a particle from the old particle set is added to the new particle set (line 11). If the index i has changed, the features of the particle have to be resampled (line 14-16).

Algorithm 3 Systematic feature resampling

```
1: procedure RESAMPLE_FEATURES
2:     $\mathcal{F}_{n,t} = \emptyset$
3:     for j = 1 to J do
4:         if #(new measurements) < $M$ then
5:             continue
6:         $\mathcal{F}_{n,t}^{(j)} = \emptyset$
7:         calculate $w_{n,1,t-1}^{(j)}$
8:         $c_1 = w_{n,1,t-1}^{(j)}$
9:         for k = 2 to K do
10:            calculate $w_{n,k,t-1}^{(j)}$
11:            $c_k = c_{k-1} + w_{n,k,t-1}^{(j)}$
12:        $i = 1$
13:        $u_1 \sim \mathcal{U}(0, \frac{1}{K})$
14:        for l = 1 to K do
15:            while $u_l > c_i$ do
16:                $i = i + 1$
17:            $u_{l+1} = u_l + \frac{1}{K}$
18:            get $\hat{\mathbf{f}}_{n,i,t-1}^{(j)}$ from $\mathcal{F}_{n,t-1}^{(j)}$
19:            $\hat{\mathbf{f}}_{n,i,t}^{(j)} = \hat{\mathbf{f}}_{n,i,t-1}^{(j)}$ + noise
20:            insert $\hat{\mathbf{f}}_{n,i,t}^{(j)}$ into $\mathcal{F}_{n,t}^{(j)}$
21:        insert $\mathcal{F}_{n,t}^{(j)}$ into $\mathcal{F}_{n,t}$
22:    return $\mathcal{F}_{n,t}$
```

The pseudo code of the modified systematic resampling algorithm for the feature particles can be seen in Algorithm 3. This version of the systematic resampling algorithm has two major differences compared to the one used for the agent particles. There is one more for-loop (line 3). This for-loop iterates over all the features where measurements were already provided. In addition, a resampling step for the feature particles is not necessary after each measurement. Therefore, the jth feature particle is resampled after M new measurements (line 4-5). This number was empirically determined to be $M = 3$.
The second change is the little noise added to each resampled feature position $\hat{\mathbf{f}}_{n,i,t-1}^{(j)}$. If no noise was added, the feature PF would be stuck with the particles from the initialization.

4.3.4 Numerical stability of agent particle weight

The smallest positive normalized floating-point number in IEEE double precision for MATLAB is $2.225073858507201 \cdot 10^{-308}$. With a high number of features J, the state particle weight $\hat{w}_{n,t}$ can get smaller than the smallest floating point number. Single state particle weights falling below the smallest floating point number would not be a problem but if all the state particle weights get 0, a normalization is not possible anymore. To circumvent this issue, the log-PF proposed in [51] is used. This version transfers the weight to the logarithm domain with

$$\hat{w}_{n,t} = \sum_{j=1}^{J} \left(\log \left(w_{n,t-1} + \frac{1}{K} \sum_{k=1}^{K} \hat{w}_{n,k,t}^{(j)} \right) \right). \tag{4.66}$$

Before the resampling step the weight is transfered back with

$$w_{n,t}^{+} = \exp(\hat{w}_{n,t} - \max_{n}(\hat{w}_{n,t})) \tag{4.67}$$

and is normalized with

$$w_{n,t} = \frac{w_{n,t}^{+}}{\sum_{n=1}^{N} w_{n,t}^{+}}. \tag{4.68}$$

4.3.5 Reduction of feature particles

The number of particles can be calculated with the multiplication of

$$N_{\text{particles}} = N \cdot K \cdot J, \tag{4.69}$$

where N denotes the number of agent particles, K is the number of feature particles, and J is the number of features in one scenario. This is also the number of particles that have to be resampled. Most time of the algorithm is spent on resampling, due to this high number of particles. Therefore, a reduction of the particles would mean a big decrease in computing effort. The number of features J is set by the environment. So, not much can be done about that. A reduction of the N agent particles would have the biggest decrease in computing effort. Unfortunately, a certain number of particles is needed to correct the drifting behavior of the IMU. Hence, the reduction of the number of feature particles K would make sense since the feature position stays constant over time. Two different approaches are investigated for the reduction. The approaches are applied before of the resampling procedure to calculate a new number of particles for each feature PF.

The first reduction method is based on the likelihood of the particles. For that an estimate of the effective sample size $K_{\text{eff},n,t}^{(j)}$ introduced in [52] is used. The effective sample size $K_{\text{eff},n,t}^{(j)}$ for the jth feature PF is given as

$$K_{\text{eff},n,t}^{(j)} = \frac{K}{1 + \text{var}(w_{n,k,t}^{(j)*})}, \tag{4.70}$$

where $w_{n,k,t}^{(j)*}$ is referred to as the "true weight". The true weight cannot be determined, but an estimate $\hat{K}_{\text{eff},n,t}^{(j)}$ [34] can be obtained by

$$\hat{K}_{\text{eff},n,t}^{(j)} = \frac{1}{\sum_{k=1}^{K} (w_{n,k,t}^{(j)})^2}. \tag{4.71}$$

Actually those two equations are a measure for the degeneracy of the PF. In [34] this measure is applied to detect when the PF needs to be resampled. The variance of the weights can only increase within one time step. Therefore, the estimate $\hat{K}_{\text{eff},n,t}^{(j)}$ can only get smaller in each time step. The method would reduce the particle number to one over time. This would ruin the PF representation. Therefore, it must be guaranteed that the number does not fall below a certain minimal particle number K_{min}. The feature particle number $K_{n,t}^{(j)}$ is calculated as

$$K_{n,t}^{(j)} = \max(K_{\text{min}}, \hat{K}_{\text{eff},n,t}^{(j)}). \tag{4.72}$$

This ensures that the PF has to maintain at least K_{min} feature particles at all times.

The second reduction method is based on the Kullback–Leibler divergence. The key idea of the KLD-sampling method is to bound the approximation error introduced by the sample-based representation of the PF. The approximation error is measured by the Kullback-Leibler divergence from the maximum likelihood estimate to the true distribution modeled as a multinomial distribution. A small number of feature particles is chosen if the density is focused on small parts of the state space. A large number of particle is chosen if the state uncertainty is high. A derivation of the approach can be found in [53]. In contrast to the first method, the KLD-sampling method can also increase the number of feature particles. The problem with this method is that four design parameters have to be set. The four parameter are the quantile $1 - \delta$, the error bound ϵ, the n-dimensional bucket size of the multinomial distribution b_{size}, and the minimal number of particles K_{min}. Moreover, these parameters are changing depending on the application of the scenario. That is why these design parameters have been empirically determined. For integration of this reduction method the implementation in [54] was used.

4.3.6 Switching to an EKF representation

For an even greater decrease in the computing effort, the representation of the feature could be switched back to the EKF representation if the feature particles have converged. The main idea, for switching back, is that the PF representation of the features is not needed anymore since the initialization problem is overcome. The main issue is to determine when the feature particles have converged. This task is rather straightforward with the KLD-sampling method. If the minimal number of feature particles is reached, the PF has converged. With the likelihood approach this is not sufficient, since the method gives information rather on the degeneracy than on the conversion of the PF. Therefore, a second condition needs to be met. This condition is the variance of a feature PF. This variance can be calculated with the covariance matrix of the feature PF. The covariance matrix is defined as

$$\mathbf{\Sigma}^{(j)}_{\mathrm{cov},n,t} = \frac{1}{K}\sum_{k=1}^{K}(\hat{\mathbf{f}}^{(j)}_{n,t} - \hat{\mathbf{f}}^{(j)}_{n,k,t})(\hat{\mathbf{f}}^{(j)}_{n,t} - \hat{\mathbf{f}}^{(j)}_{n,k,t})^T, \tag{4.73}$$

where $\hat{\mathbf{f}}^{(j)}_{n,t}$ is the mean of the feature PF given by

$$\hat{\mathbf{f}}^{(j)}_{n,t} = \frac{1}{K}\sum_{k=1}^{K}\hat{\mathbf{f}}^{(j)}_{n,k,t}. \tag{4.74}$$

The variance of a feature PF is defined as the trace of the covariance matrix given by

$$\mathrm{var}(\hat{\mathbf{f}}^{(j)}_{n,t}) = \mathrm{Tr}(\mathbf{\Sigma}^{(j)}_{\mathrm{cov},n,t}). \tag{4.75}$$

After defining the convergence of the particle filter there is still the issue of the initialization of the EKF. Unlike in Section 4.2.1, the converged PFs can be used to initialize the EKF with

$$\boldsymbol{\mu}^{(j)}_{n,t} = \hat{\mathbf{f}}^{(j)}_{n,t} \tag{4.76}$$

$$\mathbf{\Sigma}^{(j)}_{n,t} = \begin{bmatrix} [\mathbf{\Sigma}^{(j)}_{\mathrm{cov},n,t}]_{1,1} & 0 & 0 \\ 0 & [\mathbf{\Sigma}^{(j)}_{\mathrm{cov},n,t}]_{2,2} & 0 \\ 0 & 0 & [\mathbf{\Sigma}^{(j)}_{\mathrm{cov},n,t}]_{3,3} \end{bmatrix}. \tag{4.77}$$

After the initialization the EKF is equivalent to the one derived in Section 4.2.1.

4.3.7 Outlier detection

Bad measurements can lead to the circumstance that a feature particle is initialized at the wrong position. In addition, it may occur that a feature changes its position. In the grocery store, this can happen if a customer or an employee accidentally throws down a label and puts it back at the wrong position. A similar thing can happen with workers or machines in the warehouse scenario. Thus, a mechanism has to be implemented that can recognize this behavior.
The approach has to detect two things: the PF for the feature has converged and the PF is in the wrong position. Therefore, two criteria are needed. A definition for the convergence of a particle was already encountered in the last section. The wrong position can be detected with the likelihood of the particles. The threshold for such a likelihood depends on the measurement model and the definition of the outlier in the application. In the case of a

Gaussian distribution, we define the feature particle as an outlier if the measurement is not within three standard deviations. This interval includes 99.7 % of all measurements. The likelihood threshold w_{outlier} can then be calculated as

$$w_{\text{outlier}} = \frac{1}{2\pi \text{var}(\text{d})} \exp\left(-\frac{(3\sigma)^2}{2\sigma^2}\right) = \frac{1}{2\pi \text{var}(\text{d})} \exp\left(-\frac{9}{2}\right). \tag{4.78}$$

If only the weight of one time step is used to detect the wrong position, one bad measurement can trigger an outlier. Thus, the weights of the r last measurements for a feature PF are used. Each feature PF stores the weights of the last r measurements. Thus, the likelihood can be calculated as

$$\hat{w}_{n,t}^{(j)} = \frac{1}{\sqrt{2\pi \text{var}(\text{d})}} \exp\left(-\frac{(||\hat{\mathbf{p}}_{n,t} - \hat{\mathbf{f}}_{n,t}^{(j)}||_2 - z_t^{(j)})^2}{2\text{var}(\text{d})}\right) \tag{4.79}$$

$$\mathcal{W} = \{\text{weights } \hat{w}_{n,t}^{(j)} \text{ of last } r \text{ measurements}\} \tag{4.80}$$

$$\overline{w}_{n,t}^{(j)} = \prod_{w \in \mathcal{W}} w. \tag{4.81}$$

Using more than one measurements also changes the calculated likelihood threshold w_{outlier}. The likelihood threshold has to ensure that each weight $\hat{w}_{n,t}^{(j)}$ from the last r measurements is classified as an outlier which leads to a r times multiplication of the original likelihood threshold.

Algorithm 4 Outlier detection method

```
1: procedure OUTLIER_DETECTION
2:     calculate $\text{var}(\hat{\mathbf{f}}_{n,t}^{(j)})$
3:     calculate $\overline{w}_{n,t}^{(j)}$
4:     if $\text{var}(\hat{\mathbf{f}}_{n,t}^{(j)}) < \sigma_{\text{f}}^2$ AND $\overline{w}_{n,t}^{(j)} < (w_{\text{outlier}})^r$ then
5:         set $\mathcal{F}_{n,t}^{(j)} = \emptyset$ in $\mathcal{P}_t$
6:         reinitialize $\mathcal{F}_{n,t}^{(j)}$ and insert into $\mathcal{P}_t$
```

The pseudo code for the outlier detection mechanism can be seen in Algorithm 4. For the EKF representation the feature particles have already converged. Hence, only the likelihood condition needs to be checked.

The problem of this approach is that is does not distinguish between a bad localization or the wrong positioning of a feature. Therefore, a decent localization is needed for this outlier detection to work. The calculated likelihood threshold should be chosen with this in mind. It is better to choose the threshold bigger than too small.

5 Results

From the previous chapters it can be seen that there are various parameters which influence the results of the SLAM algorithm. They are related to the assumed mobile agent, the environment and the algorithm itself.
The ideal evaluation would be the simulation of all combinations of all parameters with each other. Such an approach would lead to a multi dimensional grid search which would not be computable in a reasonable time. Consequently, the number of parameters for evaluation have to be reduced, not every parameter can be evaluated. Moreover, it is not feasible to analyze every parameter combination even with a reduced parameter set. Based on these restrictions only the most important parameters are analyzed one at a time, the rest of parameter set is fixed. Due to the high number of parameters, the focus is on the parameters that have the biggest influence on the performance of the algorithm. For the mobile agent this will be the trajectories and the IMUs. For the environment, the different scenarios, number of anchors, and the measurement precision are evaluated. Furthermore, the number of agent particles, the different reduction methods for the feature particles and the outlier detection will be covered.
The first section introduces the RMSEs used to evaluate the different influences on the SLAM algorithm. The remaining sections in this chapter deal with the analyses of the mentioned parameters.

5.1 Evaluating the algorithm

The SLAM algorithm has to be tested with different realizations of the measurements. This ensures that the SLAM algorithm not only works for one specific instance of measurements. The realization is conducted with the control of the random number generator. The random number generator has the option of an initialization with a natural number denoted as the seed. The seed determines the sequence of the drawn random numbers. All the RO measurements are generated directly after the initialization of the random number generator. So, simulations with the same seed s get the same measurements.
To evaluate the SLAM algorithm, a suitable measure for the performance has to be found. The obvious choice is to analyze and quantify the two parts of the algorithm: the localization and the mapping. The RMSEs is used to evaluate the two parts. The RMSE is a suitable measure because it accounts for the bias and the variance of the particles. The RMSE for

an estimator $\hat{\zeta}$ of a true parameter ζ can be written as

$$\mathrm{RMSE}(\hat{\zeta}) = \sqrt{E((\hat{\zeta} - \zeta)^2)} = \sqrt{\mathrm{var}(\hat{\zeta}) + \mathrm{bias}(\hat{\zeta})^2}. \tag{5.1}$$

Furthermore, the RMSE has an intuitive interpretation since the RMSE of vector to another vector in space is the Euclidean distance between them.
The RMSE for the localization is calculated over the whole time of the simulation. Thus, it is the RMSE of the Euclidean distances from the average agent position $\hat{\mathbf{p}}_t$ to the real agent position $\mathbf{p}_t$ at every time instant t. The localization RMSE for one simulation with seed s is defined as

$$e_{\mathrm{loc},s} = \sqrt{\frac{1}{T \cdot \mathrm{f}_{\mathrm{IMU}}} \sum_{t=\Delta t}^{T} ||\hat{\mathbf{p}}_t - \mathbf{p}_t||_2^2}, \tag{5.2}$$

where $\hat{\mathbf{p}}_t$ is the mean of the agent particles given by

$$\hat{\mathbf{p}}_t = \frac{1}{N} \sum_{n=1}^{N} \hat{\mathbf{p}}_{n,t} \tag{5.3}$$

as defined in (4.24) and T is the time elapsed. One localization RMSE is calculated for all S simulations with

$$e_{\mathrm{loc}} = \sqrt{\frac{1}{S} \sum_{s=1}^{S} e_{\mathrm{loc},s}^2}. \tag{5.4}$$

Compared to the RMSE for the localization, the RMSE for the mapping is calculated after the simulation has finished. It is defined as the RMSE of the three axes of the position of the feature particle $\hat{\mathbf{f}}_{n,k}^{(j)}$ to the real position of the feature $\mathbf{f}_j$. Therefore, the definition is formulated as

$$e_{\mathrm{map_{3D}},s} = \sqrt{(e_{\mathrm{map_x},s})^2 + (e_{\mathrm{map_y},s})^2 + (e_{\mathrm{map_z},s})^2}, \tag{5.5}$$

with the RMSEs of the axes given by

$$e_{\mathrm{map_x},s} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \frac{1}{J} \sum_{j=1}^{J} \frac{1}{K} \sum_{k=1}^{K} (\hat{f}_{\mathrm{x},n,k}^{(j)} - f_{\mathrm{x}}^{(j)})^2} \tag{5.6}$$

$$e_{\mathrm{map_y},s} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \frac{1}{J} \sum_{j=1}^{J} \frac{1}{K} \sum_{k=1}^{K} (\hat{f}_{\mathrm{y},n,k}^{(j)} - f_{\mathrm{y}}^{(j)})^2} \tag{5.7}$$

$$e_{\mathrm{map_z},s} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \frac{1}{J} \sum_{j=1}^{J} \frac{1}{K} \sum_{k=1}^{K} (\hat{f}_{\mathrm{z},n,k}^{(j)} - f_{\mathrm{z}}^{(j)})^2}. \tag{5.8}$$

Table 5.1: SLAM results of the two scenarios with different resampling noise for z.

	Grocery store	**Warehouse**
Number of agent particles N	100	100
Number of feature particles K	2500	2500
Number of features J	200	70
Anchor density $\frac{N_{\text{anchors}}}{J}$ (%)	10	10
IMU frequency f_{IMU} (Hz)	100	100
Feature frequency f_{feature} (Hz)	5	5
Estimator variance var(d) (m^2)	1	1
Number of runs N_{runs}	10	10
Change of height within runs h_{runs} (m)	0.1	0.1
Trajectory	Trajectory 2 in Figure 3.1	Trajectory 2 in Figure 3.1
Outlier detection	off	off
Detection radius warehouse r_{det} (m)	3.5	7

Additionally, a 2D RMSE for the mapping is defined as

$$e_{\text{map}_{\text{2D}},s} = \sqrt{(e_{\text{map}_{\text{x}},s})^2 + (e_{\text{map}_{\text{y}},s})^2}, \tag{5.9}$$

$$\tag{5.10}$$

just taking into account the x- and y-directions. The 2D mapping RMSE is used to analyse the influence of the missing movement in z-direction.
The two mapping RMSEs over all S simulations with different seeds can be calculated with

$$e_{\text{map}_{\text{3D}}} = \sqrt{\frac{1}{S}\sum_{s=1}^{S} e^2_{\text{map}_{\text{3D}},s}} \tag{5.11}$$

$$e_{\text{map}_{\text{2D}}} = \sqrt{\frac{1}{S}\sum_{s=1}^{S} e^2_{\text{map}_{\text{2D}},s}}. \tag{5.12}$$

In general, a simulation S consists of several runs. A run is defined as one simulated trajectory with a fixed height $\hat{p}_z$. Between the different runs the height is increased by $h_{\text{runs}} = 0.1\,\text{m}$. The following results were achieved with $N_{\text{runs}} = 10$ runs of different heights, where the starting position is known.
Furthermore, a standard parameter set is defined in Table 5.1. All the following simulations are conducted with the standard parameter set if not stated differently. Only the parameter to be evaluated is changed at a simulation. The choice of the individual parameters in the standard parameter set should become clearer in the sections that are analyzing the influence of the parameter.

Table 5.2: SLAM results of the two scenarios

	Grocery store		**Warehouse**	
	only localization	only mapping	only localization	only mapping
e_{loc} (m)	0.19	-	0.12	-
$e_{\text{map}_{\text{3D}}}$ (m)	-	0.23	-	0.55
$e_{\text{map}_{\text{2D}}}$ (m)	-	0.06	-	0.24
$e_{\text{map}_{\text{x}}}$ (m)	-	0.05	-	0.18
$e_{\text{map}_{\text{y}}}$ (m)	-	0.04	-	0.16
$e_{\text{map}_{\text{z}}}$ (m)	-	0.22	-	0.50

It is very hard to find appropriate parameters since many of the parameters influence each other. Changing two parameters simultaneously could result in no effect. For instance, the movement speed of the mobile agent and the frequency of the features. If the movement speed of the mobile agent and the frequency of the features are increased in the same manner, there is no observable effect. The positions of the mobile agent where measurements are received stay the same. Only the overall time of the trajectory decreases. In real world applications such a situation is not present since many parameter are fixed beforehand. Especially, most of the environment related parameters will be fixed. Therefore, the remaining parameters have to be adapted to the application and the required performance.

To get a lower limit on the RMSEs of the localization and the mapping, the CRLB could be used. Calculating the CRLB would involve the variance of the measurements, the position of the mobile agent at all time, and the position of the features. This calculation is very cumbersome and impractical. Therefore, to get benchmark performance results, the two problems that make up the SLAM problem are simulated independently, assuming perfect knowledge of the other part.
The localization part is simulated with anchors only[1]. Thus, imprecise features cannot interfere with the localization procedure. The other way around, the mapping part is simulated with known mobile agent positions. Thus, a poorly localized mobile agent cannot interfere with the mapping of the features. It should be noted that this limit only applies to the extended FastSLAM algorithm with the standard parameter set. For every change in the parameters the simulations have to be repeated.
The results can be seen in Table 5.2. The simulations show that the grocery store scenario achieves a better mapping RMSE but the warehouse scenario achieves a better localization RMSE. The difference of the localization RMSE is small and can be an artifact of the number of simulations. In contrast, the difference in mapping RMSE is more significant and cannot be neglected. The same observation is made in Section 5.2 where the two scenarios will be compared.

[1]All features are simulated as anchors

Table 5.3: SLAM results of the two scenarios with different resampling noise for z.

	Grocery store			**Warehouse**		
	$\mathbf{\Sigma}_{\mathrm{r}_1}$	$\mathbf{\Sigma}_{\mathrm{r}_2}$	$\mathbf{\Sigma}_{\mathrm{r}_3}$	$\mathbf{\Sigma}_{\mathrm{r}_1}$	$\mathbf{\Sigma}_{\mathrm{r}_2}$	$\mathbf{\Sigma}_{\mathrm{r}_3}$
e_{loc} (m)	0.57	0.47	0.55	0.45	0.47	0.49
$e_{\mathrm{map_{3D}}}$ (m)	0.79	0.57	0.62	0.79	0.77	0.77
$e_{\mathrm{map_{2D}}}$ (m)	0.52	0.42	0.50	0.39	0.50	0.43
$e_{\mathrm{map_x}}$ (m)	0.38	0.30	0.37	0.26	0.36	0.29
$e_{\mathrm{map_y}}$ (m)	0.36	0.30	0.33	0.29	0.35	0.32
$e_{\mathrm{map_z}}$ (m)	0.60	0.38	0.37	0.69	0.58	0.63

5.2 Scenario comparison

The most fundamental change in the environment can be attributed to the two scenarios, since they change the position of the features and their density. To get comparable results, the two scenarios are simulated with similar trajectories. Moreover, the detection radius for the grocery store scenario is set to 3.5 m to match the smaller size of the environment. Simulations were conducted with three different noise variances for the resampling of the feature particles. In the first simulation the noise for the resampling was chosen to be

$$\mathbf{\Sigma}_{\mathrm{r}_1} = \begin{bmatrix} \sigma^2_{\mathrm{x}_1} & 0 & 0 \\ 0 & \sigma^2_{\mathrm{y}_1} & 0 \\ 0 & 0 & \sigma^2_{\mathrm{z}_1} \end{bmatrix} = \begin{bmatrix} 0.01^2 & 0 & 0 \\ 0 & 0.01^2 & 0 \\ 0 & 0 & 0.02^2 \end{bmatrix}. \tag{5.13}$$

The noise regarding the z-axis was twice as high as the noise of the x- and y-axis. In the second simulation the chosen resampling noise was

$$\mathbf{\Sigma}_{\mathrm{r}_2} = \begin{bmatrix} \sigma^2_{\mathrm{x}_2} & 0 & 0 \\ 0 & \sigma^2_{\mathrm{y}_2} & 0 \\ 0 & 0 & \sigma^2_{\mathrm{z}_2} \end{bmatrix} = \begin{bmatrix} 0.01^2 & 0 & 0 \\ 0 & 0.01^2 & 0 \\ 0 & 0 & 0.01^2 \end{bmatrix}. \tag{5.14}$$

The noise has the same value for all three axes. In the thrid simulation the chosen resampling noise was

$$\mathbf{\Sigma}_{\mathrm{r}_3} = \begin{bmatrix} \sigma^2_{\mathrm{x}_3} & 0 & 0 \\ 0 & \sigma^2_{\mathrm{y}_3} & 0 \\ 0 & 0 & \sigma^2_{\mathrm{z}_3} \end{bmatrix} = \begin{bmatrix} 0.01^2 & 0 & 0 \\ 0 & 0.01^2 & 0 \\ 0 & 0 & 0.005^2 \end{bmatrix}. \tag{5.15}$$

In this case the noise regarding the z-axis was half the size of the other two axes. The rest of the parameters stay the same. The results of the three simulations are stated in Table 5.3.

The three scenarios show different results depending on the resampling noise. Especially the grocery store scenario is sensitive to the change in resampling noise of the z-axis. For the bigger resamling noise, both scenarios perform equally well in the overall 3D mapping.

The grocery store scenario performs slightly better than the warehouse scenario at the mapping for the z-axis. In turn, the warehouse scenario performs better at the mapping for the x- and y-axis, which is reflected in the localization. The 2D mapping influences the localization. This can be seen with the direct relation of the mapping RMSE $e_{\mathrm{map}_{\mathrm{2D}}}$ and the localization RMSE e_{loc}. If the 2D mapping RMSE worsens, then the localization RMSE also deteriorates by about the same value.
The results change with the smaller resampling noise for the z-axis. The 2D mapping RMSE stays more or less the same but the mapping RMSE of the z-axis get better. In particular, the grocery store scenario outperforms the warehouse scenario, in this aspect.
The difference in the performance has various reasons. First of all, the smaller environment in the grocery store combined with a shorter detection radius has an advantage in the initialization. The RO measurements are limited to smaller distances, where the same number of feature particles cover less volume. The smaller volume for the initialization helps to get smaller ranges for possible positions of the feature on the z-axis.
The bigger environment in the warehouse and the larger detection radius enables more distinct positions of the mobile agent since there is more room for movement. This movement helps to increase the mapping results for the x- and y-axes. The mapping for z-axis converges slower than the other two axes in both scenarios due to the missing movement of mobile agent in the z-direction.

Nevertheless, the findings of one scenario can also be applied to the other scenario. Furthermore, our simulations suggest that the grocery store scenario would achieve similar or even better results than the warehouse scenario with accurately adapted parameters. In addition, the warehouse scenario simulation needs considerably less time due to the fewer features. That is why, most of the simulations are conducted with the warehouse scenario. Also the smaller variance $\mathbf{\Sigma}_{\mathrm{r}_3}$ is used for the remaining simulations.

5.3 Convergence of the feature maps

An interesting aspect is the evolution of the mapping RMSE over time. In the last section it was recognized that the resampling noise has a big influence on the performance. Thus, the evolution of the mapping RMSE has also a close relation to the added noise after the resampling procedure discussed in Section 4.3.3. The added noise was introduced so that the particles do not stick with the initialization particles. The noise also serves a second purpose, it limits the minimal variance of the feature PF. A converged PF has at least a variance equal to the added noise after resampling. This means the PF cannot converge to one point and the feature particles change their positions a little each resampling step. Therefore, the feature PFs are still able to improve their representations. In Figure 5.1 the result of the grocery store scenario is visualized for each axis. In Figure 5.2 the results of the warehouse scenario is visualized for each axis. The vertical gray lines indicate the end of each run.

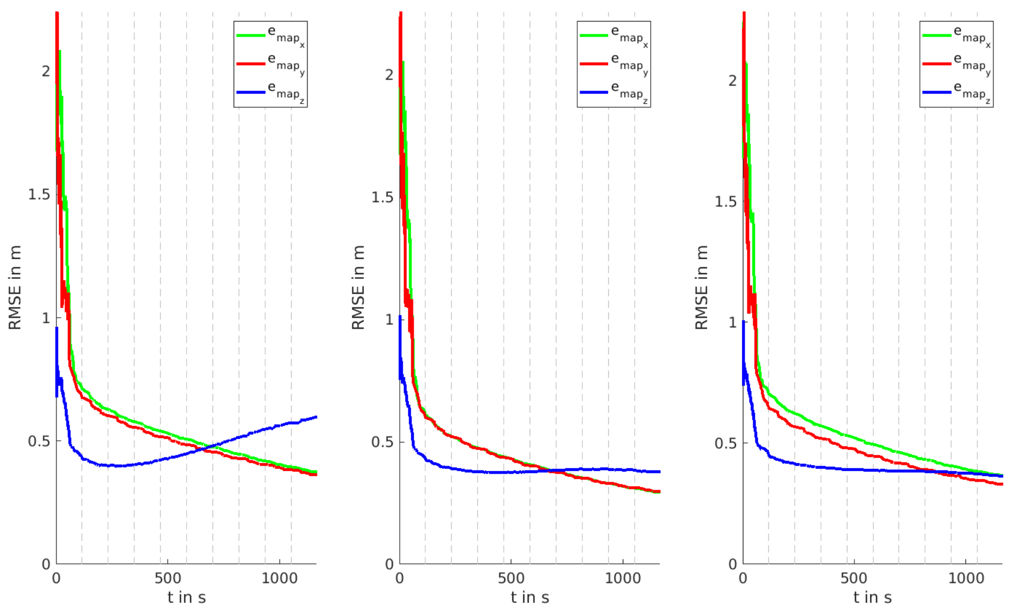


Figure 5.1: The RMSEs of the axes for the grocery store scenario. The left graph shows the results of the resampling variance $\mathbf{\Sigma}_{r_1}$. The middle graph shows the results of the resampling variance $\mathbf{\Sigma}_{r_2}$. The right graph shows the results of the resampling variance $\mathbf{\Sigma}_{r_3}$.

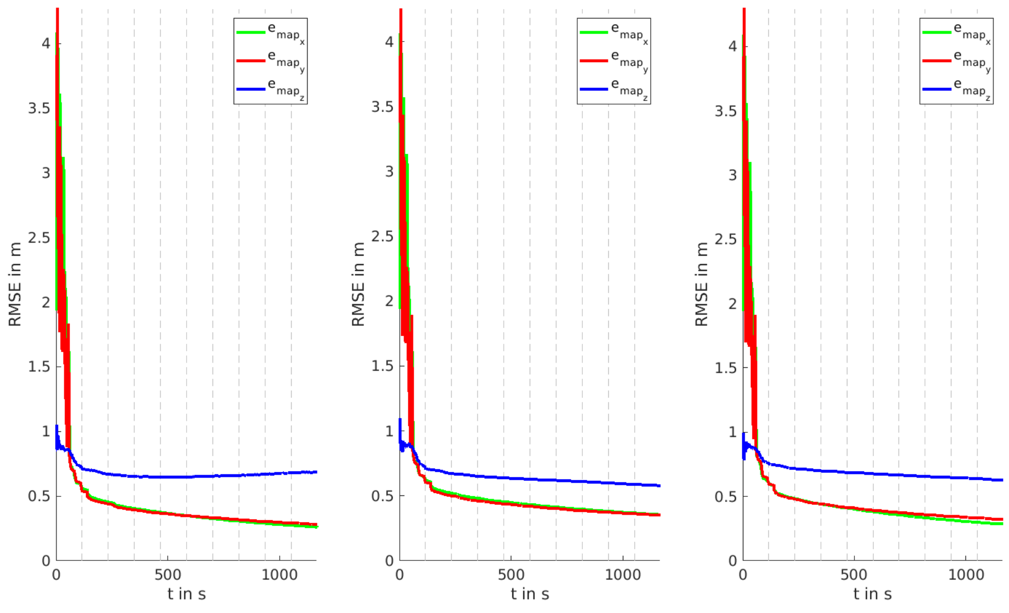


Figure 5.2: The RMSEs of the axes for the warehouse scenario. The left graph shows the results of the resampling variance $\mathbf{\Sigma}_{r_1}$. The middle graph shows the results of the resampling variance $\mathbf{\Sigma}_{r_2}$. The right graph shows the results of the resampling variance $\mathbf{\Sigma}_{r_3}$.

The RMSEs in both figures suggest that the feature particles converge to a point after approximately 1 run. This means the remaining 9 runs are used for improving the mapping. Both figures show that the mapping RMSE for the x- and y-axis decreases with more runs. This decrease gets smaller over time. With a bigger resampling noise for the z-axis an anomaly occurs. The mapping RMSE for the z-axis starts to diverge. This is especially apparent in the grocery store scenario, whereas in the warehouse scenario the z-axis also diverges but much slower.
Such a behavior can be explained with the information entropy of the feature PF. The ideal case would be that the information entropy decreases with each resampling step due to the integration of the information in the measurements. A problem occurs if the resampling procedure increases the information entropy more than the information in the measurements can decrease it. In this case, the particles would start to spread and diverge to a certain value. This is what happening to the mapping of the z-axis. The trajectory only has movement in the x- and y-direction. For that reason, the measurements include more information about these two axes.
With a smaller resampling noise for the z-axis the mapping RMSE improves slowly. In particular the grocery store scenario benefits from the smaller resampling noise since the z-axis RMSE stops diverging.

5.4 Trajectories

The movement of the mobile agent has a huge influence on the mapping results. Determining the 3D location of a feature from multiple perfect range measurements is simply a matter of finding where the corresponding spheres intersect. But even with non-noisy measurements and perfect localization, a movement in just one direction would yield four indistinguishable possible feature positions.
Unfortunately, perfect measurements are difficult to achieve in the real world. With noisy measurements the area of possible feature positions gets bigger. Hence, the trajectory of the mobile agent is crucial for the mapping of the features. The movement of the mobile agent can be controlled. Therefore, a good trajectory can be chosen to help the mapping. To determine a good trajectory, an investigation of different trajectories is necessary. For this investigation the three trajectories in Figure 3.1 are simulated.

The first trajectory is characterized by movement in one direction at a time. Exception are the corners and the turnaround. Otherwise, the trajectory consists mostly of movement solely in y-direction or x-direction. Hence, many features provide measurements to the mobile agent with the same x- and y-position. The second trajectory has a lot of combined movement in x- and y-direction. The mobile agent gets a lot of measurements from the features in different positions. The third trajectory has a similar movement pattern as the second trajectory, but the environment has one more corridor. The mobile agent never passes through the middle corridor, so, the features in the middle corridor provide

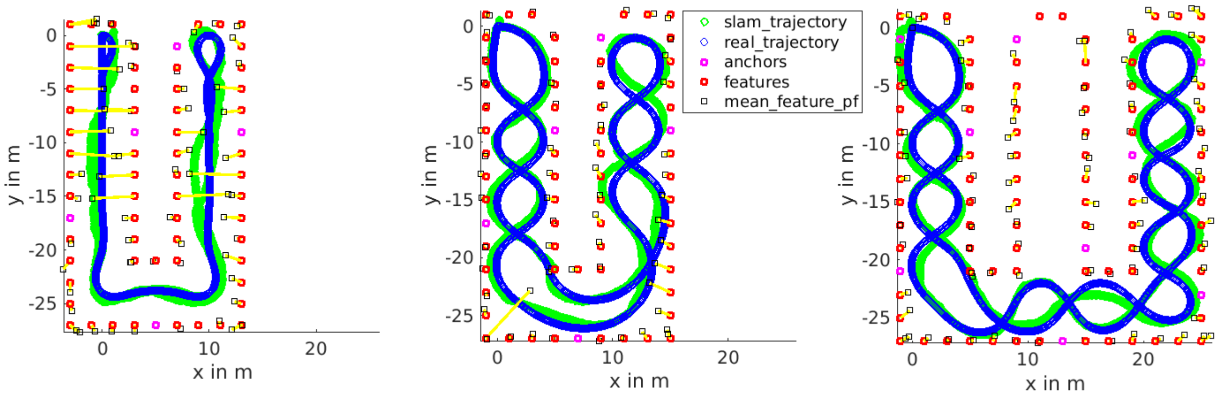


Figure 5.3: The three real and estimated trajectories with the real and estimated features after one run. The blue circles show the real trajectory. The green circles show the estimated trajectory. The magenta points depict the positions of the anchors. The red points depict the positions of the features. The black squares show the estimated feature positions.

Table 5.4: SLAM results of the different trajectories.

RMSE	**Trajectory 1**	**Trajectory 2**	**Trajectory 3**
e_{loc} (m)	1.16	0.49	0.58
$e_{\text{map}_{\text{3D}}}$ (m)	2.55	0.77	0.95
$e_{\text{map}_{\text{2D}}}$ (m)	2.22	0.43	0.59
$e_{\text{map}_{\text{x}}}$ (m)	2.12	0.29	0.43
$e_{\text{map}_{\text{y}}}$ (m)	0.67	0.32	0.39
$e_{\text{map}_{\text{z}}}$ (m)	1.25	0.63	0.75

measurements to mobile agent positions in the other two corridors, only.

A snapshot of the environment with the trajectories after one run can be seen in Figure 5.3. Moreover, also the feature and their corresponding PF estimates are visualized. For each graph the origin of the coordinate system is the starting point of the trajectory. The RMSEs for 100 simulations are stated in Table 5.4 and reflect the insights from the snapshots.
The first trajectory leads to problems with the mapping of the x-axis of some features since they are positioned where the mobile agent only move in the y-direction. All the features around the corners and the turnaround converge to a single position.
In the simulation with the second trajectory, the features PFs show better results. The combined movement in both axes lets all the feature particles converge close to the real features. One feature in the lower left corner is not converged yet. This feature is the furthest away from the path of the mobile agent and provides the fewest measurements. Also some feature PFs in the lower right have problems due to the positions of the trajectory. Other than that, the mapping of the remaining features is working well.
The third trajectory shows the best convergence because this trajectory has the most turns. For this reason the trajectory has the most distinct positions of the mobile agent. The PFs

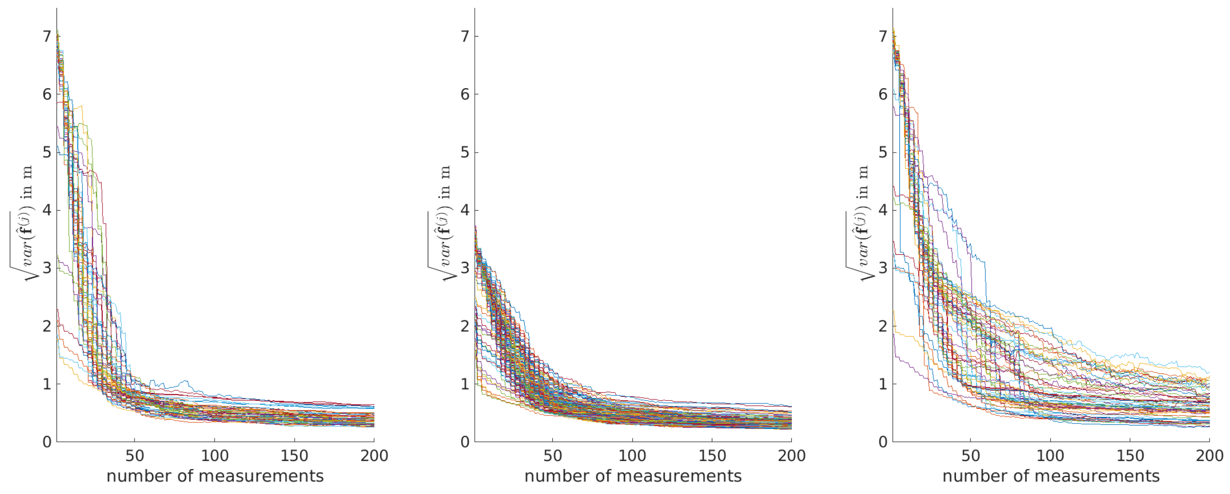


Figure 5.4: The standard deviation of the feature PFs over the number of measurements. The left graph shows the standard deviations for the warehouse scenario with the second trajectory. The middle graph shows the standard deviations for the grocery store scenario with the second trajectory. The right graph shows the standard deviations for the warehouse scenario with the first trajectory.

in the middle corridor are mapped slightly worse than the rest, since they provide fewer and more distant measurements. These are also the features that make up the difference of the results between the second and the third trajectory.
One issue that affects all trajectories is the quality of the estimated trajectory. If the estimated trajectory does not match the real one in some direction, the initialized feature particles are also mismatched in the same manner. With many realizations the second trajectory achieves the best results. Hence, this trajectory is used for further simulations.

5.5 Convergence of feature particles

The convergence was already mentioned in Section 5.3. In this context, the convergence is shortly analyzed with the number of runs and the mapping RMSE. Another interesting aspect is to look at the convergence behavior of the feature PFs regarding the number of measurements. Every feature provides a different number of measurements depending on the trajectory, detection radius and the layout of the environment. Thus, the best way to analyze the convergence behavior of the feature PFs is to look at the standard deviations of PFs with the same number of provided measurements. If all the particles are positioned within a small volume, the PF has converged. This definition is used to analyze the average number of measurements for the convergence of the feature PF.

Figure 5.4 shows the development of the standard deviations of the different feature PFs over the number of measurement they have provided. It can be seen that the standard

Table 5.5: SLAM results of the different estimator variances.

RMSE	$\mathbf{var}(d_1) = (1m)^2$	$\mathbf{var}(d_2) = (0.1m)^2$
e_{loc} (m)	0.49	0.18
$e_{\text{map}_{\text{3D}}}$ (m)	0.77	0.50
$e_{\text{map}_{\text{2D}}}$ (m)	0.43	0.28
$\text{var}(e_{\text{map}_{\text{3D}}})$ (m^2)	$(0.13)^2$	$(0.30)^2$

deviations have steps where they stay the same. That occurs because of the resampling after M measurements discussed in Section 4.3.3. The results show that all the particles have converged within 100 measurements for the second trajectory which can be seen in the middle graph of Figure 5.4. The middle graph also indicates that the feature particles in the grocery store scenario converge slightly slower compared to the warehouse scenario. Furthermore, it can be seen that the warehouse scenario has a higher initialization standard deviation since the detection radius was twice the range of the grocery store scenario. However, regardless of the initialization standard deviation, the feature PFs seem to converge to roughly the same standard deviation. The trajectory of the mobile agent has a huge impact on the convergence behavior. The right graph in Figure 5.4 shows that even after 200 measurements some features have not fully converged. This observation is consistent with the findings in Section 5.4. The features which provide measurements to mobile agent positions with a change in only one direction converge to two possible positions.

5.6 Estimator variance

In Section 3.2.1 two variances were derived for the distance estimator. The quality of the variance is mostly determined by the bandwidth since it reduces the estimator variance as a quadratic term. The SNR effects the variance calculation only in a linear manner. Therefore the analysis of different estimator variances is mainly about the influence of the bandwidth. Measurements with these two variances are simulated. The results of warehouse scenario with the standard parameter set can be seen in Table 5.5.

Figure 5.5 shows the individual simulations of the SLAM problem and the mapping-only problem over the seeds. An interesting phenomenon was observed. The variance of the RMSEs, given by

$$\text{var}(e_{\text{map}_{\text{3D}}}) = \frac{1}{S}\sum_{s=1}^{S}(e_{\text{map}_{\text{3D}}} - e_{\text{map}_{\text{3D}},s})^2, \tag{5.16}$$

with a better estimator variance is higher compared to those with a worse one. This can be clearly recognized by the red spikes of the 3D mapping error. This red spikes show a significantly worse mapping RMSE for some simulations. The mapping-only problem shows

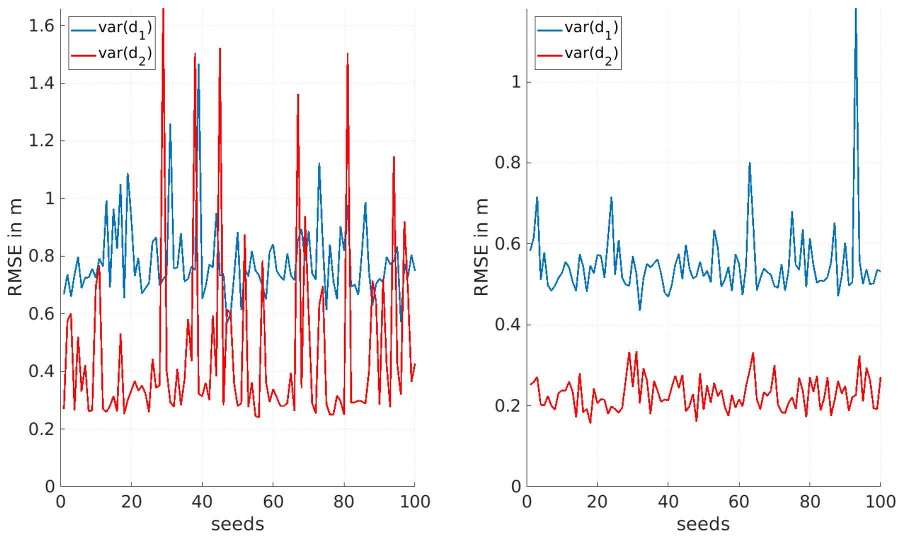


Figure 5.5: The left graph shows the mapping RMSE for different estimator variances of the SLAM problem over different seeds. The right graph shows the mapping RMSE for the mapping-only problem over different seeds.

almost no such spikes.
In general, with the higher estimator variance the SLAM problem tends to be prone to outliers. The simulations for the mapping-only problem show no such outliers. Therefore, it can be concluded that a poor localization leads to wrong convergence of individual feature PFs. The reason for this is the annular initialization of the feature particles. More precisely, the reason is the radius of the spherical coordinate system. The equation for the radius is given by

$$r = \mathcal{N}(z_t^{(j)}, \mathrm{var}(d)). \tag{5.17}$$

A closer look at (5.17) reveals that the RO measurement defines the radius of the ring and the variance defines the annular size. The annular size can be seen as the deviation from the radius of the ring where feature particles are initialized. For a normal distribution 99.7 % of the feature particles are positioned within the 3σ-interval. The higher the variance, the bigger this interval and vice versa. For the larger variance $\mathrm{var}(d_1)$ this interval is quantified as ± 3 m. This means the initialization covers a larger volume due to the measurement with a higher noise. However, the larger volume makes it also more robust to deviations from the localization. For the smaller variance $\mathrm{var}(d_2)$ the interval is only ± 0.3 m. This means if the localization is off, the feature particles get initialized at the wrong place. Moreover, several badly initialized feature particles pull the localization of future runs towards their direction and reduce the influence of the anchors. This phenomenon can be remedied by increasing the variance of the radius of the initialization.

If this phenomenon is neglected, general trends can be recognized. Especially, the lo-

Table 5.6: SLAM results of the different IMUs.

RMSE	Random walk	IMU in [42]	MTi 1 series	iPhone 4
e_{loc} (m)	10.03	0.49	0.48	3.06
$e_{\text{map}_{3\text{D}}}$ (m)	11.27	0.77	0.84	3.29
$e_{\text{map}_{2\text{D}}}$ (m)	11.19	0.43	0.41	3.00

calization is improved with better measurements. This can be explained that more precise measurements make it easier to rate the agent particles. Therefore, only the best agent particles get resampled. However, this behavior makes it easier for badly initialized feature particles to alter the estimated path. Nonetheless, the feedback from the mean position of the agent particles is better on average and subsequently, the correction of the drifting error in the IMU works better. A more precise localization combined with a smaller initialization volume naturally improves the mapping.

5.7 Inertial measurement unit

Three different IMUs were introduced in Section 3.1.2. The parameter of the three different IMUs were simulated and compared to a random walk configuration. For the random walk, the motion model was changed to

$$\hat{\mathbf{p}}_{n,t+1} = \hat{\mathbf{p}}_{n,t} + \mathcal{N}(\mathbf{0}, \Sigma_{\text{random\_walk}}). \tag{5.18}$$

The variance of the added normal distributed noise $\Sigma_{\text{random\_walk}}$ was chosen to be bigger compared to the noise of the IMU motion model. This was done to account for the missing change in position $\Delta\hat{\mathbf{p}}_t$. The result can be seen in Table 5.6.
The two IMUs, besides the inertial sensors of the iPhone 4, achieve similar results. With a certain quality of the IMU, the RMSEs stay roughly the same. This can be interpreted that using a higher quality IMU would not necessarily contribute to the improvement of the RMSE in the same way. Therefore, it would be better to apply a sensor fusion with other sensor types used for dead reckoning to further improve the localization.
The inertial sensors used in the iPhone are not suitable for solving the initialization problem. While they still achieve better results than the random walk approach, it is the worst out of the three tested IMUs. Especially, the bias of the gyroscope is too big to get corrected. The weak point in the orientation gets reduced due to the help of the simulated magnetometer, since the same magnetometer is used for all the IMUs.
It can be assumed that in reality, there are more possibilities to improve the localization. For example, in [55] a gyroscope, wheel encoders, and a camera are used to localize the robot. Similar results as in our simulation are obtained there.

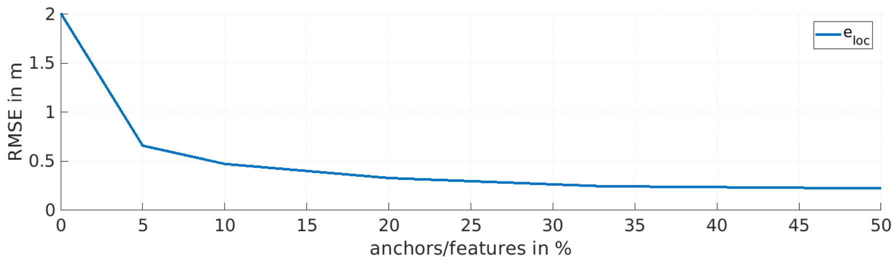


Figure 5.6: The localization RMSE of the SLAM problem over different anchor densities.

5.8 Anchors

One essential part of the environment are the anchors. They are needed to prevent the localization from divergence, especially for the drifting bias of the IMU. Thus, it is necessary to analyze the influence of the density of anchors. Six different densities from 5 % to 50 % were evaluated. The results for the localization can be seen in Figure 5.6 and the results for the mapping in Figure 5.7.

The increase of the number of anchors is just useful to a certain percentage. From 20 % to 50 % only minor improvements of the RMSE are achieved.
In the beginning of each run, the mobile agent is well localized because the starting position is known. Therefore, the localization RMSE is kept within limits. Also the mapping works for the first sensors even with no anchors. The effects of having no anchors would be worse with a bigger environment or a longer trajectory.

It is not realistic to have have a higher anchors density than 10 %. In the warehouse scenario such a percentage would correspond to an anchor positioned every 20 m on the circumference of the walls. In the grocery store scenario this percentage would be even lower since the feature density is higher and the positions of the anchors have to be determined by hand. This would mean a lot of anchors would have to be mapped manually.
In addition, the detection radius and the feature density influences the needed percentage of anchors. As long as enough anchors are providing measurements to the mobile agent, the localization procedure will work. It does not matter if this number is obtained by having a high anchor density, a big detection radius or a high feature density.

The anchors are needed for the localization part. They have only an indirect influence on the mapping part. A different type of feature could be used for a more practical realization of the anchors. Another reason to use a different type of feature for the anchors is the impracticality to measure the exact position of many ESL. As an alternative, ArUco markers [56] could be used. A calibrated camera is able to detect the ID and estimate the marker's 6DoF pose. With known position, an ArUco marker could provide more information than a shelf label.

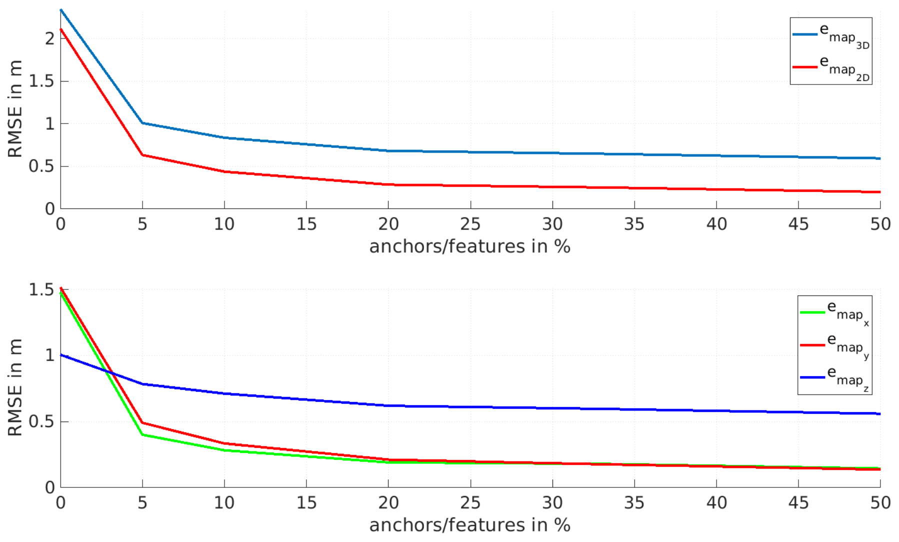


Figure 5.7: The mapping RMSEs of the SLAM problem over different anchor densities.

5.9 Feature particle reduction algorithms

In Section 4.3.5 different approaches to reduce the number of feature particles K are introduced. The performance of the different approaches are compared with the same setup and the standard parameter set. The algorithms are evaluated for the two different resampling variances. The results are stated in Tables 5.7 and 5.8.

The best results are achieved by the extended FastSLAM. This was expected since a higher number of particles leads to a more accurate representation of the posterior distribution. This accuracy comes at the cost of a higher computational effort which can be seen from the computation time t_{comp} in both tables. This computational effort can be brought down a lot if the feature PFs are switched to EKFs after convergence. Such an approach combines the PF as a solution to the initialization with the EKF as a more computationally efficient feature representation.
Moreover, the EKF versions of the reduction algorithms achieve a better performance. This is especially apparent with the bigger feature resampling variance $\mathbf{\Sigma}_{\text{r}_1}$. There are two reasons for this behavior. The z-axis is not diverging and it is better to switch back to the EKF representation than having few particles to cover the posterior distribution. The only exception is the extended FastSLAM algorithm using the smaller resample variance $\mathbf{\Sigma}_{\text{r}_3}$ where none of those two reasons are apparent.

Furthermore, the PF converges faster with the smaller resampling variance. This can be seen if the computation time t_{comp} of the KLD-sampling and the EKF versions of both tables are compared. The computation time for KLD-sampling and all the EKF versions

Table 5.7: SLAM results of the particles reduction algorithms with $\mathbf{\Sigma}_{\mathrm{r}_1}$.

	Extended FastSLAM		**Likelihood-Sampling**		**KLD-Sampling**	
	only particles	EKF	only particles	EKF	only particles	EKF
e_{loc} (m)	0.45	0.51	1.00	0.84	0.64	0.61
$e_{\mathrm{map}_{3\mathrm{D}}}$ (m)	0.79	0.77	1.48	1.37	1.10	1.07
$e_{\mathrm{map}_{2\mathrm{D}}}$ (m)	0.39	0.48	1.11	1.01	0.66	0.66
$e_{\mathrm{map}_{\mathrm{x}}}$ (m)	0.26	0.33	0.84	0.70	0.47	0.44
$e_{\mathrm{map}_{\mathrm{y}}}$ (m)	0.29	0.34	0.72	0.72	0.47	0.49
$e_{\mathrm{map}_{\mathrm{z}}}$ (m)	0.69	0.60	0.98	0.93	0.89	0.84
t_{comp} (s)	28857	10045	2293	1823	10467	9800

Table 5.8: SLAM results of the particles reduction algorithms with $\mathbf{\Sigma}_{\mathrm{r}_3}$.

	Extended FastSLAM		**Likelihood-Sampling**		**KLD-Sampling**	
	only particles	EKF	only particles	EKF	only particles	EKF
e_{loc} (m)	0.49	0.57	0.93	0.88	0.92	0.62
$e_{\mathrm{map}_{3\mathrm{D}}}$ (m)	0.77	0.86	1.29	1.33	1.18	0.95
$e_{\mathrm{map}_{2\mathrm{D}}}$ (m)	0.43	0.56	1.01	1.05	0.91	0.66
$e_{\mathrm{map}_{\mathrm{x}}}$ (m)	0.29	0.39	0.74	0.74	0.62	0.46
$e_{\mathrm{map}_{\mathrm{y}}}$ (m)	0.32	0.40	0.68	0.75	0.67	0.47
$e_{\mathrm{map}_{\mathrm{z}}}$ (m)	0.63	0.65	0.81	0.81	0.75	0.68
t_{comp} (s)	29024	8292	2003	1361	6230	4798

decreases while the computation time of the remaining algorithms stays roughly the same. That can only be the case if the KLD-Sampling uses less particles throughout the whole simulation and the switch to the EKF representation happens earlier.
The likelihood approach attains the biggest speed up but also the worst results. Although, the likelihood-sampling starts with the same number of particles than the rest, the number of feature particles are reduced too fast. The KLD-sampling does a better job than the likelihood-sampling but it is still worse than the extended FastSLAM with fixed number of feature particles. This means, that a certain number of particles is beneficial at all times.

The EKF version of the extended FastSLAM offers the best trade-off between accuracy and computational effort. It comes close to the computational effort of the KLD-sampling but achieves better results. Therefore, this version could be an alternative to the basic version of the extended FastSLAM. The lack in performance can be counteracted with the usage of a bigger number of feature particles in the beginning.

5.10 Outlier detection

To test the outlier detection introduced in Section 4.3.7 an already preinitialized environment is assumed. The setup is initialized that it roughly matches the results of the extended FastSLAM in Table 5.7. To determine the variance of the feature PF, the findings in Section 5.5 are used. A feature PF is initialized as

$$\hat{\mathbf{f}}_{t_0}^{(j)} = \mathbf{f}^{(j)} + \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_j), \tag{5.19}$$

where the feature covariance matrix is given by

$$\mathbf{\Sigma}_j = \begin{bmatrix} e_{\mathrm{map_{3D}},\mathrm{x}} & 0 & 0 \\ 0 & e_{\mathrm{map_{3D}},\mathrm{y}} & 0 \\ 0 & 0 & e_{\mathrm{map_{3D}},\mathrm{z}} \end{bmatrix}. \tag{5.20}$$

This is the bias part of the mapping RMSE. Furthermore, the individual feature particles are initialized as

$$\hat{\mathbf{f}}_{n,k,t_0}^{(j)} = \hat{\mathbf{f}}_{t_0}^{(j)} + \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathrm{cov},n}^{(j)}), \tag{5.21}$$

where $\mathbf{\Sigma}_{\mathrm{cov},n,t}^{(j)}$ is the feature covariance matrix. This is the variance part of the mapping RMSE. After the initialization of the feature particles, two predetermined features are misplaced by the distance d_{mp} on the shelf. The equations for the misplacement can be

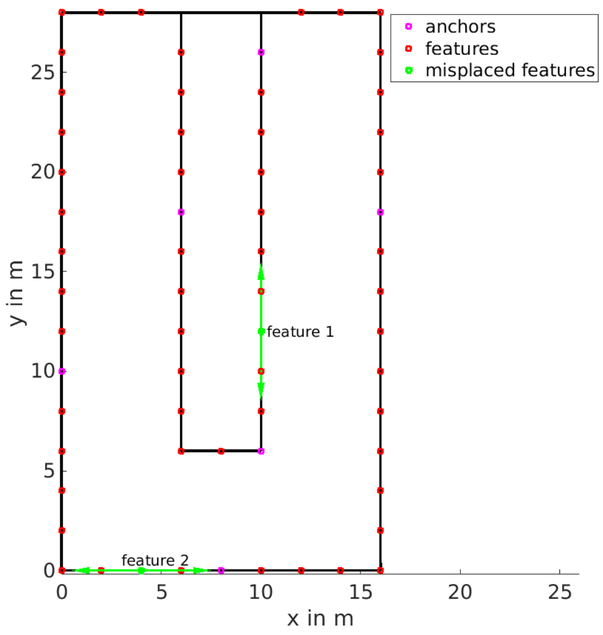


Figure 5.8: The used environment to evaluate the outlier detection. The green point depicts the positions of the two chosen features. The green arrows show the possible misplacements.

written as

$$\mathbf{f}^{(1)} = \mathbf{f}^{(1)} - \frac{\hat{f}^{(1)}_{y,t_0}}{|\hat{f}^{(1)}_{y,t_0}|} \begin{bmatrix} 0 \\ d_{\mathrm{mp}} \\ 0 \end{bmatrix} \tag{5.22}$$

$$\mathbf{f}^{(2)} = \mathbf{f}^{(2)} - \frac{\hat{f}^{(2)}_{x,t_0}}{|\hat{f}^{(2)}_{x,t_0}|} \begin{bmatrix} d_{\mathrm{mp}} \\ 0 \\ 0 \end{bmatrix} . \tag{5.23}$$

Two different features were chosen to analyze influences of the position of a feature. Moreover, the two chosen features were placed farther from the known starting position to include the effect of a flawed localization. Figure 5.8 illustrates the two chosen features and their possible misplacements.

Figure 5.9 shows the results of the two features. The percentage of detection indicates that the outlier detection method works equally well for both features. A difference can be seen in the mapping RMSE. The first feature achieves a better performance than the second feature. This difference can be explained with a more favorably positioned first feature compared to the second one. As a consequence, the first feature provides more measurements for distinct positions of the mobile agent and converges towards the misplaced feature position even if the misplacement is not detected.
One problem with this outlier detection method is that it is prone to false positives. The high variance of the measurements and a bad localization of the mobile agent can interfere with the detection procedure. So far, this approach can only deal with outliers in the PF

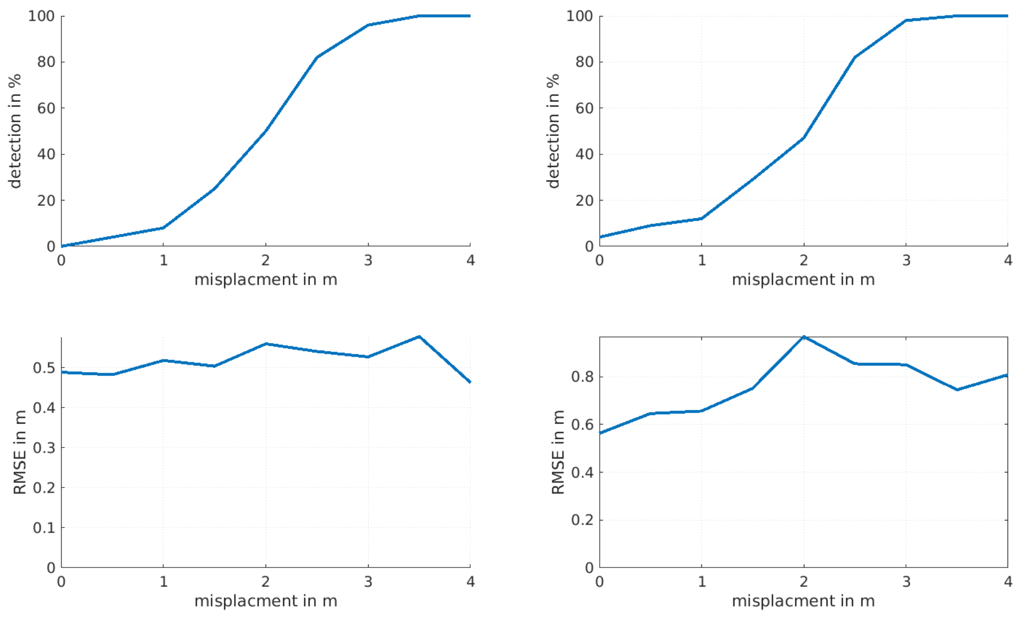


Figure 5.9: The results for the outlier detection of the first (left) and the second feature (right). The upper graphs show the detection percentage over the misplacement distance. The lower graphs show the mapping RMSE over the misplacement distance.

representation. The evaluation of the outlier detection for the EKF representation implies that the EKF converges towards the misplaced feature. The question is now whether an outlier detection method for the EKF representation is necessary at all.

5.11 Number of agent particles

The number of agent particles N scales linearly with the computational effort. This means reducing the number of agent particles N reduces the computational effort in the same manner and vice versa. It is still unclear how the number of agent particles affects the results in terms of the RMSEs. Therefore, a simulation was conducted with different numbers of agent particles from 10 up to 300. The simulations consist of only 5 runs instead of 10 to save time. Therefore, the results are a little worse compared to similar results discussed so far. In Figure 5.10 the outcome of the simulations can be seen.

The figure implies that the number of agent particles influences the localization and the mapping in a very similar way since both decrease in the same manner. Furthermore, the results show that a higher number of agent particles achieves better results. This gain in the performance decreases with a higher number of agent particles. The biggest improvement happens from using only 10 agent particles to about 50 agent particles. There

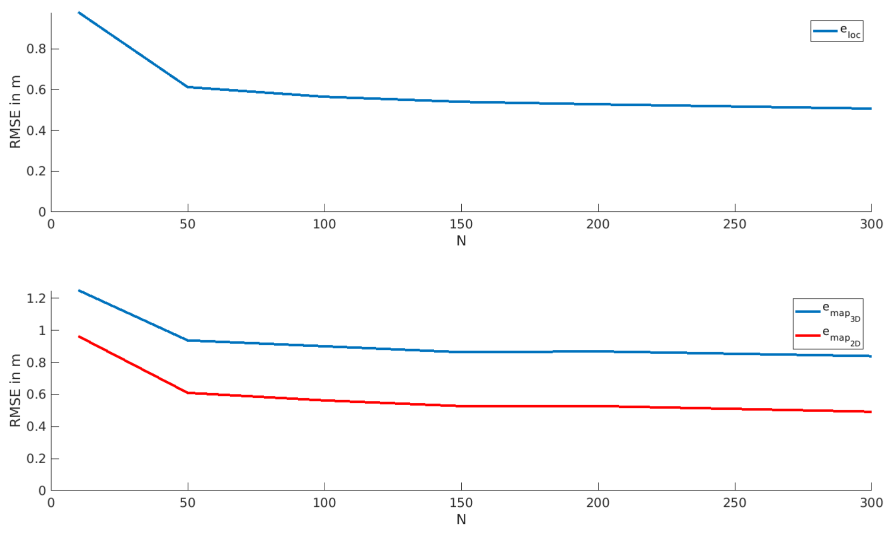


Figure 5.10: The localization and mapping RMSEs of the SLAM problem over different agent particles numbers.

is still a reasonable improvement in the performance from 50 to 150. Afterwards, the improvement seems to flatten more and more with a higher number of agent particles. Hence, a break-even point between performance and computational effort is found around a number of 100 agent particles.

6 Conclusion

A SLAM algorithm has been proposed that can handle the initialization problem arising from the RO measurements. The approach is based on the factored solution to the SLAM problem by M. Montemerlo [25]. This extension to the FastSLAM uses a PF representation instead of the EKF representation for the features. A solution to switch back to the EKF representation has been shown. This switch does worsen the results slightly but speeds up the algorithm by a factor of 3. Furthermore, a method to detect outliers has been developed.

The evaluations within the simulation environment demonstrate that the algorithm is able to solve the initialization problem. An important consideration regarding the initialization is the choice of the trajectory. Beyond that, the algorithm improves the positions of the features over time. Thus, also the localization of the mobile agent gets more precise. From that observation, it can be concluded that the localization and the mapping are conditionally dependent on each other. Improving the performance of one of those two, improves also the performance of the other one.

Moreover, the various parameters and their influence have been analyzed in many simulations. From the findings, requirements on the capabilities of the mobile agent and the features can be derived. Thus, basic preconditions for a certain accuracy can be estimated, e.g, the needed localization accuracy for the mobile agent or the required bandwidth for the range measurements.

6.1 Outlook

A logical next step would be to construct an experimental setup and take RO measurements from similar hardware used in the ESL. With real-world observations the measurement model can be adapted. Thus, the evaluation of the SLAM algorithm is more informative with a more realistic measurement model. Moreover, the robustness of the algorithm will be tested if real measurements are used instead of synthetic ones.

Future work should also deal with the problem that the first run of the initialization influences all subsequent runs. The reason for this influence are wrongly initialized features that pull the position of the mobile agent towards their location. The wrongly initialized features mainly originate from the errors of the IMU and the shape of trajectories which have a bad dilution of precision in certain parts of the environment. Many independent runs with varying trajectories can be combined to overcome this issue.

The findings in Section 5.3 indicate that the results improve very slowly after the convergence of the feature PFs. Different approaches have to be applied to accelerate the improvement of the results. Moreover, the correspondence of the measurements to the feature are known and could be used as a source for additional information. Just the availability of a measurement from a feature at a certain time instance carries information about the position of the mobile agent. Therefore, it would make sense to use approaches using optimization strategies over several measurements, e.g., Graph-based SLAM algorithms. This would utilize the correspondence of the measurements to make indirect connections between agent positions. Furthermore, such an approach can be easily extended with measurements between features.

Incorporating a-priori knowledge of the environment can help the optimization strategies to achieve better results. A floor plan could help to restrict the possible positions of the mobile agent and discretize the positions of a feature. Especially, the mapping of the z-axis still needs improvement, since its contribution to the mapping error is highest. Such a discretization can be easily applied to the positions for the z-axis since the ESLs can only be mounted on racks with certain heights. Therefore, each feature can only be positioned in certain positions on the z-axis. A problem would be that this positions have to be determined for each application scenario, for example, a grocery store has different possible heights compared to a warehore. Another disadvantage is that the noise is then split between the x- and y-axis which would probably decrease their mapping results.

Bibliography

[1] S. Alletto, R. Cucchiara, G. Del Fiore, L. Mainetti, V. Mighali, L. Patrono, and G. Serra, "An indoor location-aware system for an iot-based smart museum," *IEEE Internet of Things Journal*, vol. 3, no. 2, pp. 244–253, 2015 (cit. on p. 1).

[2] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, "Recent advances in indoor localization: A survey on theoretical approaches and applications," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1327–1346, 2016 (cit. on p. 1).

[3] G. Acampora, D. J. Cook, P. Rashidi, and A. V. Vasilakos, "A survey on ambient intelligence in healthcare," *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2470–2494, 2013 (cit. on p. 1).

[4] *Urus: Ubiquitous networking robots in urban settings*, EU Project, 2009 (accessed August 09, 2019). [Online]. Available: `https://www.iri.upc.edu/project/show/59` (cit. on p. 1).

[5] N. Sünderhauf, "Robust optimization for simultaneous localization and mapping," PhD thesis, Technischen Universitat Chemnitz, 2012, ch. 2 (cit. on pp. 4, 6).

[6] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010 (cit. on p. 4).

[7] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," *Springer handbook of robotics*, pp. 871–889, 2008 (cit. on p. 5).

[8] 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, 2006 (cit. on p. 5).

[9] D. Hai, Y. Li, H. Zhang, and X. Li, "Simultaneous localization and mapping of robot in wireless sensor network," in *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, IEEE, vol. 3, 2010, pp. 173–178 (cit. on p. 5).

[10] F. R. Fabresse, F. Caballero, I. Maza, and A. Ollero, "Undelayed 3d ro-slam based on gaussian-mixture and reduced spherical parametrization," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 1555–1561 (cit. on p. 5).

[11] G. Vallicrosa and P. Ridao, "Sum of gaussian single beacson range-only localization for auv homing," *Annual Reviews in Control*, vol. 42, pp. 177–187, 2016 (cit. on p. 5).

[12] A. Torres-González, J. R. Martinez-de Dios, and A. Ollero, "Range-only slam for robot-sensor network cooperation," *Autonomous Robots*, vol. 42, no. 3, pp. 649–663, 2018 (cit. on p. 5).

[13] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006 (cit. on p. 5).

[14] G. Grisetti, C. Stachniss, and W. Burgard, "Nonlinear constraint network optimization for efficient map learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 428–439, 2009 (cit. on p. 6).

[15] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006 (cit. on p. 6).

[16] T. Duckett, S. Marsland, and J. Shapiro, "Fast, on-line learning of globally consistent maps," *Autonomous Robots*, vol. 12, no. 3, pp. 287–300, 2002 (cit. on p. 6).

[17] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 196–207, 2005 (cit. on p. 6).

[18] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 3607–3613 (cit. on p. 6).

[19] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006 (cit. on p. 6).

[20] M. Kaess, A. Ranganathan, and F. Dellaert, "Isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008 (cit. on p. 6).

[21] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "Isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012 (cit. on p. 6).

[22] K. Konolige, "Large-scale map-making," in *AAAI*, 2004, pp. 457–463 (cit. on p. 6).

[23] A. Eliazar and R. Parr, "Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *IJCAI*, Acapulco, Mexico, vol. 3, 2003, pp. 1135–1142 (cit. on p. 6).

[24] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *IJCAI*, 2003, pp. 1151–1156 (cit. on p. 6).

[25] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002 (cit. on pp. 6, 22, 61).

[26] N. M. Kwok, G. Dissanayake, and Q. P. Ha, "Bearing-only slam using a sprt based gaussian sum filter," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, IEEE, 2005, pp. 1109–1114 (cit. on p. 6).

[27] G. De Chambrier and A. Billard, "Non-parametric bayesian state space estimator for negative information," *Frontiers in Robotics and AI*, vol. 4, p. 40, 2017 (cit. on p. 6).

[28] S. Thrun, D. Koller, Z. Ghahmarani, and H. Durrant-Whyte, "Slam updates require constant time," in *Workshop on the Algorithmic Foundations of Robotics*, Citeseer, 2002 (cit. on p. 6).

[29] S. Holmes, G. Klein, and D. W. Murray, "A square root unscented kalman filter for visual monoslam," in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 3710–3716 (cit. on p. 6).

[30] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, International Society for Optics and Photonics, vol. 3068, 1997, pp. 182–194 (cit. on p. 6).

[31] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960 (cit. on p. 7).

[32] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter," 1995 (cit. on p. 8).

[33] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," 2004 (cit. on p. 8).

[34] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002 (cit. on pp. 11, 38).

[35] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 656-704, p. 3, 2009 (cit. on p. 11).

[36] O. J. Woodman, "An introduction to inertial navigation," University of Cambridge, Computer Laboratory, Tech. Rep., 2007 (cit. on pp. 12, 13).

[37] E. Edwan, "Novel approaches for improved performance of inertial sensors and integrated navigation systems," 2013 (cit. on pp. 13, 17).

[38] M. Garaus, E. Wolfsteiner, and U. Wagner, "Shoppers' acceptance and perceptions of electronic shelf labels," *Journal of Business Research*, vol. 69, no. 9, pp. 3687–3692, 2016 (cit. on p. 13).

[39] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, 2019 (cit. on p. 14).

[40] D. Lymberopoulos and J. Liu, "The microsoft indoor localization competition: Experiences and lessons learned," *IEEE Signal Processing Magazine*, vol. 34, no. 5, pp. 125–140, 2017 (cit. on p. 14).

[41] S. M. Kay, *Fundamentals of statistical signal processing.* Prentice Hall PTR, 1993 (cit. on pp. 15, 19).

[42] P. Davidson, J. Hautamäki, and J. Collin, "Using low-cost mems 3d accelerometer and one gyro to assist gps based car navigation system," in *Proceedings of 15th Saint Petersburg international conference on integrated navigation systems*, 2008 (cit. on pp. 18, 19, 53).

[43] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using allan variance," *IEEE Transactions on instrumentation and measurement*, vol. 57, no. 1, pp. 140–149, 2007 (cit. on p. 18).

[44] V. Renaudin, M. H. Afzal, and G. Lachapelle, "Complete triaxis magnetometer calibration in the magnetic domain," *Journal of sensors*, vol. 2010, 2010 (cit. on p. 18).

[45] *MTi 1-series data sheet*, Xsens Technologies B.V., 2019 (accessed August 5, 2019). [Online]. Available: `https://xsens.com/download/pdf/documentation/mti-1/MTi-1-series-datasheet.pdf` (cit. on p. 19).

[46] X. Niu, Q. Zhang, Y. Li, Y. Cheng, and C. Shi, "Using inertial sensors of iphone 4 for car navigation," in *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, IEEE, 2012, pp. 555–561 (cit. on p. 19).

[47] K. Witrisal, P. Meissner, E. Leitinger, Y. Shen, C. Gustafson, F. Tufvesson, K. Haneda, D. Dardari, A. F. Molisch, A. Conti, *et al.*, "High-accuracy localization for assisted living: 5g systems will turn multipath channels from foe to friend," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 59–70, 2016 (cit. on p. 19).

[48] W. Hörmann, J. Leydold, and G. Derflinger, *Automatic nonuniform random variate generation.* Springer Science & Business Media, 2013 (cit. on p. 31).

[49] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies," *IEEE Signal processing magazine*, vol. 32, no. 3, pp. 70–86, 2015 (cit. on p. 34).

[50] R. Douc and O. Cappé, "Comparison of resampling schemes for particle filtering," in *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, IEEE, 2005, pp. 64–69 (cit. on p. 34).

[51] C. Gentner, S. Zhang, and T. Jost, "Log-pf: Particle filtering in logarithm domain," *Journal of Electrical and Computer Engineering*, vol. 2018, 2018 (cit. on p. 37).

[52] J. S. Liu and R. Chen, "Sequential monte carlo methods for dynamic systems," *Journal of the American statistical association*, vol. 93, no. 443, pp. 1032–1044, 1998 (cit. on p. 37).

[53] D. Fox, "Kld-sampling: Adaptive particle filters," in *Advances in neural information processing systems*, 2002, pp. 713–720 (cit. on p. 38).

[54] Nickels, Kevin, *Kld sampling for particle filters - using kullback-leibler distance*, version 1.1.0.0, Jul. 25, 2019. [Online]. Available: `https://de.mathworks.com/matlabcentral/fileexchange/44735-kld-sampling-for-particle-filters-using-kullback-leibler-distance` (cit. on p. 38).

[55] P. Nazemzadeh, D. Fontanelli, D. Macii, and L. Palopoli, "Indoor localization of mobile robots through qr code detection and dead reckoning data fusion," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2588–2599, 2017 (cit. on p. 53).

[56] S. Garrido-Jurado, R. Muñoz-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, 2014 (cit. on p. 54).