# Feature Based SLAM using High-Noise Low-Cost Automotive Sensors

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Frank Schuster
aus Aurich

Tübingen
2019

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

| | |
|---|---|
| Tag der mündlichen Qualifikation: | 13. Januar 2020 |
| Dekan: | Prof. Dr. Wolfgang Rosenstiel |
| 1. Berichterstatter: | Prof. Dr. Cristóbal Curio |
| 2. Berichterstatter: | Prof. Dr. Andreas Schilling |

# Contents

**Abstract** **1**

**Zusammenfassung** **3**

**1 Introduction** **5**
1.1 Objectives and Contributions . . . . . 6
1.2 Publications . . . . . 9
1.3 Thesis Outline . . . . . 10

**2 Fundamentals** **11**
2.1 The Driving State . . . . . 11
2.1.1 Ackermann Steering Geometry . . . . . 12
2.2 The RADAR Sensor . . . . . 14
2.2.1 RADAR Architectures . . . . . 14
2.2.2 Physical Principle of RADAR Sensors . . . . . 17
2.2.3 Properties of RADAR Measurements . . . . . 20
2.3 The Simultaneous Localization and Mapping (SLAM) Problem . . . . 25
2.3.1 The Bayes Filter . . . . . 27
2.3.2 Extended Kalman Filter SLAM . . . . . 28
2.3.3 Particle Filters . . . . . 31
2.3.4 GraphSLAM . . . . . 33
2.3.5 Other SLAM Algorithms . . . . . 42
2.4 Conclusion . . . . . 45

**3 Experiments** **47**
3.1 Sensor Setup . . . . . 47
3.2 Coordinate System and Calibration . . . . . 50
3.3 Dataset . . . . . 51

**4 Radar Based SLAM** **55**
4.1 Radar Grid Mapping . . . 55
4.1.1 The Sensor Model . . . 55
4.1.2 Layers of the RADAR Grid . . . 56
4.1.3 Joint Occupancy Grid Maps . . . 59
4.2 Robust RADAR Cluster Map Representation . . . 62
4.2.1 Map Representation . . . 63
4.2.2 Particle Filter . . . 69
4.2.3 Evaluation of ClusterSLAM . . . 72
4.2.4 Conclusion . . . 77
4.3 Graph Based RADAR SLAM . . . 77
4.3.1 RADAR Based Landmarks . . . 79
4.3.2 Graph Construction . . . 83
4.3.3 Experimental Results . . . 88
4.3.4 Conclusion . . . 92
4.4 Radar Map Updates using Joint Graph Optimization . . . 93
4.4.1 Joint Graph Optimization . . . 93
4.4.2 Map Refinement . . . 98
4.4.3 Experimental Results . . . 100
4.4.4 Conclusion . . . 105

**5 Sensor Fusion in GraphSLAM** **107**
5.1 LiDAR Landmark Integration . . . 109
5.1.1 Generation of LiDAR landmarks . . . 109
5.1.2 GraphSLAM with LiDAR . . . 113
5.1.3 Evaluation . . . 114
5.2 Stereo Camera Integration Using Stixel . . . 120
5.2.1 The Stixel Representation . . . 120
5.2.2 GraphSLAM with Stixel . . . 122
5.3 Conclusion . . . 126

**6 Conclusion and Future Work** **127**
6.1 Conclusion . . . 127
6.2 Future Work . . . 131

**List of Figures** **136**

**List of Tables** **142**

**List of Algorithms** **142**

**Bibliography** **145**

# Abstract

Ever since the 1980s, researchers in computer science and robotics have been working on making autonomous cars. Due to recent breakthroughs in research and development, such as the Bertha Benz Project [ZBS+14], the goal of fully autonomous vehicles seems closer than ever before. Yet a lot of questions remain unanswered. Especially now that the automotive industry moves towards autonomous systems in series production vehicles, the task of precise localization has to be solved with automotive grade sensors and keep memory and processing consumption at a minimum.

This thesis investigates the Simultaneous Localization and Mapping (SLAM) problem for autonomous driving scenarios on a parking lot using low cost automotive sensors. The main focus is herby devoted to the RAdio Detection And Ranging (RADAR) sensor, which has not been widely analyzed in an autonomous driving scenario so far, even though they are abundant in the automotive industry for applications such as Adaptive Cruise Control (ACC). Due to the high noise floor, the radar sensor has widely been disregarded in the Intelligent Transportation Systems and Robotics communities with regards to SLAM applications. However in this thesis, it is shown that the RADAR sensor proves to be an affordable, robust and precise sensor, when modeling its physical properties correctly.

In this regard, a GraphSLAM based framework is introduced, which extracts features from the RADAR sensor and generates an optimized map of the surroundings using the RADAR sensor alone. This framework is used to enable crowd based localization, which is not limited to the RADAR sensor alone. By integrating an automotive Light Detection and Ranging (LiDAR) and stereo camera sensor, a robust and precise localization system can be built that that is suitable for autonomous driving even in complex parking lot scenarios. It it is thereby shown that the RADAR sensor is strongly contributing to obtaining good results in a sensor fusion setup.

These results were obtained on an extensive dataset on a parking lot, which has been recorded over the course of several months. It contains different weather conditions, different configurations of parked cars and a multitude of different trajectories to validate the approaches described in this thesis and to come to the conclusion that the RADAR sensor is a reliable sensor in series autonomous driving systems, both in a multi sensor framework and as a single component for localization.

# Zusammenfassung

Schon seit den 1980er Jahren ist das autonome Fahren ein entscheidendes Forschungsgebiet in der Informatik und Robotik. Dank jüngster Erkenntnisse, die etwa durch die Bertha Benz Fahrt [ZBS+14] gewonnen wurden, rückt das Ziel des vollautonomen Fahrzeugs wieder in greifbare Nähe. Dennoch bleiben viele Fragen bisher unbeantwortet. Besonders nun, wo die Automobilindustrie in Richtung autonomer Fahrbetriebe für den Serieneinsatz drängt, bleibt weiterhin die Aufgabe der Fahrzeuglokalisierung mit serientauglichen Sensoren bisher unbeantwortet. Außerdem müssen in diesem Fall Speicher- und Rechenressourcen auf einem Minimum gehalten werden.

In dieser Arbeit wir das Simultaneous Localization and Mapping (SLAM) Problem für autonome Fahrszenarien auf einem Parkplatz näher analysiert. Dabei kommen vor allem Radio Detection and Ranging (RADAR) Sensoren zum Einsatz, die bisher zur Lokalisierung noch nicht im Detail analysiert wurden, obwohl sie zahlreich im Rahmen von Fahrerassistenzsystemen wie Abstandregeltempomaten zum Einsatz kommen. Da RADAR Sensoren ein hohes Grundrauschen haben, wurden diese bisher nur spärlich bei den hochpräzisen Verfahren, wie sie für das autonome Fahren nötig sind, eingesetzt, weder in der Robotik noch in der Intelligent Transportation Systems Community. In dieser Arbeit wird gezeigt, dass der RADAR Sensor ein kosteneffizienter, robuster und präziser Sensor zur Fahrzeuglokalisierung sein kann, wenn seine physikalischen Eigenschaften korrekt modelliert sind.

In diesem Zusammenhang wurde ein GraphSLAM basiertes Rahmenwerk entwickelt, das markante Punkte nur auf Basis von RADAR Daten extrahiert und so eine optimierte Karte der Umgebung des RADAR Sensors generiert. Dieses wird weiterhin benutzt, um eine Crowd basierte Lokalisierung umzusetzen, welche nicht nur auf den RADAR Sensor limitiert ist. Indem weitere Sensoren, wie z.B. ein Laserscanner und eine Stereokamera, hinzugenommen wurden, konnte ein robustes Lokalisierungssystem entwickelt werden, welches das autonome Fahren auf dem Parkplatz ermöglicht und den hohen Genauigkeitsanforderungen gerecht wird. Es wird gezeigt, dass der RADAR Sensor eine entscheidende Rolle bei der durchgführten Sensorfusion spielt und somit die hohe Gesamtperformance erst möglich macht.

Die Ergebnisse dieser Arbeit werden auf einem Datensatz auf einem öffentlichen Parkplatz evaluiert, der über mehrere Monate hinweg verschiedenste Szenarien abdeckt. Neben verschiedenen Wetterbedingungen und unterschiedlicher Parkplatzauslastung, wurden auch eine Reihe verschiedener Parkmanöver erfasst. Die Auswertung dieser Verfahren lässt die Aussage zu, dass der RADAR Sensor sich sowohl allein als auch in einem größeren Framework für das autonome Fahren eignet.

# 1 Introduction

The quest of autonomous cars in public roads is almost as old as the automobile itself. In the 1920s, Houdina Radio Control, a company specializing in radio equipment, built a radio controlled car that could roam the streets of downtown New York City without a driver behind the steering wheel. The driver was instead controlling the vehicle via a remote transmitter from a following car. The vision of driverless and even self driving cars was born. Roughly a decade later, at the 1939 World's Fair (later called Expo) in New York, the first autonomous car guided by electric circuits in the ground was presented. But the completely unattended, fully autonomous vehicle was still science fiction, though very present in the minds of car manufacturers and research facilities.

It was not until the mid 2000s that the fully autonomous systems gained great attention once again. In 2004, the first DARPA Grand Challenge offered a $1M prize for the research company who could create an autonomous vehicle capable of finishing a 150 mile route through the Mojave Desert. None of the participating research teams were able to complete the course though. At the second DARPA Grand Challenge in 2005, the participants had to create a fully autonomous car that would reach a series of predefined GPS checkpoints, also in a desert environment. Five cars succeeded in completing the course.

The DARPA Grand Challenge and at a later time the DARPA Urban Challenge (2007) showed that fully autonomous driving had become almost real. Even though typically huge contraptions on the roof of the car were required for environment perception, the feasibility of autonomous driving had been proven. This impulse started strong engagements both in academic research and industry towards series production of autonomous cars.

In 2013, Daimler AG showed off how the technology could be brought to a close-to-series production car with the S500 Intelligent Drive. This research vehicle was equipped with a stereo camera and RADAR sensors for environment perception that did not require large mounts on top of the car. The necessary sensors were seamlessly integrated into the body of the car. With this sensor setup, the car drove the 105 km Bertha Benz memorial route including urban, overland and highway parts without the need of highly accurate GNSS/INS systems.

Since then, automotive manufacturers have been investing more and more into an autonomous system that is robust enough to be bought and operated by customers without the need of specially trained safety drivers or other means of human intervention, without losing their aesthetic appeal due to disturbing sensor placement.

In the work presented here, the RADAR sensor as a well-established automotive sensor will be examined to solve one of the most challenging problems for autonomous driving: the localization problem. In order to build reasonably moving vehicles that can find their way in known or unknown areas, the localization problem is essential, since it both enables path planning, as well as locating the vehicle's position and orientation with respect to potential points of interest, such as parking positions or the driver's destination.

To complete these tasks, localization with respect to a map is required. But since the robot is moving in a potentially unknown environment, a high dimensional state estimation needs to be performed. This includes not only estimating the current vehicle position, but all the previous positions in the trajectory as well. This problem has to be solved especially in GPS denied areas, such as parking garages or urban traffic, where the line of sight to satellites is typically obstructed by tall buildings and roofs. Thus to determine the vehicle's pose, a map has to be generated.

A localization system capable of meeting the stated requirements with deployment to customers in mind has to deal with two important factors: On the one hand, maps used for localization and path planning must be up-to-date in order to achieve good localization results. On the other hand, the accuracy of the solution has to remain high enough, both with respect to sensor noise and to limited sensor information due to highly changing environments or architectural constraints inside a vehicle, such as transfer bandwidths of sensor data over a vehicle data bus.

## 1.1 Objectives and Contributions

One of the major areas of research for many manufacturers is the parking use case, i.e. driving autonomously on a public parking lot. Significant progress has been achieved by the V-Charge project [SSF15] for valet parking applications. Valet parking allows vehicles to be parked fully autonomously in designated parking areas such as parking garages equipped with special hardware infrastructure [TSW16] to coordinate the vehicles. This use case was described by the Waymo CEO John Krafcik as one of the most challenging ones for autonomous driving systems [Mat19].

In contrast to valet parking project, the main focus was placed on parking lots located at office buildings (Figure 1.1) and on private property. Specifically, it was not permitted to make use of any sort of active infrastructure or rely on any sort of highly accurate global positioning, such as Differential Global Navigation Satellite System (DGNSS). This way, the system was still capable of operating in parking garages that typically do not have clear sky conditions. The lower image of the figure shows the characteristics of the environment. On the one hand, the scenario is rich in rigid structures very useful for localization. On the other hand, there are numerous areas that are dynamic over time. One can expect different configurations of parked cars each day and the occupancy of the parking lot will vary vastly over


**Figure 1.1:** The parking scenario in front of an office building (upper image, Google Maps) is shown as an areal photography. Parking scenarios as shown schematically in the lower image, are both an important and challenging problem for autonomous driving applications.

the course of the day. At night the parking lot will be almost empty, while during the day, the parking lot will be completely full, obstructing many of the immobile structures from the line of sight of the vehicles sensors.

This thesis focusses on the localization aspects an autonomous driving project set on a public or private parking lot without access to a highly accurate GNSS system. The goal is to generate an accurate map using only automotive sensors and to provide a robust localization that can ensure the safety and availability of the system even in changing environments. Thus this thesis contributes to four main areas of research.

**The Lifelong Mapping** problem is one of the most challenging ones when researching SLAM. Because SLAM is performed in a high dimensional state space, the world is usually assumed to be quasi-static due to computing limitations. In a changing environment, the association of features to a map is much more challenging than in a static environment. Additionally, the probability of existence of an object becomes another variable in the state space. If the lifelong mapping is solved, the localization system becomes more robust and the availability of the system increases, even if the environment changes significantly, as e.g. in a public parking garage.

**The RADAR sensor** has been researched very little in the context of SLAM because of its high noise floor and otherwise adverse physical properties. Even though it measures range and angle similar to a laser scanner, the accuracies and noise characteristics are very different due to the physical behavior of light in the microwave range. Interference can create ghost objects or hide real physical objects, the scattering properties are highly material dependent and the angular resolution is much worse than for laser scanners. On the positive side, due to the microwave lengths of the RADAR waves, it penetrates thin layers of material, making them ideal for invisible placement behind the bumpers of the car. They are also cost efficient and well established in the automotive industry for driver assistance systems such as ACC, but due to the noisy properties of the RADAR signals, solving the SLAM problem proves to be much more challenging.

**Collaborative Localization** is a key technique to solving lifelong mapping effectively and keeping maps up-to-date. Multiple vehicles contribute in a crowd based approach to one unified map of the environment. That introduces additional challenges especially in the data association area. Cars from different manufacturers may have different sensor setups or different types of sensors. To use a crowd based approach to one's advantage, compatibility of these maps needs to be ensured.

**Computing restrictions** limit the available processing power of a localization system. Signal and data processing units in a customer vehicle are typically realized on an Electronic Control Unit (ECU) with very limited computing resources, because they are cooled passively. There are also restrictions regarding the implementation, which has to fulfill certain guidelines as given in the Motor Industry Software Reliability Association (MISRA) standard. The focus of this thesis is mostly put into meeting the computing restrictions necessary for an ECU implementation of the localization. [SWK+16]

## 1.2 Publications

During the course of this research, a number of publications have been made available to the scientific community.

### Conference and Workshop Articles

- [SWK+16] F. Schuster, M. Wörner, C.G. Keller, M. Haueis, and C. Curio. Robust localization based on radar signal clustering. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016

- [SKR+16] F. Schuster, C. G. Keller, M. Rapp, M. Haueis, and C. Curio. Landmark based Radar SLAM Using Graph Optimization. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016

- [WSD+16] M. Wörner, F. Schuster, F. Dolitzscher, C. G. Keller, M. Haueis, and K. Dietmayer. Integrity for autonomous driving: A survey. In *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE, 2016

- [RDH+16b] M. Rapp, K. Dietmayer, M. Hahn, F. Schuster, J. Lombacher, and J. Dickmann. FSCD and BASD: Robust landmark detection and description on radar-based grids. In *2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. IEEE, 2016

- [SZK+17] F. Schuster, W. Zhang, C.G. Keller, M. Haueis, and C. Curio. Joint graph optimization towards crowd based mapping. In *International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017

### Invited Talk

- [SKH16] F. Schuster, C. G. Keller, and M. Haueis. Measuring the World: Designing Robust Vehicle Localization for Autonomous Driving. In *2016 IEEE Intelligent Vehicles Symposium (IV) - Workshops*, 2016

### Patents

- F. Schuster, M. Wörner, C. G. Keller, and M. Haueis. DE102015003666: Verfahren zur Verarbeitung von erfassten Messdaten eines Sensors, 2016

- F. Schuster and C. G. Keller. DE102015011358: Verfahren zum Betrieb eines Fahrzeugs, 2016

- F. Schuster, M. Wörner, C. G. Keller, and M. Haueis. DE102015011467: Verfahren zur Erstellung einer digitalen Karte eines Parkraums, 2016

## 1.3 Thesis Outline

This thesis can be divided into three major parts giving an introduction, explaining the contributions of the research and providing an outlook into further applications.

In Chapter 2 the automotive RADAR sensor and its applicability to SLAM is examined in detail, describing what has to be taken into consideration when developing a robust localization system with this particular sensor. The focus mainly lies on the physical properties of the sensor and their implication on the concept of localization. An overview of the different types of RADAR sensors and their current applications is discussed in depth.

In Chapter 3, the vehicle setup and the experiments, as well as the dataset are described in detail. The relevant information about runtime, computing resources and time management are discussed. Furthermore, an introduction to each specific sensor and a characterization of the ground truth device is given.

After the input signals have been characterized, we will focus on the major contributions of this thesis in Chapter 4, namely the architecture of the software of the GraphSLAM based on automotive sensors and the map generation process. The localization and mapping process is described in detail. This includes a solution to the lifelong mapping problem using crowd based approaches.

Since one major aspect of crowd based approaches is to handle different sensor configurations, in Chapter 5 we examine potential fusion of the RADAR based framework with an automotive grade LiDAR sensor. It can be shown how GraphSLAM can be used to keep multiple sensor layers consistent and how merging the information form multiple sources improves the result by adding robustness to the localization.

To conclude this thesis, in Chapter 6, a summary of the contributions of this thesis is providded, as well as a discussion of potential avenues of further improvements and generalizations.

# 2 Fundamentals

In this chapter the necessary components for precise localization with the RADAR sensor is introduced. This includes a description of the driving state in Section 2.1, which describes the motion of the vehicle, as well as the specific properties of the RADAR sensor in Section 2.2. In Section 2.3 an introduction to the most common SLAM algorithms is given.

## 2.1 The Driving State

The driving state usually refers to the current velocity, as well as the angular velocity of the vehicle. For the SLAM problem, this information is crucial because it usually serves as a control update in any filter based approach to localization. Starting in the early 1980s, cars have been outfitted with sensors to determine the driving state with Honda's Electro Gyro-Cator that used an Inertial Measurement Unit (IMU) containing a helium gas gyroscope and projected the coordinates onto a 6 inch CRT screen as can be seen in Figure 2.1. Even with the introduction of Global Positioning System (GPS), the driving state estimation remained a vital part of navigation algorithms as control updates in Kalman filter based position estimation.

Nowadays the driving state is used for multiple systems in the vehicle, namely the Electronic Stability Program (ESP) and Anti-lock braking system (ABS) system.


**Figure 2.1:** Honda's Electro Cyro-Cator from 1981 was said to be the first navigation system using an Inertial Measurement System to determine the driving state of a vehicle (Source: Honda.com).

These need precise information about angular velocity as well as lateral and longitudinal velocity to control the brakes of individual wheels in order to safely decelerate the car in an emergency situation. The ESP ECU in a vehicle thus has an integrated electronic gyroscope.

### 2.1.1 Ackermann Steering Geometry

Ackermann steering geometry [MSS06] is used to describe the dynamics of a vehicle while it is moving and is used to develop steering systems without wheel slip. The dynamics are especially relevant when determining the relative odometry of the vehicle for autonomous driving applications. The principle is displayed in Figure 2.2. The Ackermann motion model was introduced and patented in 1818, when it was used to construct steered four wheeled systems that follow turning circles of different radii, such that the center of rotation is given by a single point. Note that because each wheel turns around the center of rotation with a different radius, the velocity of the left wheels will be smaller than the right wheels in a left-hand turn and vice-versa. Therefore, in order to ensure smooth steering, a differential has to be introduced to the front axis to enable different rotation rates of the wheels.

Hence, the turning can be reduced to a bicycle model, describing the vehicle dynamics of two wheel steering (blue circles in Figure 2.2) to one single turning circle through the centre of mass of the vehicle.

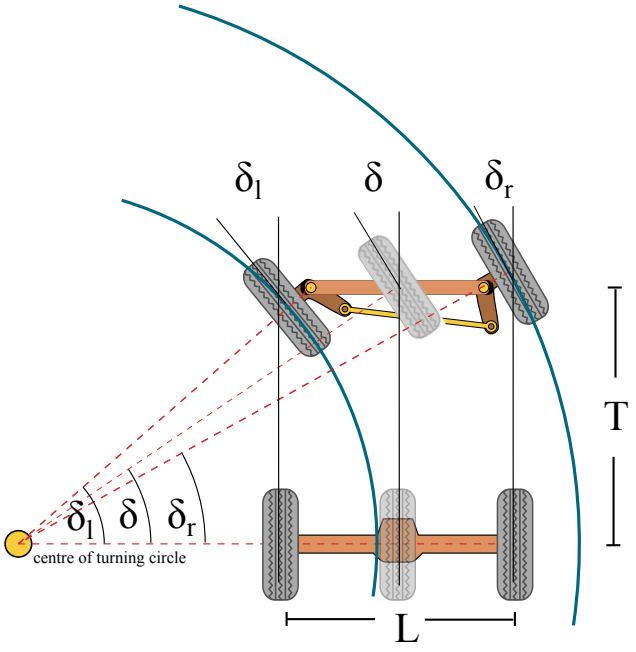



**Figure 2.2:** Ackermann steering describes the steering around a common centre of rotation, such that there is no wheel slip. This is archived by turning both front wheels with a different turning radius.

In the context of autonomous driving, the Ackermann steering model is the foundation to determining the odometry signal, which describes how the vehicle has moved between two points in time. Consider a vehicle with wheel base $L$ and tread $T$. Then, a steering angle $\delta$ of the vehicle is guaranteed to be slipless if the angles of the inner wheel $\delta_l$ and the outer wheel $\delta_r$ differ from $\delta$ such that the turning circles of the outer and inner wheels are both centered around the same point of rotation. It can immediately be seen that

$$\delta_l > \delta > \delta_r \tag{2.1}$$

and upon further inspection of Figure 2.2, we can see that the turning angles of the front wheels and the angles between the wheels and the centre of rotation are in fact equal to the steering angles of the individual wheels. Thus we can infer the following relationship for the steering angle

$$\tan\delta = \frac{T}{r}, \tag{2.2}$$

where $r$ is the curve radius. Furthermore the turning angles of each front wheel can be determined by

$$\delta_r = \arctan\frac{T}{r + L/2}, \tag{2.3}$$

$$\delta_l = \arctan\frac{T}{r - L/2}. \tag{2.4}$$

With these results it can immediately be seen that each wheel travels a different arc length when turning and is thus traveling at a different speed. For the front two wheels, the speed is given by

$$v_r = v_0 + \frac{\omega}{2\pi}(r - L/2), \tag{2.5}$$

$$v_l = v_0 + \frac{\omega}{2\pi}(r + L/2), \tag{2.6}$$

where $\omega = \dot{\delta}$ the angular velocity of the vehicle. In localization this difference in wheel speed needs to be taken into account when calculating the control update from the wheel encoders placed in each individual wheel. This process is usually referred to as the relative odometry of the vehicle and is used in almost all localization algorithms.

This set of equations can be integrated once more over time to determine the traveled distance of each wheel. These result in the desired odometry output. Note though that the double integration introduces quadratic errors into the Ackermann model. Thus it can only be used for short periods of time, because the drift in heading and position will increase quadratically over time. On top of the quadratic error, the position of the vehicle cannot be estimated from odometry alone, since the starting point is arbitrary and does not correspond to any specific position in the map.

## 2.2 The RADAR Sensor

The RADAR sensor has been among the first active sensors that have been deployed in vehicles to implement the adaptive cruise control ACC system. Since then it has proven to be a reliable sensor in parking and longitudinal control systems.

RADAR sensors are categorized as active sensors, meaning that they can detect and track objects by distributing electromagnetic waves into a volume in space. The reflected energy from the objects is being detected and the time between transmission and reception of the signal is measured. The RADAR can thus estimate angle $\theta$, distance $r$, amplitude $A$ and relative velocity $v$ of the detected object.

Several other sensors, such as camera, LiDAR or infrared sensors have been used to perform these tasks[1]. The RADAR sensor on the other hand has proven to be more robust towards weather influences [DWR+15] due to its wavelengths ranging from around 1.2 cm to 3.8 mm. Thus the signal can penetrate layers of matter.

This chapter focuses on the details of the operation of RADAR sensors including the physical principle of RADAR measurements, types of automotive RADARs and their practical benefits and merits.

### 2.2.1 RADAR Architectures

The RADAR sensor concept itself exists since the 1930s and was originally developed to detect enemy airplanes during W.W.II [Sko85]. Ever since, there have been various different RADAR hardware architectures over the years for different use cases. The focus in this thesis lies on the sensors developed in the automotive context. In this case, only the following architectures are of importance and can be divided into two main operation modes: continuous wave and pulsed RADARs. These two modes are discussed in more detail in the following sections.

#### Continuous Wave RADARs

Continuous Wave (CW) RADARs transmit continuous frequency carriers as the RADAR signal. If the carrier is unmodulated, the signal can only detect the velocity of the object via Doppler shifts and can not detect its range. Thus, those types of RADARs are not used for automotive applications and especially for the localization use case, since stationary objects are of the most interest in this case. To be able to measure a position, the RADAR signal is modulated. This can be done in two popular ways:

[1] Thus the first ACC system as introduced by Mitsubishi in 1995 using a LiDAR, but suffering from weather influences and dirt on the sensor.

**Frequency Chirped Architectures** are the most popular in the automotive industry [AJ12]. As schematically shown in Figure 2.3, the RADAR wave consists of linearly ascending and descending linear chirps. The frequency is ramped between a minimum $f_0$ and a maximum $f_1$. The variation of the frequency effectively increases the bandwidth of the RADAR wave and thus narrows the signal in the time domain as shown in the upper part of Figure 2.3. Let $Tx$ be the transmitted spectrum (as displayed in the middle of Figure 2.3), then $Rx$ shows a typical received signal from an object. The returning signal is typically shifted both in frequency ($\Delta f$) and in time ($\Delta t$). The frequency shift originates from the velocity of the object relative to the RADAR and the time shift measures the time of flight and thus the distance of the object from the sensor. Since it is a continuous wave RADAR, the actually detected signal is the result of interference between the transmitted and the returned signal $Tx - Rx$, which is depicted in the lower part of Figure 2.3, where $f_r$ is the frequency component based on the target range and $f_d$ is the frequency increase due to the target's velocity.

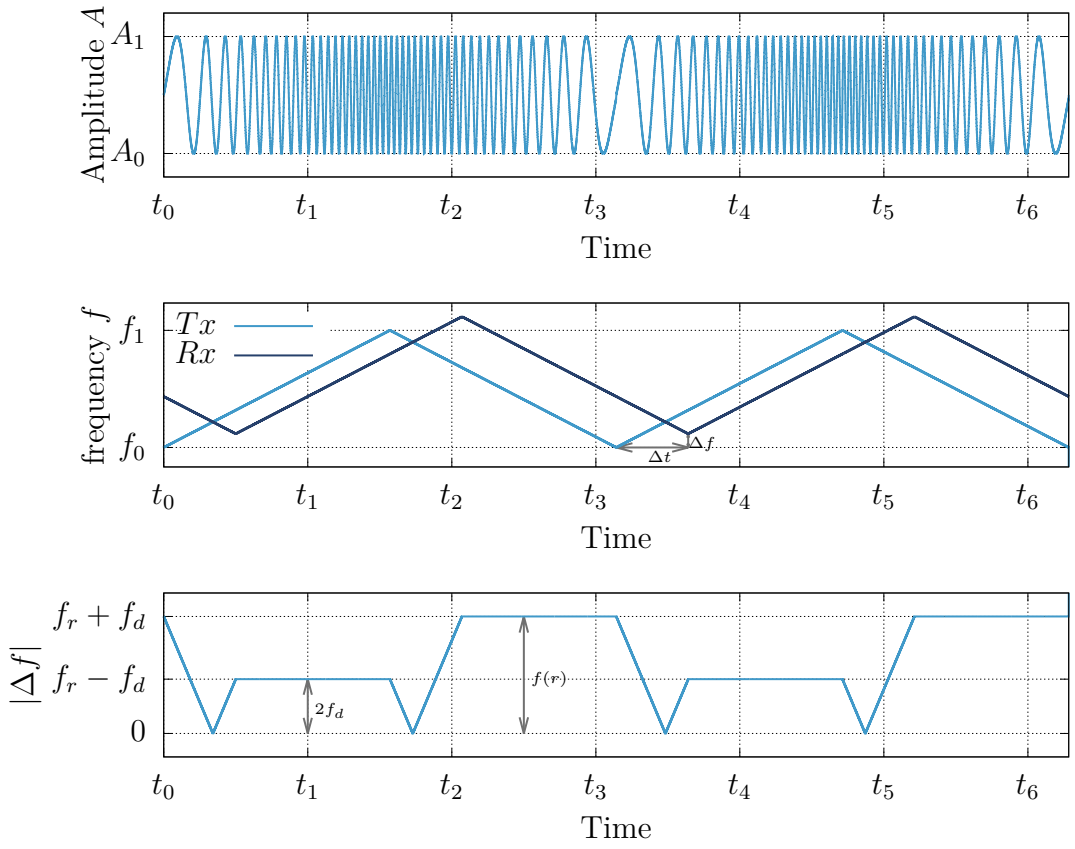



**Figure 2.3:** The emitted RADAR chirp wave form is depicted in the upper diagram, while the frequency domain is shown in the middle graph (transmitted $Tx$ and received $Rx$ signal). The lower graph shows the detected signal in the frequency domain, which forms from interference between $Tx$ and $Rx$.

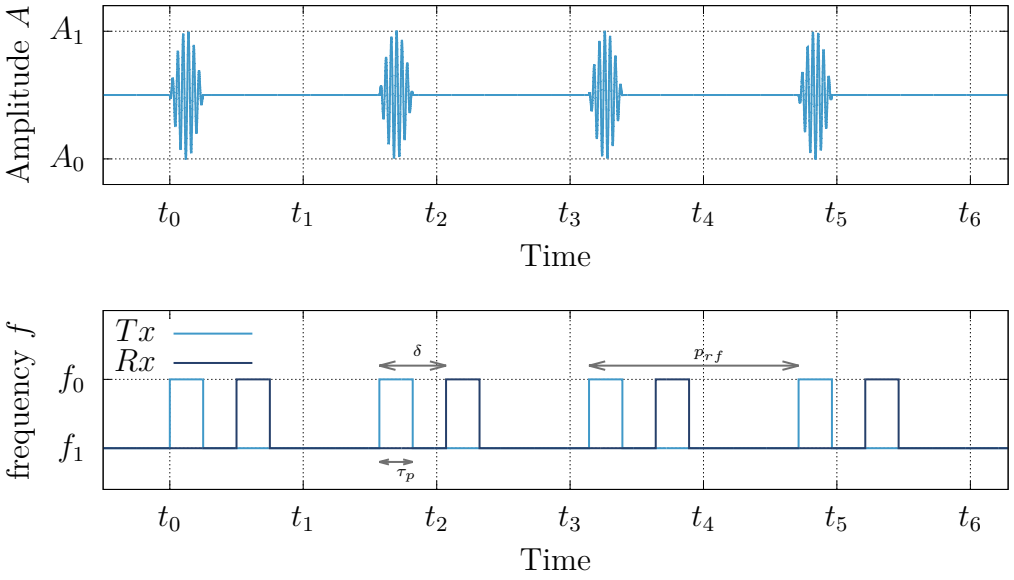



**Figure 2.4:** The pulsed RADAR consists of sequential frequency pulses. The wave form is depicted in the upper diagram, while the frequency domain is shown in the lower graph ($Tx$, $Rx$).

**Pulsed RADAR** systems transmit modulated RADAR pulses in fixed time intervals, as illustrated in Figure 2.4. It can be adjusted by three main parameters, the pulse width $\tau_p$, the carrier frequency $f_0$ and the repetition frequency $f_{pr}$, whereas the latter is the main parameter to determine the maximum range of the signal and also the Doppler measurement resolution.

**Pseudo-Random Noise Coded RADAR** are widely used in communication systems for high data rates and robustness against electromagnetic interference[2] [Sch13]. The pseudo-random code can be generated by feeding a signal through a Linear Feedback Shift Register (LFSR) generating an encoded signal with noise-like properties.

Applying this communication protocol to the RADAR sensor, the main advantage is increased robustness against interference, eliminating one of the major sources of noise in RADAR based object detection. However due to this encoding, repeating signal patterns occur after $2^m - 1$ bits, where $m$ is the length of the LFSR. In a continuous signal stream, increasing the RADAR range thus means exponentially more complex pattern generation circuitry. Thus the typical maximum range of pseudo-random noise coded RADAR systems is limited to 10 m [JH13].

[2]The communication between processing units and the detectors at CERN are implemented using PN codes to handle the harsh radiation near the particle collision point [Sch13].

### 2.2.2 Physical Principle of RADAR Sensors

The RADAR sensor actively transmits electromagnetic waves with wavelengths in the millimeter range. The signal therefore travels at the speed of light, given by $c \approx 2.99 \times 10^8 \,\mathrm{m\,s^{-1}}$. In order to detect a RADAR wave reflected by an object, it must travel twice the distance between the object and the sensor.

**The RADAR Range** is thus given by the following equation

$$\rho = \frac{c}{2}\delta, \tag{2.7}$$

where $\delta$ describes the time delay between transmission and reception of the wave. In order to avoid ambiguities between transmission and reception of signals, the RADAR must wait for a sufficient amount of time $\delta_{\max}$ before it can transmit again. Thus the maximum RADAR range is determined by

$$\rho_{\max} = \frac{c}{2}\delta_{\max} =: \frac{c}{2f_p}. \tag{2.8}$$

In pulsed RADAR systems, $\delta_{\max}$ is typically replaced by the pulse repetition frequency $f_p$.

**The RADAR Resolution** is another important characteristic of a RADAR sensor. It describes how far two objects must be apart from one another to be identified as separate objects by the RADAR. From figure Figure 2.4, it can easily be observed that two pulses must be at least $\frac{\tau_p}{2}$ apart to be visible as two distinct signals. Therefore the RADAR resolution is given by

$$\Delta R = \frac{c}{2}\tau_p = \frac{c}{2B}, \tag{2.9}$$

whereas $B := \frac{1}{\tau_p}$ denotes the bandwidth of the RADAR signal. Thus, in order to produce a signal with high resolution, the signal has to have a large bandwidth or come in a narrow pulse.

#### The RADAR Equation

The simple form of the RADAR equation describes the relationship between the received power $P_R$ from a target, its distance from the receiver $r$ and its cross section $\sigma_t$, as well as the characteristics of the antenna. This equation alongside the characteristics described above are fundamental to design RADAR sensors for specific applications, such as object detection or – in this thesis – SLAM.

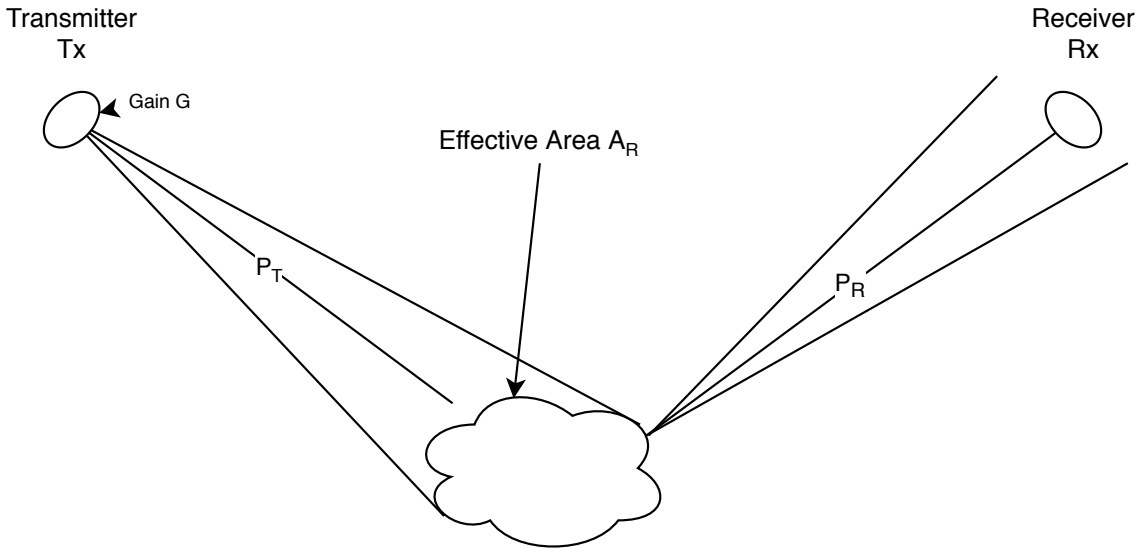



**Figure 2.5:** The RADAR waves are transmitted by the Transmitter $Tx$ with antenna Gain $G$ and power $P_T$ and scattered off an effective area $A_R$ of the target. The reflected signal that is caught by the Receiver $Rx$ then has the remaining power $P_R$.

In order to deduce the RADAR equation, consider Figure 2.5. We assume a theoretically perfect antenna with antenna gain[3] $G$. The power density of the antenna is then

$$P_R = \frac{P_T\, G}{4\pi r^2}, \tag{2.10}$$

where $P_T$ describes the transmitted power and $r$ the distance of the target from the sensor. The received power at the RADAR receiver is obtained by considering the target's RADAR cross section $\Gamma_t$. It represents the characteristics of the target, such as the material and shape. It corresponds to its size as seen by the RADAR. By also considering the receiver's effective antenna area $A_R$ we obtain for the received power

$$P_R = \left(\frac{P_T\, G}{4\pi r^2}\right)\left(\frac{\Gamma_t}{4\pi r^2}\right) A_R. \tag{2.11}$$

From antenna theory we learn that $G = \frac{4\pi A_R}{\lambda^2}$, where $\lambda$ is the wavelength of the RADAR radiation [AJ12]. To keep the size of automotive RADARs small, the RADARs use the same antennas for transmission and reception, such that $G$ is the same for transmitting and receiving signals. Thus we get the classical RADAR equation

$$P_R = \frac{P_T\, G^2\, \lambda^2\, \Gamma_t}{(4\pi)^3\, r^4\, L} \propto \frac{\Gamma_t}{r^4}, \tag{2.12}$$

[3]The antenna gain $G = E \cdot D$ is a unit-less characterizing constant composed of the efficiency $E$ and the directivity $D$ of the antenna.

by adding the RADAR loss factor $L \geq 1$ describing the loss induced by the geometric shapes of the antennas to the equation. By rearranging the equation by distance $r$, we can determine the maximum range of the RADAR with respect to the received power

$$R_{max} \propto \left( \frac{\Gamma_t}{P_{R,min}} \right)^{\frac{1}{4}}. \tag{2.13}$$

Then the maximum detection range is determined by the cross section of the target and the minimum detectable received power $P_{R,min}$, which is determined by the Signal to Noise Ratio (SNR). It can be written as

$$\text{SNR} = \frac{P_{R,min}}{k_B \, T_e \, B_n \, F}, \tag{2.14}$$

where $k_B$ is Boltzmann's constant, $T_e$ the effective noise temperature, $B_n$ the receiver noise bandwidth and $F$ the receiver noise factor [JH13].

Due to low SNR and high power requirements, RADARs usually do not signal a detection of a target from a single pulse, but rather integrate several pulses to form a consolidated detection. Ideally this results in a linear relation, meaning $N$ detections cause an $N$-fold improvement of the SNR. Thus the maximum detection radius is determined by

$$R_{max} = \left( \frac{P_T \, G^2 \, \lambda^2 \, N \, \Gamma_t}{(4\pi r^3) \, k_B \, T_e \, B_n \, F \, \text{SNR}_{\text{min}}} \right)^{\frac{1}{4}} \propto \left( \frac{N \, P_T \, \Gamma_t}{\text{SNR}_{\text{min}}} \right)^{\frac{1}{4}}. \tag{2.15}$$

With this equation, RADAR sensors can be parameterized by determining the SNR through measuring false alarm rates of individual RADAR sensors and infer the other parameters from these results.

In the automotive industry, there are mainly two different forms of RADARs: the Long Range Radar (LRR) is designed to have a high $R_{max}$, with a high directivity of the antenna and thus a narrow Field of View (FOV). It is designed to detect the traffic far ahead of the vehicle. These sensors can be utilized for ACC [DKB+12].

The Short Range Radar (SRR) on the other hand has a wide FOV and thus low directivity. It is used to detect passing cars and also to find parking lots. In this thesis, the SRR is utilized to detect the surrounding structure and serve as the measurement update for the SLAM problem.

#### Doppler Frequency

Because the RADAR wavelengths are in the millimeter range, the Doppler effect becomes significant when targets have relative velocity to the receiver. This results in a frequency shift of the received wave to the transmitted one, which is called the Doppler shift. It is given by

$$\Delta f_d = \pm \frac{2\Delta\vec{v}}{\lambda} = \pm \frac{2\Delta\vec{v} f_0}{c}, \tag{2.16}$$

where $c = \lambda f_0$ is the light speed. For an approaching target, the frequency increases, while for a receding target, the frequency decreases. The velocity $\Delta v$ can be broken down into two angular components for azimuth and elevation and a radial component $\Delta\vec{v} = (v_r, v_\theta, v_\psi)$. The Doppler shift refers only to the radial component of the velocity vector, thus the Doppler shift is determined by

$$\Delta f_d = \pm \frac{2\Delta v f_0}{c} \cos\theta \cos\psi, \tag{2.17}$$

where $\Delta v = |\Delta\vec{v}|$ and $\theta$ and $\psi$ are the current azimuth and elevation, respectively. Additionally to the shift in frequency, the antenna gain $G$ increases quadratically for smaller wavelengths, such that the SNR and the maximum detection radius increase with Doppler shift.

In practice, the Doppler shift yields two main benefits that are unique to the RADAR sensor. On the one hand, by observing the received frequency, the radial component of the target's speed can be determined. This can be used to improve tracking of targets and help in situation analysis for autonomous driving to infer what other participants in traffic are doing. On the other hand, the angular resolution of the RADAR sensor can be improved as soon as the sensor is moving[4]. In practice that means that the surrounding structure becomes sharper as soon as the vehicle starts moving. In the following section Section 2.2.3, this principle, among others, will be discussed in more detail.

### 2.2.3 Properties of RADAR Measurements

In this section the RADAR detections and the occurring artifacts due to the previously discussed physical properties of RADAR will be studied. While RADAR sensors have the advantage of being weather independent, they usually provide less accurate information about the environment than LiDAR or camera data. Out of the most common automotive sensors, namely RADAR, (stereo) camera and LiDAR the RADAR sensor is the least expensive, but the most noisy sensor. This provides a series of challenges considering their use in SLAM algorithms, which are caused by the physical properties described in the previous section.

[4]It must have a relative velocity to the surrounding environment.

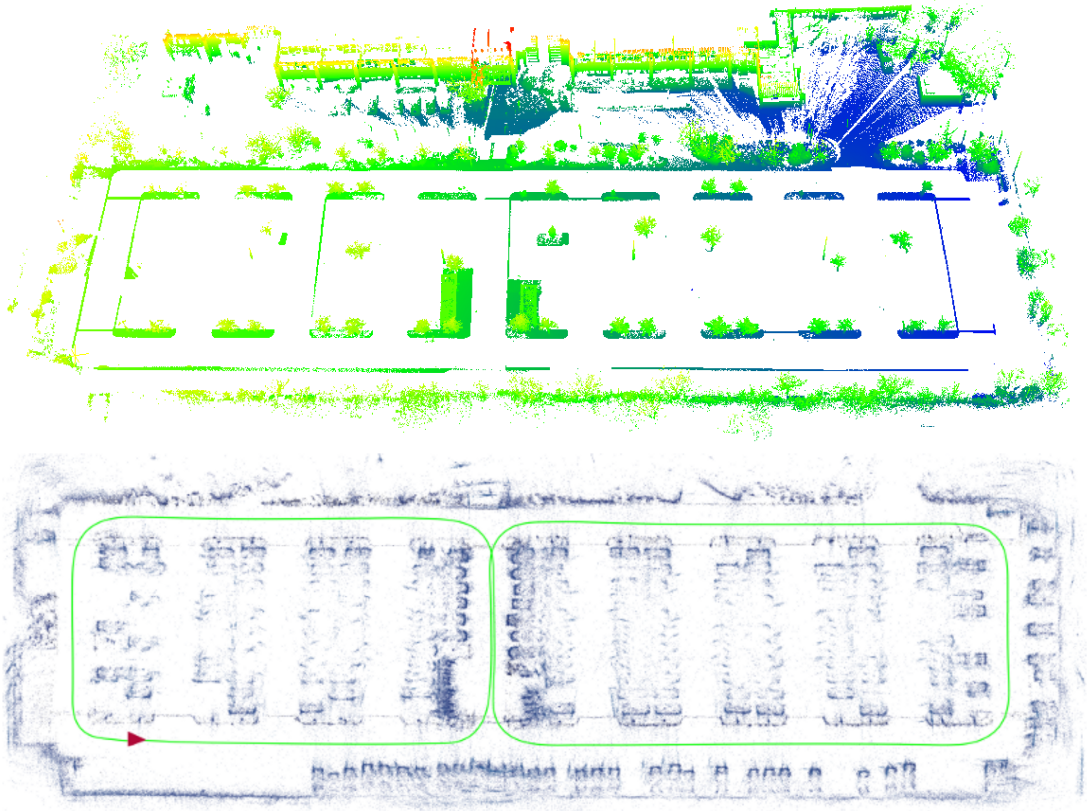


**Figure 2.6:** Comparison of a RADAR point cloud (lower) with a high precision sensor (LiDAR) point cloud (upper) on parking lot data in front of the parking lot shown in Figure 1.1.

To illustrate those changes, consider Figure 2.6. It shows RADAR targets obtained by a car equipped with four RADAR sensors, one in each vehicle corner. Parking spaces in the lot are arranged in nine rows, each accessible from both sides. There is an additional line of parking spaces at the far left, far right and bottom side of the parking lot. The parking lot is surrounded by metal fences and dense hedges. Since vehicles commonly have a metal surface, parked vehicles can be identified by a U-shape of high amplitude and target density, most prominently depicted in the center of the Figure where the trajectory crosses itself [AJ12]. The effects that lead to the noisy sensor data can be categorized into five different classes: speckle, clutter, limited angular resolution, limited resolution at low velocities and multi-path effects. All of these sources of error can be observed in the lower image of Figure 2.6.

**Speckle** are the result of an optical phenomenon caused by coherent light interfering with one another, called self-interference. Both constructive and destructive interference can take place, causing both the disappearance of actual objects as well as the emergence of ghost objects that do not correspond to any physical object. These points can show up anywhere in open space and add a constant noise floor

to the RADAR scan. It occurs especially frequently near objects with reflective surfaces or large scattering cross sections. On the parking lot, parked cars cause the most speckle. In Figure 2.7 speckle appear as numerous points appearing in empty space between parked vehicles.

Eliminating speckle has been a major research area in RADAR sensor development for many years and filtering is the most successful approach in reducing speckle. Since in later sections, the points are matched to a map for pose estimation, speckle cause problems only in the statistically unlikely case of ghost points are erroneously matched to points in the map. Random points in open space that do not correspond to anything are usually simply disregarded.

**Clutter** is generally defined as targets measured from undesired physical objects that were not removed by filtering the input data. These points can be caused by multiple sources on the parking lot such as pedestrians, heavy rain or reflections from unwanted ground points. The points cause problems for localization algorithms because they correspond to non-robust objects that are difficult or even impossible to recognize repeatedly. While in LiDAR and camera systems ground points can typically be detected very well (Figure 2.6), this becomes more difficult in RADAR data, because it does not provide any height resolution. The only filtering method is by reflectivity, which separates i.e. cars with high reflectivity very well from trees or bushes with low reflectivity. Undesired ground points with high reflectivity cannot be filtered by this method though. In Figure 2.8 the rain gutters in the ground are not filtered from the input data, because they have a high reflectivity compared to the surroundings.

Generally speaking, clutter is defined as unwanted RADAR signals exceeding the detection threshold. Since clutter is mostly random, it is one of the most important sources of error for pose estimation using SLAM as discussed in a detailed error

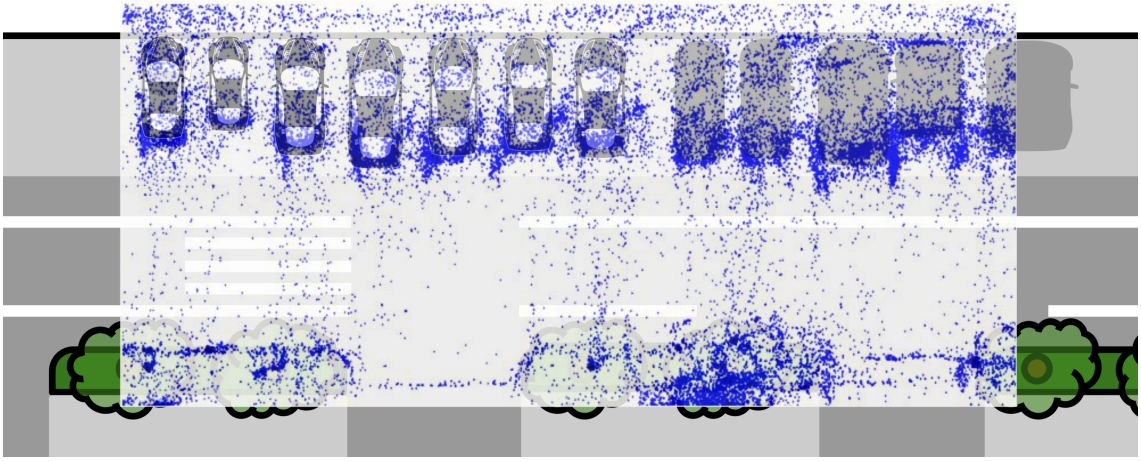


**Figure 2.7:** Increased number of targets on the road as a result of speckle.

analysis in [AJ12] for range sensors. However when clutter is sparse, there are multiple strategies to detect and remove clutter, such as clustering [SWK+16] and more advanced outlier rejection algorithms. Most of the following sections will cover how to cope with clutter in SLAM approaches in particular.

**Limited Angular Resolution** Typical angular resolutions of automotive sensors are around 1°, yielding to washed out signals in the distance. The effects on the parking lot can be seen in Figure 2.7. While the row of parked cars on the left has sharply resolved U shapes corresponding to the bumpers of parked cars, the parked cars on the right cannot be discriminated from each other.

These resolution effects have high impact on SLAM performance, because objects that are far away are most helpful when triangulating positions. The angular resolution limitation is a large source of mismatches though and will thus yield wrong position estimates quite frequently. This effect can be handled by data aggregation and weighting measurements from farther away accordingly. For a detailed explanation see Section 4.1.

**Resolution at Low Velocities** As described before, the effective angular resolution of the RADAR can be increased by measuring the relative velocity of objects to the vehicle and separating objects with different velocities to yield better angular separation. This yields a better separation of the targets if the vehicle is traveling above a certain threshold velocity of around 5 km/h. However, if the vehicle becomes too slow, the resolution of the RADAR sensor decreases and at some point it is not able to sufficiently separate different objects any longer. This effect is illustrated in Figure 2.9. The targets have been recorded when the vehicle was standing still. This can be seen from 2.16, as the frequency of the reflected signal depends on the

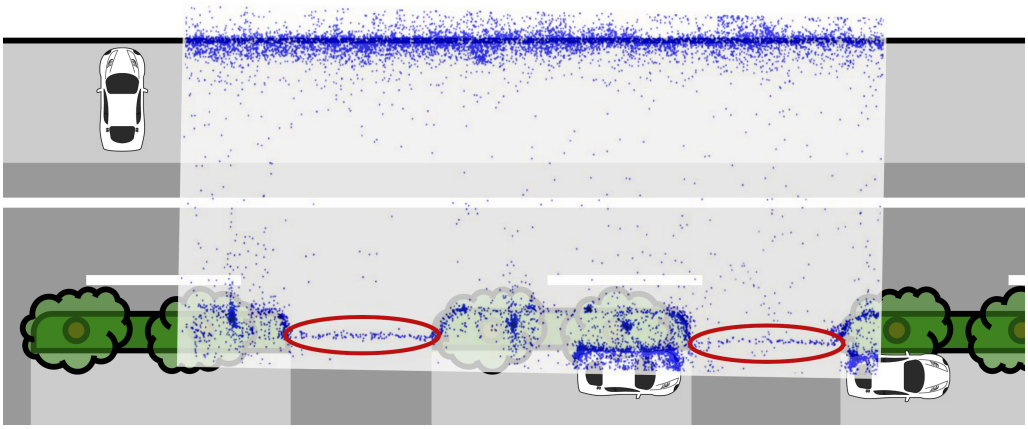


**Figure 2.8:** Targets on the road and rain gutters (marked red) as illustration of clutter. They have a high reflectivity and are therefore not filtered.

velocity of the target. Then the angular separation of 2.9 can be studied to indicate dependency of the resolution of the radar with the bandwith $B$.

There are ring shaped artifacts centered around the vehicle's position. Even though the individual points correspond to physical objects, assigning them to the map becomes increasingly difficult with the reduced angular resolution. This causes blurring effects along angular direction in polar coordinates with stronger effect, the further the detection is from the sensor. Thus we obtain a constant angular spread $\Delta\phi$ and an Euclidean spread $\Delta x$, which depends on the distance from the sensor. This effect can be observed by the keen eye in Figure 2.9.

While these effects can easily be removed by disregarding all RADAR points moving below a threshold velocity of 5 km/h, it can yield to a significant amount of data being unusable depending on the scenario at hand. In the parking lot this is especially problematic. Thus the points in the far field are typically rated more reliable in SLAM applications to reduce wrong matches due to resolution artifacts.

**Multi-Path Effects** Highly reflective objects can lead to multi-path effects manifesting in multiple positions reported for the same physical objects. The vehicle's tail lights are usually also a very good reflector of RADAR waves. In Figure 2.7 this effect can be observed. The corners of the parked vehicles reflect RADAR waves towards the ground resulting in ghost objects along the vehicle's edges on the floor.

These points can cause mismatches for certain highly reflective objects, but the most common example of parked cars plays a minor role, as parked cars are undesirable localization objects to begin with. This means that SLAM algorithms for park-

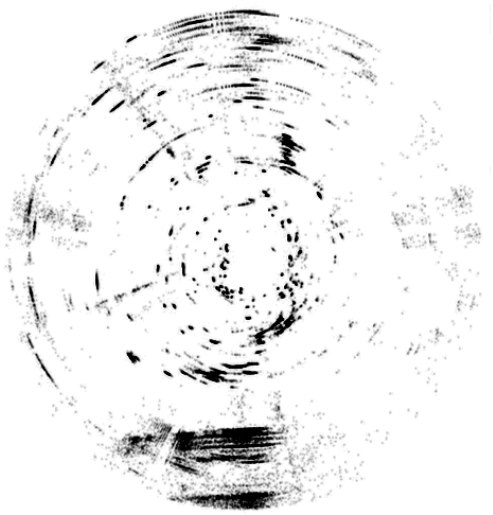


**Figure 2.9:** Circular artifacts can result from insufficient vehicle velocities. The vehicle is located at the center of the figure and has 360 degree RADAR coverage. The circles originate from low angular resolution while the vehicle is standing still.

ing lot have to have measures to prevent false positive matching due to changing environments anyway and can take care of multiple reflections in the same manner.

## 2.3 The Simultaneous Localization and Mapping (SLAM) Problem

The problem of pose estimation of a robot has been a central challenge since the beginning of automation. It provides the connecting tissue between the robot's environment perception and its planning information. Depending on the estimated pose in the map (containing planning and localization information consistently), the corresponding planning information is accessed to perform the required action of the robot. A typical autonomous vehicle architecture is provided in Figure 2.10. The architecture implies also that the localization problem does not answer the question of where the vehicle is in the world, like GNSS based use cases suggest, but rather the question of where the vehicle is located relative to its prerecorded planning information – the map. This question can be answered by GNSS based solutions though by having a sufficiently accurate georeferenced map, which is difficult to obtain in the parking use case discussed in this thesis. Thus for the further investigation of this thesis, GNSS based systems are only used as a reference system and not considered as an integral part of the localization algorithm.

In this section, the SLAM problem is described theoretically and adopted to the case of RADAR inputs. Specifically the GraphSLAM approach is described, since it can be applied well to the RADAR sensor.

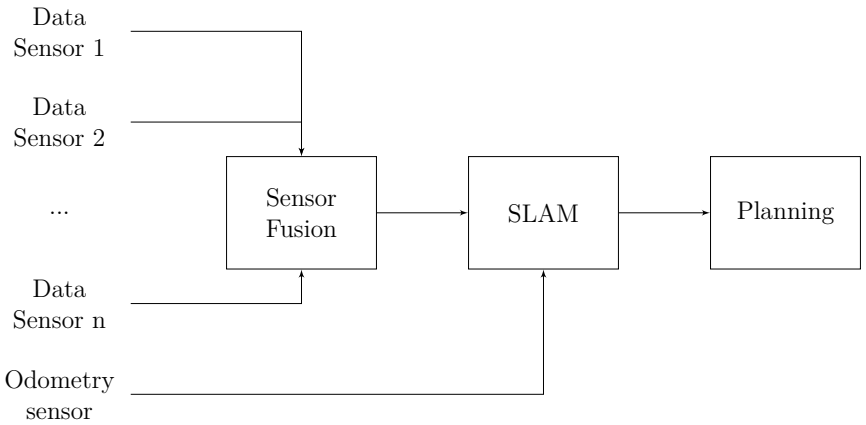
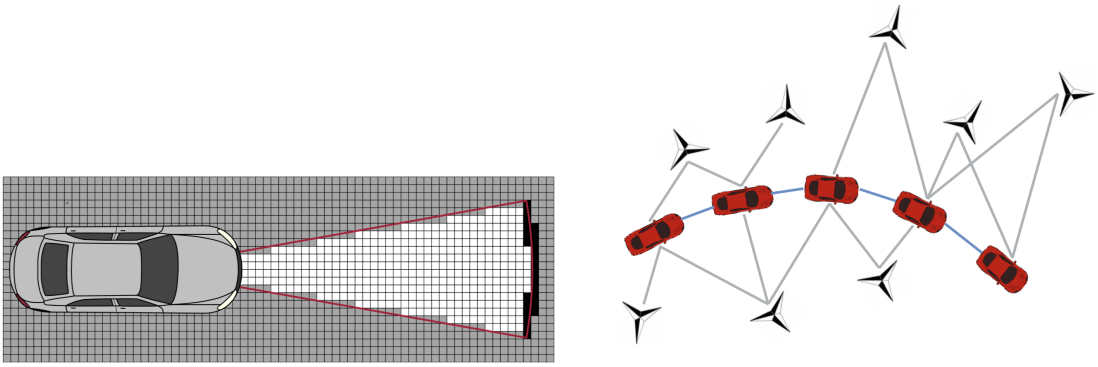



**Figure 2.10:** Typical simplified architecture of a multi sensor autonomous driving setup. The sensor data is usually aggregated in a sensor fusion module. The fused sensor data is passed to the localization module, which provides the correct planning information used for path planning.



(a) Grid-based approaches discretize the state space into static cells that represent the occupancy probability by greyscale. Thus a map of the surroundings can be generated.

(b) Feature-based approaches do not discretize the state space, but mark sparse recognizable objects in space that can be identified by sensors.

**Figure 2.11:** Grid and landmark based approaches are most common among SLAM algorithms [LCW+12].

There has been a lot of research on the SLAM problem researched over the past years [LCW+12], which show two types of SLAM problems. One is the online SLAM problem. It estimates only the posterior distribution of the vehicle pose given the most current pose $x_t$ and map $m_t$. This formulation estimates only current values of all relevant variables and discards past information. The other formulation is the full SLAM problem. It estimates the posterior over the entire trajectory $x_{1:t}$ and map information $m_{1:t}$ of the vehicle and accumulates all data that is necessary for that purpose.

There have been multiple approaches solving both the online and full SLAM problem, resulting in a large variety of SLAM algorithms for different settings and unknown environments. An overview is outlined in [LCW+12]. The algorithms can be divided into feature-based approaches, such as [TM05] and grid-based ones, as presented in [CCR07]. Feature-based approaches store a highly reduced amount of information about the environment, which result in low memory consumption, at the cost of robustness due to association problems [ZCS+08]. These techniques are common for camera sensors [ZLS+14] and have recently been applied to the RADAR sensor by Rapp et. al [RDH+16b]. Grid-based algorithms on the other hand have proven to be effective when combined with range sensors, such as laser scanners and RADARs [EP03]. Grid-based and feature-based approaches are displayed in Figure 2.11 to showcase the difference between the approaches.

### 2.3.1 The Bayes Filter

The basis of most localization algorithms is Bayesian mathematics [TBF05a], as the localization problem is typically perceived as a probabilistic one. Let $X$ be a state variable and $x$ be a specific state. Then we are interested in determining the probability that $X$ is in the particular state $x$ called $p(X = x)$. In other words, we are interested in determining the specific pose of the vehicle, given all possible poses in the map.

In Bayesian terms assume that we can determine a sufficiently accurate state estimation by processing measurements $z_t$ and controls $u_t$. The resulting probability $bel(x_t|u_{1:t}, z_{1:t})$ is called *belief*[5] and reflects the probability of the system being in one possible realization $x_t$ of all possible states $X_t$, given the measurement $z_t$ and the control $u_t$. Furthermore we assume that the state $x_t$ has been reached by a series of controls $u_{1:t}$ and measurements $z_{1:t}$. The Bayes filter assumes furthermore that each state $x_t$ is *complete*, meaning that all the information about $x_t$ is known at time $t$ and that no information about future states $X_{t+1}$ can be inferred by $u_t$ or $z_t$. The so-called *Markov* assumption thus states that each sensor measurement is independent and does not contain information about future measurements.

In reality, the Markov assumption is usually violated when working with complex measurement and motion models. Motion cannot be changed arbitrarily from one timestamp $t$ to another $t + 1$ and the FOVs of sensors overlap to produce similar measurements for each timestamp. Also external conditions, such as other dynamic objects cause occlusion, slow traffic causes state space updates with the same or similar information due to the update rate of the sensors. This valuable information about constraints between different positions is disregarded by the Markov model[6]. Adverse effects of the Markov assumption being violated have to be considered when optimizing localization algorithms for robustness.

By accepting the Markov assumption we can break down determination of the belief $bel(x_t)$ into two steps, one for the control and one for the measurements:

$$\overline{bel}(x_t) = \int_{x_t} p(x_t|u_t, x_{1:t-1}) bel(x_{t-1}) \, \mathrm{d}x_t, \tag{2.18}$$

$$bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t). \tag{2.19}$$

The Bayesian filter applies a *control update* Equation 2.18 and a *measurement update* Equation 2.19 to the prior belief $bel(x_{t-1})$ to obtain the posterior distribution $bel(x_t)$ recursively. Here, $p(x_t|u_t, x_{1:t-1})$ is the probability of obtaining state $x_t$ by applying control $u_t$ to the prior state, $p(z_t|x_t)$ is the probability of obtaining the measurement $z_t$ when in state $x_t$ and $\eta$ serves as a normalization factor. Note that for discrete state spaces, such as grid-based filtering, the integral in Equation 2.18 is replaced by a sum for the discrete state variables.

[5] The belief is also known as the likelihood or a posteriori distribution of a state variable $X_t$.

[6] The dependency of states is better modeled by graph based methods described in Section 2.3.4.

Generally $\overline{bel}(x_t)$ can be seen as an intermediate distribution that shifts the probability of all poses given a control $u_t$ and Equation 2.19 is used to validate that pose by comparing measurements to the inferred positions. The algorithm always applies the control update first and the measurement update afterwards[7]. A practical example of this is shown in Figure 2.12. The most likely pose is thus determined by $\max bel(x_t)$, that is, if it can be determined unambiguously. In Figure 2.12 (b) we can see that given the contents of the measurement $z_t$ and the geometry of the map, information can cause the distribution to be inconclusive. Additionally we assume complete certainty that the measurement indeed exists at least somewhere in the state space. This may be a sufficient approximation for sensors with low false alarm rates, but according to [AJ12] it is inappropriate when applied to RADAR sensors. As shown in Section 2.2.3, there are many sources of error, which can lead to numerous targets not corresponding to physical objects.

### 2.3.2 Extended Kalman Filter SLAM

One of the first solutions to the SLAM problem is based on an Extended Kalman Filter (EKF), which is used for nonlinear state estimation of unimodal beliefs, modeled by Gaussian noise. The discussed reasoning of this section is described in more detail in [TBF05b] and will only be introduced shortly in this section.

We assume that the next state probability is mainly influenced by non-linear functions $g$ and $h$, respectively

$$x_t = g(u_t, x_{t-1}) + \epsilon_t \tag{2.20}$$
$$z_t = h(x_t) + \delta_t, \tag{2.21}$$

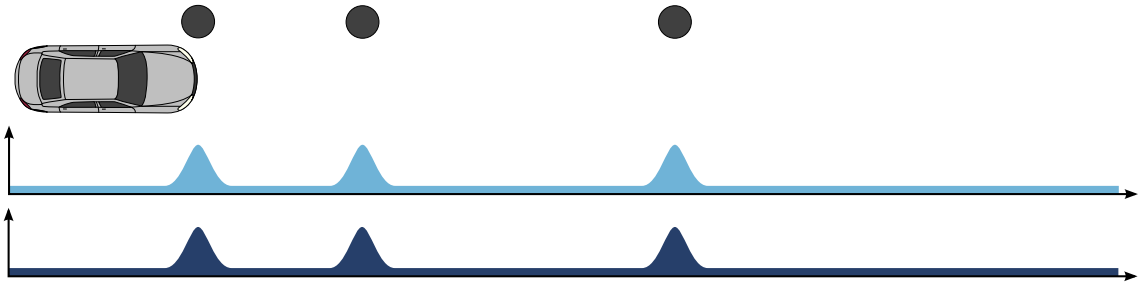
where $g(\cdot)$ is the motion model and $h(\cdot)$ is the measurement model. $x_t$ and $z_t$ are furthermore added with a constant error $\epsilon_t$ and $\delta_t$, respectively. Since for arbitrary nonlinear functions $g$ and $h$, calculating the belief of $x_t$ is almost generally not possible in closed form, the EKF calculates an approximation of the true belief by approximating it with a Gaussian. In particular the belief $bel(x_t)$ is represented by the mean $\mu_t$ and the covariance matrix $\Sigma_t$. The Kalman filter linearizes $g$ and $h$ to obtain the Gaussian nature of the belief by first order Taylor expansion. We can thus write

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) \tag{2.22}$$
$$:= g(u_t, \mu_{t-1}) + G_t g(u_t, \mu_{t-1}) \tag{2.23}$$

with $G_t$ being the Jacobian of the state. It is an $n \times n$ matrix, with $n$ being the dimensionality of the state, which includes the pose of the vehicle and when solving

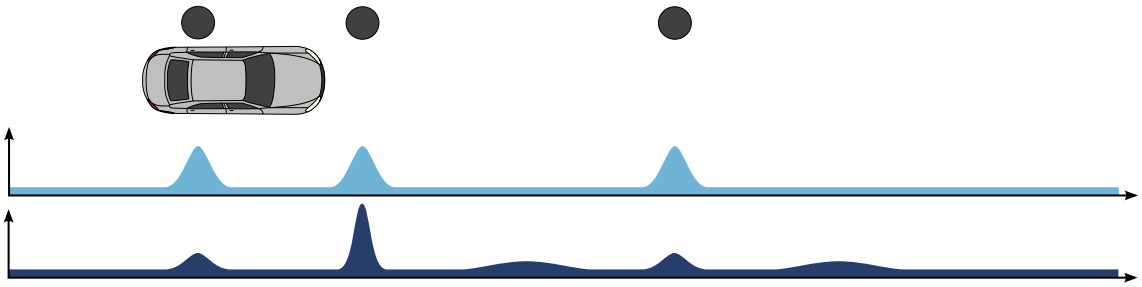
[7] Except for at $t = 1$ after initializing with a uniform distribution, because the shift would be trivial.

(a) t = 0: Initial state – the vehicle has no information and $bel(x_0)$ is assumed to be a uniform distribution.

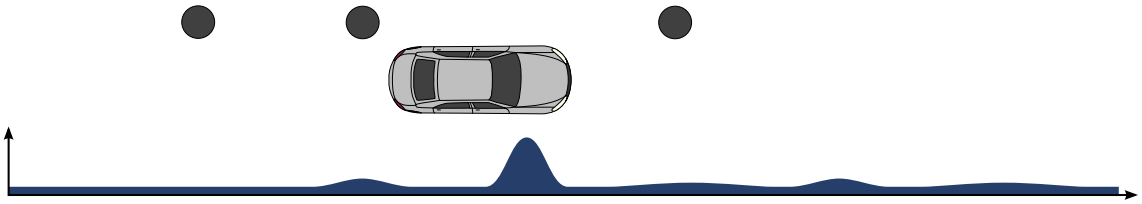


(b) t = 1: Measurement Update – The vehicle measures an object and updates $bel(x_1)$ accordingly

(c) t = 2: Control Update - the distribution $\overline{bel}(x_2)$ is determined through a shift of $bel(x_1)$ along the control $u_2$. Through the uncertainty of the control, the distribution becomes less sharp.



(d) t = 2: Measurement Update – The vehicle incorporates another measurement $z_2$ and the distribution peaks a the position the measurement was taken



(e) t = 3: Control Update – The distribution is shifted yet again. The uncertainty of the control update reduces the position confidence.

**Figure 2.12:** Visual representation of the two-fold Bayes filter process of control update and measurement update [TBF05b]. Light blue distribution: measurement update, dark blue: updated posterior and black dot: observed landmark.

the full SLAM problem, also the observations of the surroundings. We thus obtain the probability of the measurement update by a Gaussian

$$p(x_t|u_t, x_{t-1}) \approx \det(2\pi R_t)^{-\frac{1}{2}} \exp\Big(-\frac{1}{2}[x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]^T R_t^{-1} \\ [x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]\Big) \tag{2.24}$$

The same approximation can be done for the measurement model $h$ resulting in a Jacobian $H_t$ for the measurement update. We then determine the belief $bel(x_t)$ analogous to the Bayes filter approach Equations 2.18 and 2.19, but for Gaussian probabilities. We can then write the mean and covariance for the motion update as

$$\bar{\mu}_t = g(u_t \mu_{t-1}), \tag{2.25}$$
$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t, \tag{2.26}$$

where $G_t$ is the Jacobian and $R_t$ is the covariance matrix of the motion model, which is described by Ackermann steering geometry in this thesis. The measurement update is given by

$$\mu_t = \mu_t + K_t(z_t - h(\bar{\mu}_t)), \tag{2.27}$$
$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t, \tag{2.28}$$

where $K_t$ describes the Kalman gain and is given by

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}. \tag{2.29}$$

It represents the weight the measurement update of measurement $z_t$ gains over the control update. The matrix $Q_t$ describes the covariance of the measurement and $H_t$ represents the Jacobian of the measurement model $h$. The complete mathematical derivation of the EKF filter can be found in [TBF05b]. Extended Kaman filter approaches are simple to implement and yield good results in non-complex scenarios. In highly nonlinear systems though, the Extended Kalman filter approach fails due to violations of the Gaussian assumption. Additionally the EKF only allows state estimation based on the previous step. This makes recovery from false measurements very difficult. Thus EKF approaches are generally not very robust against outliers and require ample preprocessing of measurement and control inputs.

### 2.3.3 Particle Filters

The previously described method and any variation thereof have one fundamental problem. They assume that the posterior distribution is Gaussian and the control and measurement update can be approximated by a first order Taylor expansion. Now, especially when dealing with RADAR, these assumptions do not hold [RFM10]. Particle filtering has been successfully applied to robot localization with a given static map, but can be extended to fulfill the purpose of SLAM.

The FastSLAM algorithm eliminates an important drawback of EKF-SLAM. By implementing a multi-hypothesis filtering approach, singular wrong inputs into the update functions do not cause divergence of the entire system. Therefore the FastSLAM is also referred to as *Particle Filtering*. Particle filters represent the belief as a collection of samples which are drawn at random. Sampling of the belief distribution is only an approximation, but the samples are able to represent arbitrary distributions. In the context of SLAM, these samples are called particles. In general, each particle can be seen as a hypothesis of the vehicle's state vector $x_t = (x_{1:t}, m_t, w_t)$ which includes its trajectory $x_{1:t}$ and map $m_t$ as well as an importance weight $w_t$. A set of $n$ particles at time $t$ is denoted as

$$\mathcal{X}_t = \{x_t^1, x_t^2, ..., x_t^n\}. \tag{2.30}$$

In each estimation the particles are scattered in a *proposal distribution*. It is a best guess of the target distribution and can be chosen manually or result from the previous estimation. Usually an initial proposal distribution for the particles is assumed at $t = 0$. The following proposals are obtained by applying the odometry motion model to each particle taking motion noise into account. The posterior is called *target distribution*. By sampling from the proposal distribution and then assigning the importance weight to each particle based on observations of the surroundings, the target distribution is approximated as visualized in Figure 2.13.

Using this method, the particles can now approximate the target distribution, but the sampling is still based on the proposal distribution. To approximate the target distribution best, the samples should preferably be drawn based on the target distribution. In order to correct this behavior, resampling is applied.

**Resampling** To correct the samples, we consider the temporary distribution of particles displayed in Figure 2.13 (bottom). The goal is to generate a new set of $n$ particles ($n$ constant over time) that best represent the posterior distribution. This means that the state space with low probability is represented by few samples and sections of the state space with high probability should contain a lot of samples. Thus we draw $n$ new particles from the temporary set. Each particle can be drawn multiple times and the probability that each sample $x_t^i$ is drawn is given by $w_t^i$, the weight of the corresponding particle.

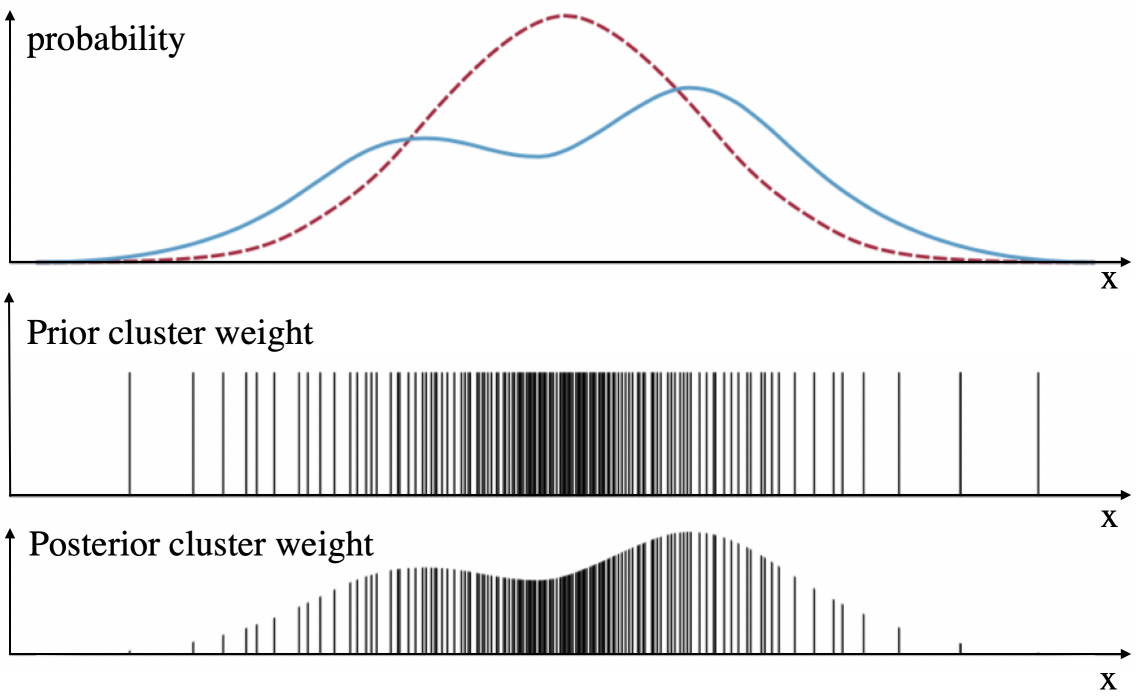



**Figure 2.13:** Top: Proposal distribution (dashed) and target distribution (solid). Middle: Particles sampled from the proposal distribution. Bottom: Sampled particles weighted according to the temporary set [TBF05b].

Generally, the more particles are used to represent the posterior distribution, the more accurate the result gets. An increasing number of particles requires a lot of processing power and memory, considering that each particle contains the entire trajectory and map in its state space. Therefore, techniques have been developed which dynamically vary the number of particles or use memory more efficiently.

However, as described above, the SLAM posterior distribution not only contains an estimate of vehicle position, but also describes the locations of map features. Sampling over such a high dimensional space would require an enormous number of particles. Thus Murphy [MG56] proposed the *Rao-Blackwellized* particle filter [GSB05] which notes that only the vehicle trajectory and its own measurement model (represented by an EKF) are enough to estimate the map. He was able to show that the probability density for a state $x_t$ can be factorized into

$$p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = \underbrace{p(x_{1:t}|z_{1:t}, u_{1:t-1})}_{\text{pose update}} \underbrace{p(m|x_{1:t}, z_{1:t-1})}_{\text{map update}}. \tag{2.31}$$

The particle filter algorithm executes the pose and map update from Equation 2.31 consecutively, including the particle update and resampling, which are depicted in Algorithm 1.

This method is called FastSLAM [TBF05b] and is a robust method of SLAM allowing for exploration into unknown terrain and finding loop closures. Over time, incorrect

**Algorithm 1** Particle Filter

**procedure** PARTICLE UPDATE
  **Input:** Particle set $\mathcal{X}_{t-1}$, motion command $u_t$
  **Output:** Posterior particle set $\mathcal{X}_t$
  $\mathcal{X}_t = \bar{\mathcal{X}}_t = 0$
  **for** $k = 1$ **to** $n$ **do**
    $x_t^k \leftarrow$ **sample_motion_model**$(u_t, x_{t-1}^k)$
    $w_t^k \leftarrow$ **process_measurement_model**$(z_t, x_t^k, m_{t-1}^k)$
    $m_t^k \leftarrow$ **update_map**$(z_t, x_t^k, m_{t-1}^k)$
    $\bar{\mathcal{X}}_t \leftarrow \bar{\mathcal{X}}_{t-1} + < x_t^k, m_t^k, w_t^k >$
**procedure** RESAMPLING
  **for** $i = 1$ **to** $n$ **do**
    $x \leftarrow$ draw from $\bar{\mathcal{X}}_t$ where probability for $i \propto w_t^i$
    $\mathcal{X}_t \leftarrow \mathcal{X}_t + < x_t^i, m_t^i >$
**return** $\mathcal{X}_t$

particles from wrong associations of features will be discarded, because they will be assigned a small weight and not replicate in the resampling step. This behavior is both an advantage and a drawback. Statistical errors are easily eliminated and particles trusting singular wrong sensor data due to their current position in the map input will be removed. Systematic errors, such as reflections from the RADAR sensor on the other hand, affect all particles the same way. This can steer the entire particle mass in an incorrect direction and the particle filter might diverge from the ground truth [Che13]. Due to the limitations of particle filters, an even more robust technique is described in the following section.

### 2.3.4 GraphSLAM

The GraphSLAM approach uses graph theory to model both the map and the trajectory of the vehicle. It has proven to outperform other approaches in terms of both mapping accuracy and robustness, as well as time complexity according to [BSG+09] and [TBF05b]. The reason for its high performance and robustness is the naturally sparse graph, which allows a solution in nearly linear time. A maximum likelihood map can be obtained by means of iterative linearization and graph optimization.

The graph based SLAM problem is strongly related to the traditional geodetic mapping problem [ABS14], since both try to answer the same fundamental question: how to minimize all errors introduced by all measurements on a large scale mapping problem? Both in geodetic large scale mapping and in GraphSLAM approaches, the minimization is based on a nonlinear least squares problem.

While the fundamental question answered is similar in geodetic mapping, applying these methods to SLAM introduces additional challenges. In geodetic mapping,

the feature association between individual measurements is typically done in an offline process that can be maintained by human refinement. An autonomous vehicle must rely on its automatic feature association to be robust enough in order to provide a good result. Typical GraphSLAM approaches are divided into a *front end* and a *back end* as depicted in Figure 2.14. The front end typically processes the sensor data and associates the incoming sensor data to the map or to previous observations to generate a graph with nodes and edges. From each associated data, a constraint is formed that is included into the full graph and thus combines all the information the sensor provides about its surroundings in a single mathematical optimization problem. The back end then optimizes the entire graph with the new information from the sensors. This mechanism does not only work unidirectional, but as Figure 2.14 shows, the front end can take a previously optimized map into consideration when performing data association and thus only adding new sensor inputs into the map. The optimization on the back end is always performed on the entire map to avoid running into local minima caused by incomplete or ambiguous sensor data.

### Graph Construction

The front end's main purpose is graph construction from the raw sensor data and to set up soft constraints between individual nodes and modeling them as edges of a graph. A general representation of a graph as used in GraphSLAM is depicted in Figure 2.15. The graph usually consists of the following components:

**Pose Node** The car-shaped node in Figure 2.15 represents the pose of the vehicle at each point in time $t = t_i$. The pose nodes usually consist of odometry measurements and are added to the graph for each new odometry reading.

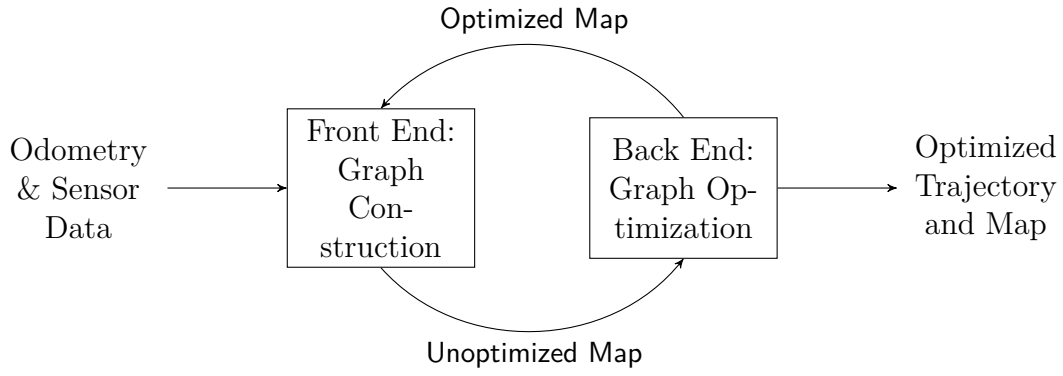



**Figure 2.14:** Fundamental architecture of the GraphSLAM algorithm.

**Pose-Pose Constraint** The gray lines between the car-shaped nodes link the poses recorded by odometry by soft constraints. Since the odometry estimates the pose incrementally $x_{1:t} = x_{1:t-1} + x_{t-1:t}$, the estimation error of $x_{t-1:t}$ is herby determined by a nonlinear function $g$, which is typically approximated by a Gaussian. Thus the error of the odometry can be modeled as

$$\epsilon^{\text{odo}}_{t-1:t} = x_{t-1} - g(x_{x_{t-1:t}}, u_t). \tag{2.32}$$

Under the Gaussian assumption the control error is normally distributed with the information matrix $\Omega^{\text{odo}}_{t-1:t}$, defined by the Fisher information [LMV+17]. If we eliminate systematic errors in the motion measurement, we can furthermore assume that $e^{\text{odo}}_{t-1:t}$ is average-free. Then the information matrix equals the covariance matrix for Gaussian error distributions. Thus we can write the cost function for the pose to pose constraints for arbitrary time steps $i$ and $j$ as [TBF05b]

$$\epsilon^{\text{odo}}_{ij}(x_i, x_j)^T \Omega^{\text{odo}}_{ij} \epsilon^{\text{odo}}_{ij}(x_i, x_j). \tag{2.33}$$

**Landmark Node** The star-shaped nodes in Figure 2.15 represent the location of the landmarks in space. Each landmark will be assigned a unique identifier to be able to differentiate them from each other. In practice obtaining these IDs from raw sensor readings requires significant effort. The important challenge of extracting suitable features for GraphSLAM is to describe them such that they can be recognized from different positions as Figure 2.15 suggests, the more poses a landmark is observed from and associated, the better the graph map. This process will be described in more detail in Section 4.3.

**Pose-Landmark Constraint** The pose-landmark constraints link the landmark to the odometry pose from which they were observed. They are depicted by the blue

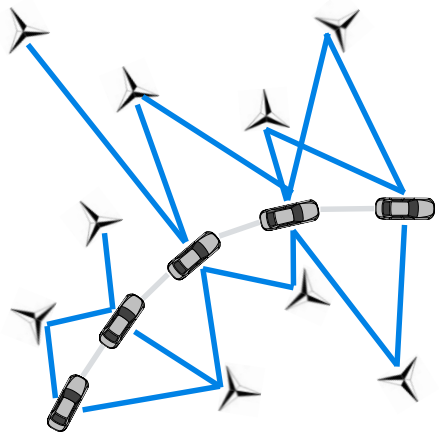


**Figure 2.15:** Model of a feature-based graph using the odometry poses and landmark observations as nodes.

line in Figure 2.15. Usually landmarks are not only observed from one single location but they are observed from multiple different spots. These repeated measurements of the same object provide the backbone structure to the entire graph and guarantee the success of the following optimization process. The sensor measures the spatial distance from the current pose of the vehicle to the object. Thus the measurement model can be used to estimate the error for each expected measurements. Thus the expected error is given by

$$\epsilon_t^{\text{sens}} = z_t - h(x_t, m), \tag{2.34}$$

where $h(\cdot)$ denotes the measurement model of our system. It returns the expected sensor measurements given a position $x_t$ and a map $m$. Analogous to the odometry constraints, average-free Gaussian distributions are assumed to formulate the constraints. This assumption is valid, because the constraint is only generated based on the distance measurement from the sensor, which – even for RADAR sensors – follows mostly linear equations. The error cost for poses $x_i$ and each landmark in the map $m_j$ can be written as

$$\epsilon_{ij}^{\text{sens}}(x_i, m_j)^T \Omega_{ij}^{\text{sens}} \epsilon_{ij}^{\text{sens}}(x_i, m_j). \tag{2.35}$$

Note that $i$ and $j$ are not indices with respect to time, but describe the unique identifiers, each landmark and pose obtain. The graph constraint definition only considers edges between specific nodes and models the time through the observations of the landmarks. Like the particle filter algorithm, GraphSLAM is able to detect loop closures, if the landmark handler in the front end assigns IDs of landmarks globally and maintains a database, such that each landmark can be found again even after a significant amount of time.

The solution to the GraphSLAM problem can be found in two different ways. The landmark-based graphs utilize all the constraints described above to generate a complete description of the world in one mathematical model, while pose graph methods rely on only pose nodes and pose-pose constraints. In this approach, landmark sightings from different locations are encoded as additional pose constraints. In [TBF05b] it is furthermore shown that the amount of information in both approaches is the same and that one can obtain a pose graph from a full graph by pruning all landmark nodes and collapsing all sets of pose-landmark constraints leading to the same landmark into a single pose-pose constraint.

### Graph Optimization

For the graph optimization, the more general case of the full GraphSLAM problem is considered. The graph optimization is identical to the mathematics of bundle adjustments from geodetic mapping techniques and solves the problem of refining the coordinates of individual features (nodes) given measurement errors. The minimization can be achieved by means of a nonlinear least-squares method to determine the global minimum of the cost function, taking all constraints into account.

The cost function of GraphSLAM is the sum of all constraints of the above described components and thus given by

$$J_{\text{Graph}} = \underbrace{x_0^T \Omega_0 x_0}_{\text{map origin constraint}} + \sum_{ij} \underbrace{e_{ij}^{\text{odo}}(x_i, x_j)^T \Omega_{ij}^{\text{odo}} e_{ij}^{\text{odo}}(x_i, x_j)}_{\text{odometry constraints}} + \sum_{il} \underbrace{e_{il}^{\text{sens}}(x_i, m_l)^T \Omega_{il}^{\text{sens}} e_{il}^{\text{sens}}(x_i, m_l)}_{\text{landmark measurement constraints}}. \tag{2.36}$$

It contains the initial constraint $x_0^T \Omega_0 x_0$ defining the center of the reference frame and the sum of all landmark and pose constraints. The goal is now to minimize $J_{\text{Graph}}$, specifically

$$x_{\min} = \arg\min_x J_{\text{Graph}}. \tag{2.37}$$

**Least-Squares Problem** Analogous to [Küm13], let $\mathcal{C}$ be the set of all constraints, such that Equation 2.36 can be simplified to

$$J_{\text{Graph}} \equiv F(x) = \sum_{c \in \mathcal{C}} e_c(x)^T \Omega_c e_c(x). \tag{2.38}$$

Since $F(x)$ is a nonlinear system, the minimum can typically not be found analytically. Let therefore $\hat{x}$ be the initial guess for $x_{\min}$. Therefore, we solve the equation iteratively. Let $\hat{x}$ be the initial guess. Then we approximate the error term linearly around $\hat{x}$ and obtain

$$\begin{aligned} e_c(\hat{x} + \Delta x) &\simeq e_c(\hat{x}) + \mathbf{J}_c \Delta x, \\ \text{where} \quad \mathbf{J}_c &= \frac{\partial e_c(\hat{x})}{\partial \hat{x}}. \end{aligned} \tag{2.39}$$

Thereby $\mathbf{J}_c$ denotes the Jacobian of the error function in $\hat{x}$ for the given constraint $c$. We can rewrite Equation 2.38 as the following

$$\begin{aligned} F_c(\hat{x} + \Delta x) &= \sum_{c \in \mathcal{C}} e_c(\hat{x} + \Delta x)^T \Omega_c e_c(\hat{x} + \Delta x) \\ &\approx \sum_{c \in \mathcal{C}} (e_c(\hat{x}) + \mathbf{J}_c \Delta x)^T \Omega_c (e_c(\hat{x}) + \mathbf{J}_c \Delta x) \\ &= \sum_{c \in \mathcal{C}} e_c(\hat{x})^T \Omega_c e_c(\hat{x}) + 2 e_c(\hat{x})^T \Omega_c \mathbf{J}_c \Delta x + \Delta x^T \mathbf{J}_c^T \Omega_c \mathbf{J}_c \Delta x \\ &= \sum_{c \in \mathcal{C}} const + 2 \sum_{c \in \mathcal{C}} e_c^T \Omega_c \mathbf{J}_c \Delta x + \sum_{c \in \mathcal{C}} \Delta x^T \mathbf{J}_c^T \Omega_c \mathbf{J}_c \Delta x \\ &:= const + 2 b^T \Delta x + \Delta x^T H \Delta x. \end{aligned} \tag{2.40}$$

Note that the cost function after approximation can be written as a function of $\Delta x$ only and we call it $G(\Delta x) := const + 2b^T \Delta x + \Delta x^T H \Delta x$. This function is of

quadratic form and therefore has a global minimum. The minimum $\Delta x_{\min}$ with respect to the initial guess can be found by taking the second derivative:

$$\begin{aligned} G'(\Delta x) &= 2b + 2H\Delta x \stackrel{!}{=} 0 \\ G''(\Delta x) &= 2H \geq 0. \end{aligned}$$

The minimum can thus be found as the solution of the first derivative $G'(\Delta x_{\min}) = 0$. It follows that

$$H\Delta x_{\min} = -b. \tag{2.41}$$

This procedure finds an adequate solution $x_{\min}$ that is closer to the global minimum than the initial guess. The Gauss-Newton method repeats this minimization for a quadratic approximation and converges to the minimum of $J_{\text{Graph}}$. A further defined refinement is given by the Levenberg-Marquardt algorithm and is defined below.

**Gauss-Newton Algorithm** The Gauss-Newton algorithm is a popular method used to solve nonlinear least-squares problems. It utilizes the basic optimization method as described above, but applies it iteratively. This process is displayed in Algorithm 2. The procedure of the Gauss-Newton algorithm first requires a manual initial guess of the position to enable the first linearization. In each further iteration, the solution of the previous step serves as the initial guess. The difference between the current and the previous solution is given by $\Delta x$. Once the solution has converged, the solution is returned.

While this method is more reliable than using a single least square estimation of the minimum, the Gauss-Newton method is still very dependent on a good initial guess and is prone to return local minima close to the initial guess. But since GraphSLAM is interested in the global optimum, there is still room for improvement.

**Algorithm 2** Gauss-Newton Algorithm

```
1: procedure GAUSS-NEWTON
2:     Input: x̂ initial guess
3:     Output: x_min approximated minimum
4:     x ← x̂
5:     while E < threshold : do
6:         (H, b) ← build_linear_system(x)
7:         Δx ← solve(H, b)
8:         E ← error(Δx)
9:         x ← x + Δx
10: return x
```

**Algorithm 3** Levenberg-Marquardt Algorithm

```
1: procedure LEVENBERG-MARQUARDT
2:     Input: x̂ initial guess
3:     Output: x_min approximated minimum
4:     while E < threshold : do
5:         (H, b) ← build_linear_system(x)
6:         E ← error(x, x_old)
7:         Δx ← solve((H − λI)Δx = −b)
8:         x ← x + Δx
9:         if E < error(Δx) then
10:            x ← x_old
11:            λ ← 2λ
12:        else
13:            λ ← λ/2
14: return x
```

**Levenberg-Marquardt Algorithm** Another method to solve nonlinear least squares problems is the Levenberg-Marquardt optimization algorithm [Mar63]. The approach uses the least-square linearization method iteratively, similar to the Gauss-Newton approach, but it inserts a dampening factor and a backup and restore mechanism at each step. The update is computed via

$$(H + \lambda I)\Delta x = -b. \tag{2.42}$$

The dampening factor $\lambda$ can be added to the linear equation system, because any superposition of solutions of $G'(\Delta x) = 0$ is also a solution.

The dampening factor serves two purposes: On the one hand, the rank of the matrix $H$ may become incomplete if the solution is far from the initial guess and a standard Gauss-Newton approach will terminate without a solution. This can be mitigated, because $\lambda I$ always has full rank. Another benefit of this approach is that the dampening factor may be adjusted dynamically to control the step size of the minimization. If the initial guess is far from the minimum, the step size is increased, whereas when it is close to the minimum, the step size is reduced. Thus the Levenberg-Marquardt algorithm will converge in much fewer iterations. The algorithm is denoted in Algorithm 3. Note that the dampening factor is increased or decreased depending on the error function.

**Robust Kernels** The above described optimization methods will find an adequate minimum to solve the GraphSLAM problem if the constructed graph has reasonably much information about its surroundings. Bad sensor measurements will automatically be weighed less in comparison to more reliable sensor measurements. Thus for the RADAR sensor well formed landmarks on highly reflective surfaces will count more towards the overall solution than bushes and trees with high noise and low reflectivity. But because of the systematic errors of the RADAR sensor described in Section 2.2, there are many ways that false constraints can find their way into the cost function $J_{\text{Graph}}$. As the graph optimization was described so far, these faulty constraints will affect the global minimum significantly, if they result from highly reflective patterns such as the speckle displayed in Figure 2.7. Thus an external method to dampen the constraint as a whole is required.

In order to prevent outlier constraints from deteriorating the optimization, robust kernels are introduced as yet another modification of the cost function that mathematically does not affect the minimum. Consider a single constraint $F_c = e_c^T \omega_c e_c$. Note that the error $e_c$ has a quadratic influence in the optimization. Thus we need to introduce a robust kernel $\rho(\cdot)$ mitigating the influence on the optimization for very large errors. The typical robust kernels described in this thesis are displayed in Figure 2.16. They can have the following form:

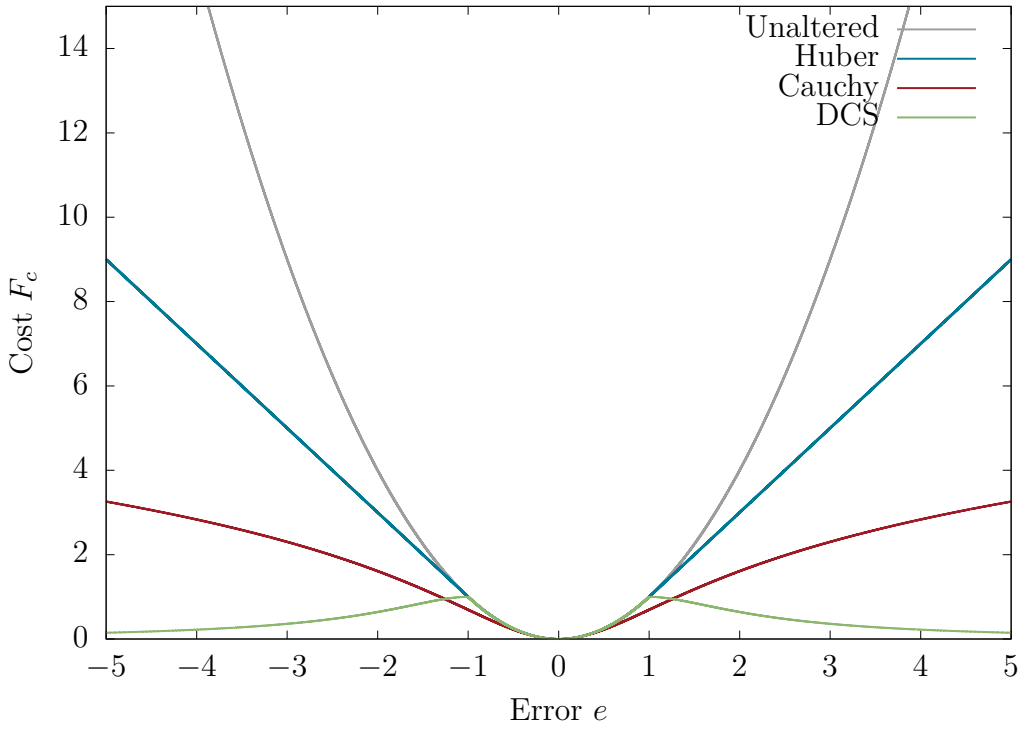



**Figure 2.16:** The robust kernels plotted alongside the unaltered quadratic cost function.

$$F_c = \rho(\sqrt{e_c^T \omega_c e_c}), \tag{2.43}$$

where $\rho(\cdot)$ is a symmetric function that leaves the solution unchanged. Several variations of the robust kernels were proposed in the past, one of them being the Huber kernel [Hub73]:

$$\rho_{\text{Huber}}(x) = \begin{cases} x^2 & \text{for } |x| \leqslant \lambda \\ 2\lambda \cdot |x| - \lambda^2 & \text{for } |x| > \lambda \end{cases}. \tag{2.44}$$

The Huber kernel retains the same behavior as the quadratic least-squares problem for costs below $\lambda$. If the error becomes larger than $\lambda$ the cost function increases only linearly in $e$ and not quadratic.

The variant used in the the GraphSLAM techniques in this thesis is the Cauchy kernel, which is defined by

$$\rho_{\text{Cauchy}}(x) = \lambda^2 \cdot \log\left(\frac{x^2}{\lambda^2} + 1\right). \tag{2.45}$$

Compared to the Huber kernel, the Cauchy kernel grows slower and the influence of high costs is reduced even further.

Finally, the Dynamic Covariance Scaling (DCS) kernel proposed by [ATS+13] is given by

$$\rho_{\text{DCS}}(x) = \begin{cases} x^2 & \text{for } x \leqslant \lambda \\ (wx)^2 & \text{for } x > \lambda \end{cases} \text{ with } w = \frac{2\lambda}{\lambda + x^2}. \tag{2.46}$$

It reduces the influence of the higher-error constraints even in the assumption that all large errors are outliers. This will cause the optimization to only significantly affect local portions of the branch, as will be seen in real RADAR data in Section 4.4. These kernels are an essential ingredient to successful graph optimization using RADAR data, because it allows for adaptation to the physical properties of the sensor. In the following sections the feature extraction and SLAM methods are described in further detail.

**Loop closures** and the recognition of them play a significant role in the graph optimization process. If constraints are only found between consecutive poses, a part of the motion errors introduced into the graph cannot be recovered by the graph optimization. In order to keep the map consistent nonetheless, the feature detection algorithm must be able to detect these loop closures by recognizing previously seen landmarks. Then, constraints can be introduced into the graph that can remove the motion inaccuracies fully.

### 2.3.5 Other SLAM Algorithms

Besides the major categories of algorithms described in the previous sections, the research field of SLAM has been active over the past years, resulting in a large variety of SLAM algorithms for different settings and environments. There are many variations and adaptations to specific scenarios based on the fundamental principle of EKF-SLAM, FastSLAM and GraphSLAM. An overview is outlined in [LCW+12].

The main distinction of these approaches is the map representation and measurement model, which can be divided into feature-based algorithms, such as [TM05] and grid-based ones, as overviewed in [CCR07]. Feature-based approaches store locally condensed information about the environment, usually as point features, which results in low memory consumption, at the cost of reduced robustness due to association problems [ZCS+08]. These techniques are common for camera sensors [ZLS+14] and have recently been applied to the RADAR sensor [RDH+16b]. Grid-based algorithms on the other hand have proven to be effective when applied to range sensors, such as laser scanners [EP03].

**Grid-based Approaches** are usually combined with particle filters, because they constantly validate multiple hypotheses to determine the most likely map. A lot of effort was put into making grid lookups more efficient and to access information from large datasets quickly. This can be done by approximating regularly shaped areas in the map by polygons [PT05], by making spatial search more efficient with octrees [FKW07, WHB+10] or by modifying the measurement model of the particle filter such that it operates in a single global map [EP03].

Another map representation similar to the grid-based approach utilizes the idea of GraphSLAM and Normal Distribution Transformation (NDT), intended for lifelong navigation [EG13] in an attempt to make the SLAM problem independent of the underlying sensor using NDT and a pose graph optimization. This approach is largely independent of the sensor in terms of data structure (i. e. point clouds can come from any sensor), but does not take into account unique physical phenomena, inherent to the different sensors, such as RADAR.

In [KHD+09] a multi-layer grid-based approach is presented that obtains consistency between each layer by matching the layers to one another. The joint registration of individual layered floors of a large parking center provides a highly accurate map of a multi-level parking lot using a precise LiDAR sensor. The presented approach stores the interconnectivity of parking layers in the map through the edges of the graph. In this paper however, the map generation is handled by multiple input sources by a crowd of vehicles. All maps are therefore completely independent from each other and have to be unified into one global map.

**Feature-Based Approaches** have been mainly used with GraphSLAM and EKF-SLAM in the past [TM05, PRLM+09]. Compared to grid-based approaches, memory requirements decrease, while the number of preprocessing steps to achieve a good map increases significantly. In addition to the feature extraction of robust features, the feature association problem needs to be solved. These techniques are mostly applied to image sensors, since feature extraction is a common task in image processing [ZLS+14] and methods for outlier rejection are abundant.

An advantage of feature-based approaches is the added flexibility. Feature-based maps merely consist of a list of data points and can easily be updated by adding or removing them. It is more involved, however, to keep grid maps up to date [ZCS+08, MDBB12].

**Localization using a RADAR sensor** has been recognized in literature in recent research [DAB+14]. Grid maps are used to aggregate the RADAR data into a map that can be used for a semi-Markov chain localization [RHT+15], which was performed on static maps. The full SLAM problem has not been addressed in this approach. Thus, unlike [DAB+14], the presented approach in this thesis provides a solution to the SLAM problem generating an optimized map.

The RADAR sensor has been less popular for solving the SLAM problem, especially because of its noise characteristics and the fact that in an automotive RADAR sensor one typically only has access to the detections [RSH+16] of the sensor and not the full spectral information. Each detection consists of a range, an angular and an amplitude measurement. There have been approaches to simulate the behavior of a RADAR sensor to make the automotive grade hardware more accessible [BY06], but there has not been an extensive real world dataset in the past.

Exploration of feature-based algorithms or solutions to the full SLAM problem has yielded insular results with very different approaches [MVAV11]. Hence in localization for seaborne vessels [CTG+11], after initial shape recognition, the final localization algorithm is also performed on a grid map.

The resulting map is treated as an image, such that common feature detection algorithms generate landmarks on the coast used for SLAM. Another approach [CGB+10] uses scan matching on the reflected RADAR spectra in combination with EKF-SLAM. By Fourier-Mellin transformation, the authors find an efficient technique to match scans and generate hypotheses for the vehicle pose. This approach relies on the full spectral information. For the task of object classification using shape information, such as cars [DHS+14], grid maps can be utilized.

Beside using the grid map representation for localization, one can also use it as an intermediate step for feature extraction [RGH+13]. This way data is locally aggregated into a grid. Then the information is reduced in the form of individual features. While the feature extraction in [RGH+13] is used for grid map registration [TM05], it is shown in later sections that they can be used for a full SLAM.

**Contributions to Lifelong Mapping** is one of the most challenging variants of the SLAM problem, as it describes the unlimited maintenance of the map without any human interaction. This thesis tries to solve this problem for the specific scenario of a parking lot. In this specific scenario of SLAM in a defined area at different times of the day, utilizing different vehicles, the environment is too dynamic to achieve a robust localization with an outdated map, so it is vital to actively update the map with more recent sensor data.

While robust localization algorithms can be designed that ensure a long half-life of a map [ZLS+14], a continuous update allows for the most robust localization [ZCS+08]. Map updates have been studied in the past using both landmark maps [SWK+16] and grid maps [MDBB12]. These approaches decide on a landmark or cell based level whether it is currently up-to-date.

In order to handle changing environments, there are approaches that create temporary maps [MDHGB10] to account for dynamic changes of the environment, thus generating multiple map hypotheses for environments that looked differently in the past. This improves localization, but it does not scale for rapid changes or a high number of cars contributing, because it would generate a large amount of temporary maps. Our approach maintains one globally consistent solution across inputs from multiple vehicles and under changing environments. This has many advantages. Mainly the map can be kept lean by pruning redundant feature information that will only decrease performance without adding new information.

While with feature-based SLAM, the map update can be achieved fairly easy, more complex methods have to be devised to keep grid maps up to date [ZCS+08], [MDBB12]. Because of the static nature of grid maps, transition matrices to update each cell with new data have to be defined in order to calculate the probability of grid cell occupancy.

## 2.4 Conclusion

In this section, we have highlighted the theory behind the RADAR sensor and its Key Performance Indicator (KPI)s, such as resolution and SNR. Furthermore the physical phenomena when working with an automotive RADAR on real data was explained. RADARs emit microwave light of about 77 GHz either as a modulated continuous wave CW or as pulsed frequency bursts. The most important KPI of the sensor were introduced, including the RADAR resolution and range, depending on the frequency bandwidth $B$ and the repetition frequency $f_p$ respectively. Furthermore the RADAR equation shows us the decline of received power proportional to $r^{-4}$, which determines the signal to noise ratio of the sensor. The latter can be increased by better angular resolution while the vehicle is moving because of the Doppler frequency. Finally a series of physical effects of the described phenomena in the real world were discussed.

Additionally the theory behind SLAM algorithms, especially RADAR based SLAM was introduced. Here, EKF-SLAM utilizing a Kalman filter update scheme to estimate the posterior distribution was introduced. The FastSLAM based approach on the other hand samples from the prior distribution and propagates each sample, also called particle with measurement update and control update to determine the posterior distribution.

It was found explaining the different approaches to SLAM that for the automotive case, a landmark based concept is the most promising, as it eases map updates and does not come with large computation or memory demands. Furthermore the GraphSLAM approach, which is the focus of this thesis, is introduced. It interprets the SLAM problem as a graph optimization problem, generating a set of constraints from measurements and odometry poses that can be optimized using Gauss-Newton or Levenberg-Marquardt methods. It will be shown in Chapter 4 that feature based solutions SLAM with RADAR fulfill the requirements for autonomous driving purposes on parking lots.

Most of the current publications in the field of SLAM are based around those three fundamental methods. Specific forms of these algorithms were surveyed in the related work in the area of SLAM and RADAR based localization. In the following chapters, the setup and dataset used in this thesis, as well as the applications of the elaborations of this chapter will be discussed.

# 3 Experiments

The experimental setup is based on a Mercedes-Benz E-Class with additional sensors, that could be interfaced directly by regular PC hardware. This setup enabled the execution of a large amount of experiments collecting a sizable data set with a broad sensor setup to validate the approaches presented in this thesis.

## 3.1 Sensor Setup

The vehicle is equipped with a set of standard and non-standard sensors for evaluation to use in future products. An overview of the sensors and their opening angles are depicted in Figure 3.1.

While in normal operation, the sensor data is only transmitted to specific ECUs within the vehicle to perform very specific tasks, the goal for the autonomous driving project is to utilize all of them at the same time and in the same framework.

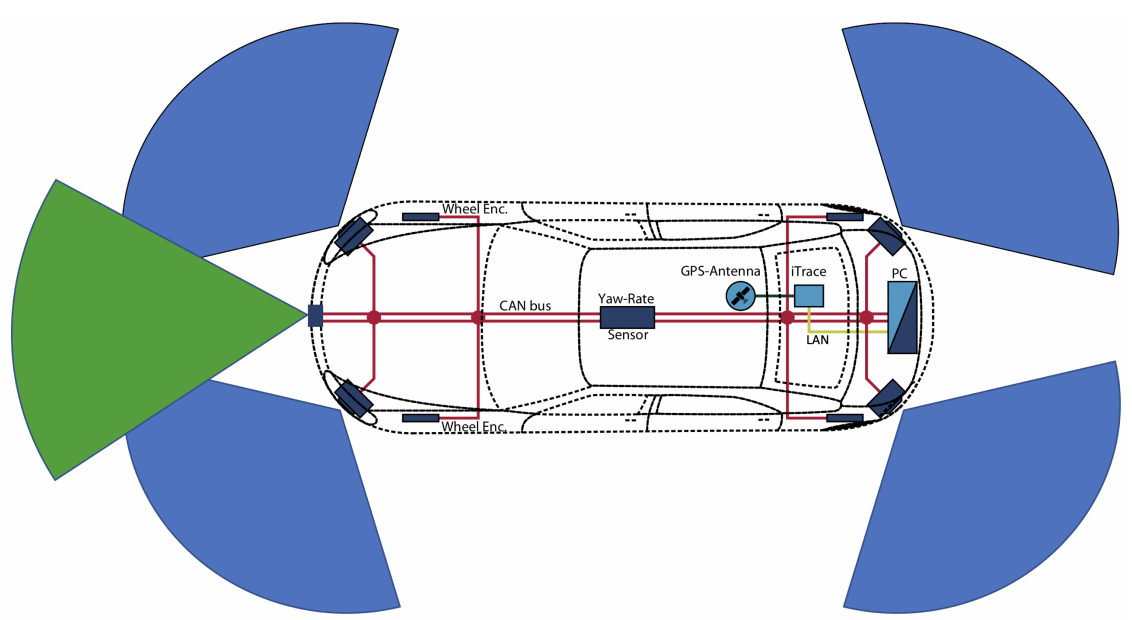



**Figure 3.1:** Schematic representation of the vehicle equipped with additional sensors. The LiDAR opening angle can be seen in green and the four RADAR opening angles are displayed in blue.

Different sensors for different use cases are also communicating with different protocols depending on data rate requirements, as well as security and timing aspects. The most common used in the automotive industry are Controller Area Network (CAN), FlexRay and Ethernet. All of these data sources need to be bundled in a single computing unit and aligned according to their specific characteristics for the data to be usable for the autonomous driving system.

**The Computing Hardware** used in this setup is a standard Intel Core $i7-3820QM$ desktop PC with 16 GB of onboard storage and an Nvidia $GTX\ 780\ Ti$. It houses interface cards for both CAN and FlexRay as well as standard $10G$ Ethernet. It serves as the single computing unit where all algorithms for the autonomous driving setup, not only the SLAM algorithm, is being run. In order to combine all the sensor information in an underlying framework, the Automotive Data and Time-Triggered Framework (ADTF)[1] on Linux is being used to retrieve the data and assign concurrent timestamps to all incoming signals. It handles the processing from pressing a single button on the steering wheel to high data rate LiDAR sensors all in one framework. All further remarks about algorithm runtime and memory consumption will be based on this system.

**Odometry Sensors** are located in every wheel and use magnetic encoders to determine how far the vehicle has moved within a certain time interval. Each wheel has 96 permanent magnets generating 96 pulses per revolution. The wheel encoders are thus capable of determining the velocity with a resolution of $u/96$, whereas $u \approx 2\,\mathrm{m}$ is the circumference of the wheel. Additionally to the wheel encoders, there is a Micro Electro Mechanical System (MEMS) based IMU located in the ESP ECU, which transmits the roll, pitch and yaw rate to the computing unit. The odometry information updates every 50 ms and is transmitted over CAN, which can cause non-constant latency of the odometry signal, resulting in larger than expected drifts.

**The RADAR Sensors** are located in the corners of the vehicle behind the bumpers. Since the bumpers are made out of a thin plastic, the RADAR is able to penetrate the material without much loss of signal strength.

The Frequency Modulated Constant Wavelength (FMCW) RADAR sensor specifically built for automotive applications emits millimeter waves and records the reflected power distribution with respect to distance, the so called A-scope. In the automotive industry though, A-scopes are preprocessed and packaged into target lists, containing a reduced amount of information [BY06].

This configuration enables a 360° vision (with blind spots close to the sides of the vehicle) of the environment. Each RADAR operates at a frequency of 76 GHz and is

[1] ADTF was developed by a consortium of the automotive industry and is widely being used for experimental sensor integration and data acquisition.

able to transmit up to 64 data points [SKR+16] . The maximum range in the data sheet is listed as 40 m [RHT+15]. The angular resolution of the signal is 1°. A typical RADAR sensor can be seen in Figure 3.2 (a). These numbers have proven realistic in the experiments conducted in this thesis, with the exception of measurements near standstill, where the angular resolution decreases, as explained in Section 2.2.3.

**The LiDAR sensor** is mounted in the center of the front bumper and is very unobtrusive compared to other LiDAR sensors like a Velodyne. It is the Ibeo Scala research sensor, such as the one described in Ibeo's technical report [ibe19] and displayed in Figure 3.2 (b). It has an opening angle of 145° with a resolution of 0.25°. It has four layers and a vertical field of view of 3.2°. Its distance resolution is listed as 4 cm and the update rate is 40 ms, while only two layers are transmitted per 25 Hz cycle. The sensor alternates between transmitting Layer 1 and 3 and Layer 2 and 4, effectively giving each layer an update rate of 12.5 Hz.

Due to the low mounting position and the narrow field of view, objects visible from LiDAR are typically curbs, tree trunks and parked vehicles. The automotive LiDAR does not provide high resolution point clouds of the entire environment like a roof mounted Velodyne would.

**The Stereo Camera** equipped in the vehicle was used only sparingly within this setup. It consists of the same setup as used by [Muf18], which contain two synchronized cameras with 1400 × 1024 px with a FOV of 80° and a focal length of 740 px [MMPF]. The images are cropped vertically to 1400 px × 400 px to focus on the relevant scene content. The Stereo processing with Semi-Global Matching (SGM) [Hir06] was performed using a real-time implementation on a Field Programmable Gate Array (FPGA).

**Reference Positioning Sensors** were equipped to the vehicle to measure the accuracy and reliability of the localization. The system for the dataset of this thesis was

(a) Typical RADAR sensor unit from Continental AG [con19]


(b) Typical LiDAR sensor unit from Valeo Automotive [ibe19]

**Figure 3.2:** Representations of the sensors used in the prototype vehicle.

an iMAR iTrace F400-E [iMA14]. It uses a deeply coupled DGNSS/INS system fusing a high-end dual frequency GNSS receiver with a military grade IMU. According to the dataset, the precision of the device is $2\,\text{cm} \pm 2\,\text{mm}$, which is accurate enough to validate the position requirements targeted in this thesis, hence the iTrace system is furthermore referred to as the ground truth.

**Reference Mapping Sensors** were used to validate the map with an accurate point cloud. It was generated using a *total station*, the Leica MS50 [ZLN09]. It was operated in scanning mode to generate a millimeter accurate 3D point cloud of the parking lot where most of the experiments were performed. The map was then manually pruned of obstacles to generate a reference map to compare the SLAM results to. The matching to the maps under test is done manually by identifying common points from both maps visually. Newer total stations also generate textures in addition to the highly accurate point clouds that can be used to generate even more accurate reference maps.

## 3.2 Coordinate System and Calibration

To fuse the sensor information into a consistent result, each sensor has to be extrinsically calibrated beforehand. For this thesis the vehicle coordinate system is chosen based on the DIN ISO 8855 [iso11], which defines the origin a right-handed Cartesian coordinate system centered on the rear axis midpoint. The coordinate systems of each individual sensor can be seen in Figure 3.3.

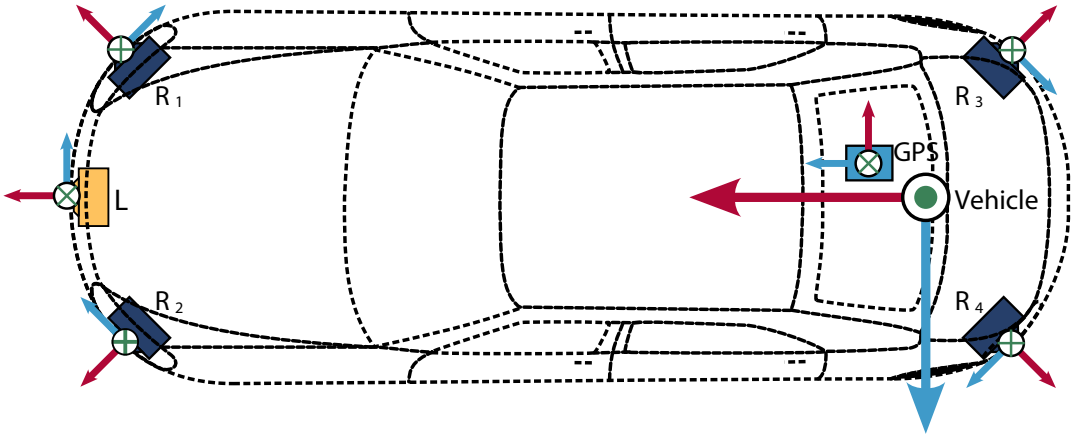



**Figure 3.3:** The sensor coordinate systems for all relevant components (RADARs $R_1$ through $R_4$ and LiDAR L). All sensor inputs are calibrated to the middle of the rear axis of the vehicle. The red and blue arrows represent the $x$ and $y$ axis, respectively and the green dots/crosses represent the $z$ axis.

The extrinsic mounting positions of the sensors are determined during mounting by precisely measuring a reference target that is being tracked by a total station. By also mounting a reflector on the vehicle close to the rear axis, the extrinsic calibration parameters can be estimated. In addition to this initial calibration, the RADAR sensors could be online calibrated by a hierarchical calibration routine [KBD+15] to continuously update and improve the calibration. For any offline processing done in this thesis, the calibration is considered to be static though.

The GNSS/INS sensor was mounted close to the rear axis to ensure a fast-converging and precise calibration to the rear axis. The iTrace F400-E comes with an integrated calibration mode that requires the vehicle to be driven in a series of shapes to estimate the calibration parameters from the vehicle dynamics. The approach is proprietary to iMAR, but appears to be close to the method described in [ABY+13].

## 3.3 Dataset

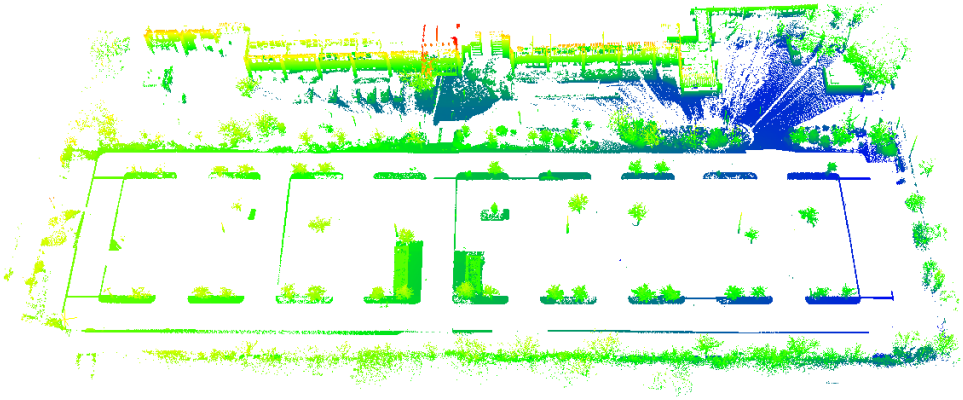
The parking lot presented in Figure 3.4 (a) was used for most of the experiments performed in this thesis. To perform reliable benchmarks for SLAM, not only the positioning accuracy, but also the accuracy of the map is of interest. Thus the parking lot was digitalized into a highly precise, georeferenced reference map, which can be seen in Figure 3.4 (b). The parking space consists of an area of approximately 150 m × 35 m with nine double rows of parking spots and is surrounded by fences and vegetation.

The parking lot looks quite different during the day. Thus, when collecting data from different times of the day and on different days, the configuration of parked cars is always significantly different. This setting was used to generate two main interesting datasets to evaluate the contributions of this thesis:

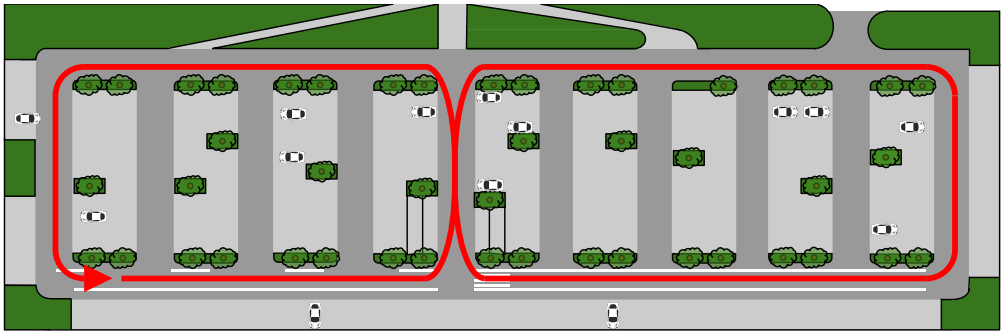
**The eight shaped** dataset consists of many different measurement days spread across different times of the day. In each sequence, the trajectory across the parking lot is similar and encompasses most of the parking lot surface. Thus the changing environment and the effects on the proposed SLAM algorithms can be studied in detail. As can be seen in Figure 3.4 (c), the trajectory has two loop closures that should give immediate visual feedback on how consistent the SLAM algorithms perform globally.

**The realistic parking maneuvers** are collected to simulate real world scenarios of vehicles entering the parking lot and finding a suitable parking location autonomously. The single entrance to the parking space is the common origin of all trajectories. As can be seen in Figure 3.4 (d), each trajectory then finds a different parking spot and comes to a stop. The trajectories do not contain any loop closures

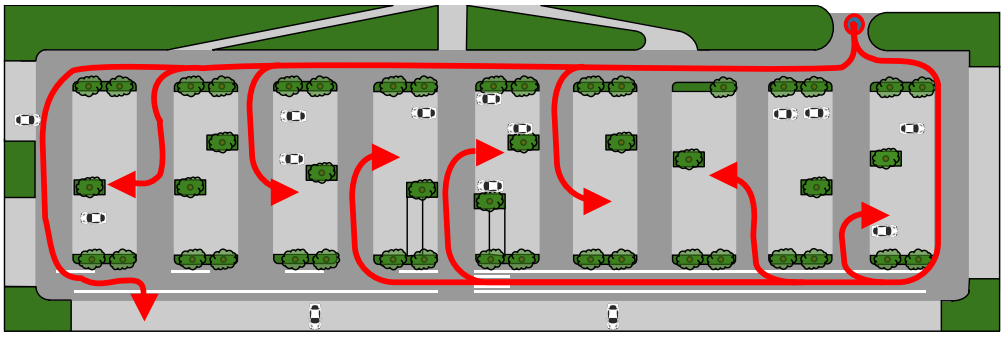
(a) Aerial picture of the parking lot (Source: Google Maps).



(b) 3D reference point cloud from a total station pruned from mobile objects.



(c) The eight shaped trajectory.



(d) The realistic parking maneuvers.

**Figure 3.4:** A Google Maps plot of the parking lot, alongside the 3D reference point cloud and the available datasets.

and represent the real world scenario of having short trajectories when the parking lot is empty and long trajectories, when the parking lot is mostly occupied.

## Characterization of Ground Truth

Ground truth of the vehicle pose has been recorded across the entire data set and a statistical analysis of the standard deviation reported by the GNSS/INS system is given by the following quantities. The mean error margin of the ground truth was determined using the iTrace internal performance estimation. We consider the latitude and longitude accuracy, as well as the Euclidean error

$$\bar{\sigma}_{\mathrm{GT,d}} = \sqrt{\bar{\sigma}^2_{\mathrm{GT,lat}} + \bar{\sigma}^2_{\mathrm{GT,lon}}}. \tag{3.1}$$

For the entire data set, the accuracies of the ground truth are given by:

$$\begin{aligned} \bar{\sigma}_{\mathrm{GT,lat}} &= (0.071 \pm 0.232)\,\mathrm{m} \\ \bar{\sigma}_{\mathrm{GT,lon}} &= (0.062 \pm 0.213)\,\mathrm{m} \\ \bar{\sigma}_{\mathrm{GT,d}} &= (0.094 \pm 0.315)\,\mathrm{m} \end{aligned}$$

The result on the actual data set thus varies significantly from the data sheets specifications, but is still suitable to serve as a ground truth system for the real test ahead, because a mean error of below 10 cm is sufficient to prove reliability for autonomous driving scenarios.

The increased deviation can be explained by the specific scenario. Parked cars, as well as the multi story building, as well as the trees of the parking lot can block a few satellites from view of the Global Navigation Satellite System (GNSS) antenna. Additionally, the metallic surface of the vehicle's roof may cause multiple reflections and thus falsify signal travel times.

# 4 Radar Based SLAM

In RADAR-based SLAM it is essential to take the physical properties of the sensor into account. While the sensor has been widely used for object tracking, it was never applied to SLAM so far. In the following sections the methods specifically designed for RADAR sensors are described.

## 4.1 Radar Grid Mapping

As discussed previously in Section 2.3.5, grid maps are common representations of the world and allow for SLAM algorithms like FastSLAM to work efficiently on two dimensional grid maps. In this chapter, the approach by [DWR+15] is discussed in detail to show the drawbacks and merits of grid map representations for RADAR data.

The idea of discretizing the world into grid cells and representing the environment by measuring the occupancy probability of each grid cell was first introduced in sonar based mapping [ME85]. The two main assumptions for this technique are that the world is static and that for any given time $t$ the pose of the vehicle $x_t$ is known. We can then discretize the the sensor view into a set of $\mathcal{M} = \{m_i | i = 1...n\}$ two dimensional cells. Each cell stores the probability of occupancy of the area that it represents. In the following section, we will present the sensor model, translating sensor measurements into occupancy probabilities.

### 4.1.1 The Sensor Model

The sensor uncertainty model proposed by [DWR+15] for the RADAR sensor assumes that the world is static and that each target can be described by a range and an angle measurement (two dimensional) and an amplitude. The received RADAR amplitude thus depends on three different physical properties:

**The radial distance** of the target from the sensor is the biggest influence on the amplitude, because the amplitude $A \propto \frac{1}{r^2}$, as given by Equation 2.11. To achieve a normalized amplitude, we increase the sensor sensitivity based on the measured distance, compensating the dampening behavior at the cost of a higher SNR for larger distances.

**The material, shape and viewing angle** of the RADAR target helps identify known targets [WBK+14]. These factors describe the RADAR cross section $\Gamma_t$ of the target.

**Disrupting factors** such as noise and multiple scattering (see Section 2.2.3) need to be accounted for in the measurement model to incorporate the specific behavior. This can be achieved by the Swerling models for the RADAR cross section $\Gamma_t$ of observed targets. To build the grid map, the Swerling III model [Sko08] is used in this thesis, assuming a target consists of one major and several minor scatters, then the probability $p(\Gamma_t)$ of a cell being occupied is $\chi^2$ distributed

$$p(\Gamma_t) = \frac{4\Gamma_t}{\widehat{\Gamma_t}^2} \exp\left(-\frac{2\Gamma_t}{\widehat{\Gamma_t}}\right), \tag{4.1}$$

where $\widehat{\Gamma_t}$ is the expected value of $\Gamma_t$. Furthermore, the distribution of the received amplitude is then given by

$$p(A) = \frac{8A^3}{\hat{A}^4} \exp\left(-\frac{2A^2}{\hat{A}^2}\right) \text{ with } \hat{A} = c\sqrt{\widehat{\Gamma_t}}, \tag{4.2}$$

with the sensor specific constant $c$. The expectation and standard deviation of the above $\chi^2$ distribution then follows as

$$\mu_A = \frac{3}{4}\sqrt{\frac{\pi}{2}}\hat{A},\ \sigma_A = \hat{A}\sqrt{1 - \frac{9}{32}\pi}. \tag{4.3}$$

This RADAR specific sensor model can be used to build a grid map by aggregating the detections into a grid and assigning probabilities to each cell by using the above distribution function to determine the probability of a cell being occupied.

### 4.1.2 Layers of the RADAR Grid

The RADAR sensor returns the distance $r$, the bearing $\varphi$, the relative radial speed and the amplitude $A$ of all targets within the FOV. The amplitude $A$ and the relative position in space are the most interesting information to build the grid map $\mathcal{M}$. The proposed algorithm builds two layers of the grid maps, one containing the occupancy grid map, reflecting the probability that a cell is occupied, and the other layer contains the amplitude information. The latter can be used to gain information about the RADAR cross section and thus about the reflectivity and material of the obstacles. Highly reflecting objects are usually metallic, while vegetation typically has low reflectivity.

**RADAR Occupancy Grid Maps** The occupancy grid mapping is common in SLAM [TBF05b]. It relies on the binary assumption that each grid cell can only be occupied (1) or free (0). Then the probability that all grid cells $m_i$ are in state $s_i$ is given by the posterior distribution

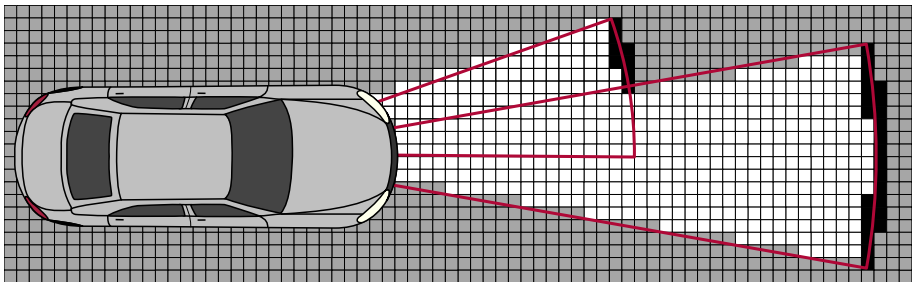
$$p(\mathcal{M}_t|z_{1:t}, x_{1:t}) = \prod_{i \in \mathcal{M}} p(m_i(t)|z_{1:t}, x_{1:t}). \tag{4.4}$$

Because the large product over all cells in the grid map is hardly numerically stable, [TBF05b] propose the log-odds representation of the individual grid cell given by

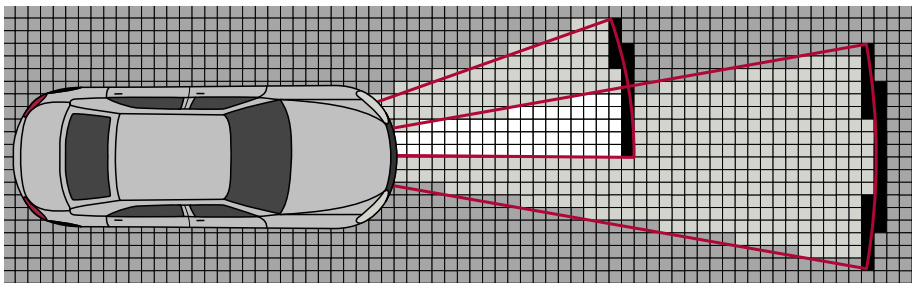
$$l_{t,i} = \log\left(\frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})}\right) \text{ with } p(m_i|z_{1:t}) = 1 - \frac{1}{1 + \exp\left(l_{t,i}\right)} \tag{4.5}$$

being the sensor model. It models the influence of the measurement to the grid cells and takes sensor specific effects into account.

In typical applications, such as LiDAR-based SLAM, the *beam model* is used. It assumes that each measured target has a high probability of occupation and the space in between the sensor and the target is free. This assumption is usually correct for LiDAR and camera based approaches, because the sensors require line of



(a) Grid map based measurement model commonly used for LiDAR sensors, where the space between the detection and the sensor is marked as free (white).



(b) Adaptation proposed by [DWR+15]. For RADAR sensors, free space is not certain between sensor and target (light grey) cells. If multiple overlapping scans confirm free space, it is marked as free (white cells in the center).

**Figure 4.1:** Sensor models showing the grid update based on the measurement uncertainty of different sensors.

sight to detect an object. Thus the resulting grid map looks like it was sketched in Figure 4.1 (a). However this assumption does not hold true for the RADAR sensor. As discussed previously, in some cases the RADAR sensor can see around obstacles due to multiple scattering. Thus, an adaptation was proposed by [DWR+15], as shown in Figure 4.1 (b). Fortunately, the sensor model in Section 4.1.1 reflects the RADAR characteristics. Furthermore, Werber et al. [DWR+15] propose additional plausibility factors, namely by a Plausibility Scaling (PS) factor and a Plausibility Offset (PO).

**Angle Plausibility** punishes the detections close to the edge of the sensor FOV by an angular plausibility scaling $\varphi_{\mathrm{PS}}$ and plausibility offset $\varphi_{\mathrm{PO}}$ as given by

$$p_{\varphi} = 1 - \frac{1}{(1 + \exp(-\varphi_{\mathrm{PS}}(|\varphi| + \varphi_{\mathrm{PO}})))}. \tag{4.6}$$

**Range Plausibility** favors detections close to the sensor, because due to the quadratic loss in amplitude, the signal to noise ratio for high distance targets is deteriorating. This will be compensated by a range plausibility scaling $r_{\mathrm{PS}}$ as given by

$$p_r = \exp(-r_{\mathrm{PS}} r^2). \tag{4.7}$$

**Amplitude Plausibility** favors highly reflecting targets to elevate good reflections from the noise floor with an amplitude plausibility scaling factor $A_{\mathrm{PS}}$ and a plausibility offset $A_{\mathrm{PO}}$ given by

$$p_A = 1 - \frac{1}{1 + \exp(-A_{\mathrm{PS}}(A + A_{\mathrm{PO}}))}. \tag{4.8}$$

The plausibility is then combined to the total plausibility $p = p_{\varphi} + p_r + p_A$. The free parameters were determined empirically by Werber et al. [DWR+15] and increase the robustness of the sensor model significantly.

As the vehicle drives through the parking lot and constantly updates the grid map, free space becomes more certain. If an obstacle has been scanned from multiple angles, the contours of the objects sharpen to provide a clear representation of the obstacle.

The algorithm was implemented based on the paper [DWR+15] for this thesis as shown in Algorithm 4 and provides the output given in Figure 4.2. For each grid cell $m_i$ in the FOV, the inverse sensor model is calculated based on the distribution described above. Then the plausibility correction are applied to correct for the radar specific behavior. The final output is an updated likelihood for $m_i$.

The variable $l_0$ in Algorithm 4 refers to the initial probability when no knowledge from any sensor has been processed. It is usually initialized with $l_0 = 0$, which corresponds to a probability of $p_0 = 0.5$.

**Algorithm 4** Radar Grid Mapping

1: **procedure** OCCUPANCY UPDATE
2: **Input:** Prior occupancy $l_{t-1,i}$, state and measurement $x_t, z_t$
3: **Output:** Posterior occupancy $l_{t,i}$
4: **for** $m_i \in \mathcal{M} \cap FOV(x_t)$ **do**
5: $l'_{t,i} = l_{t-1,i} + \textbf{inverse\_sensor\_model}(m_i, x_t, z_t) - l_0$
6: $l_{t,i} = \textbf{run\_plausibility\_model}(l'_{t,i})$
7: **return** $l_{t,i}$

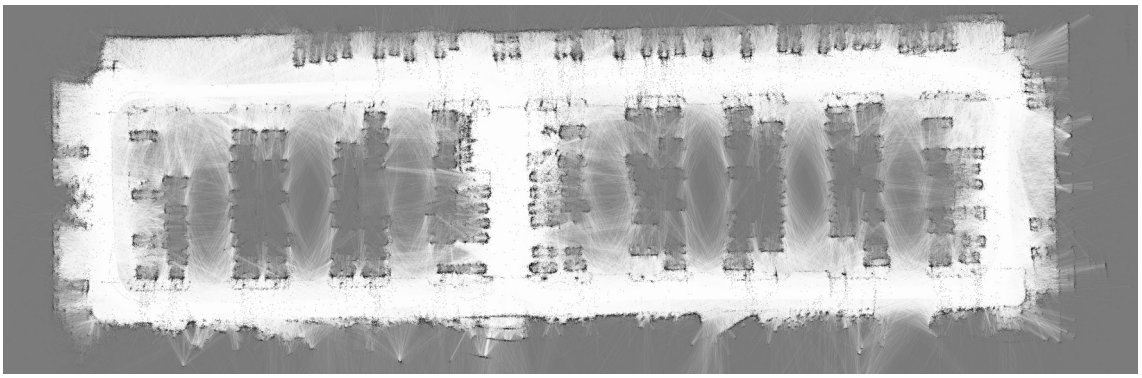


**Figure 4.2:** Resulting occupancy grid map.

### 4.1.3 Joint Occupancy Grid Maps

While the RADAR grid map approach was developed by [DWR+15] and serves as a foundation for the GraphSLAM framework described in the following chapters, in this thesis an improvement to the grid map approach to incorporate map updates has been developed to improve the feature extraction on occupancy grid maps that will be utilized in the following sections.

When cars are localizing on an occupancy grid map, they are bound to collect new or updated information about their surroundings, especially in a highly dynamic environment like a parking lot. Besides developing a robust localization approach that can handle discrepancies in sensor and map data, the updated information should end up in the map to prevent further deterioration of the map content. The grid cells can be merged by the following logic:

**If Previous and Current map agrees** the occupancy grid map increases, because the occupancy level of the grid cell was confirmed by two independent measurements. This scenario is met if the grid cell is measured as occupied $l_{\text{prev}}, l_{\text{current}} > 0$ or as free space $l_{\text{prev}}, l_{\text{current}} < 0$. Then

$$l_{\text{update}} = l_{\text{prev}} + l_{\text{current}}. \tag{4.9}$$

**If Previous and Current map disagrees** and the cell has changed from occupied $l_{\text{prev}} > 0$ to free space $l_{\text{current}} < 0$, then the information from the previous cell is disregarded

$$l_{\text{update}} = l_{\text{current}}. \tag{4.10}$$

**Unexplored regions** that remain unexplored after the update $l_{\text{prev}} = l_{\text{current}} = 0$ will not be updated

$$l_{\text{update}} = 0. \tag{4.11}$$

## Evaluation

The evaluation is performed on the realistic parking maneuvers, which were described in Section 3.3. The occupancy grid map and the corresponding update is visualized in Figure 4.3 indicating the update of an occupied parking space in the red box.

Accordingly the occupied cells were replaced by free space. While the update mechanism disregarding all previous information in Equation 4.10 seems too radical at first, it has proven necessary in the highly dynamic environment of the parking lot to incorporate the numerous changes that do not happen gradually but can rather happen within minutes. Thus this update scheme was chosen also to achieve a solid foundation for Section 4.3 and Chapter 5, where the RADAR occupancy grid map is utilized for a feature based GraphSLAM approach.

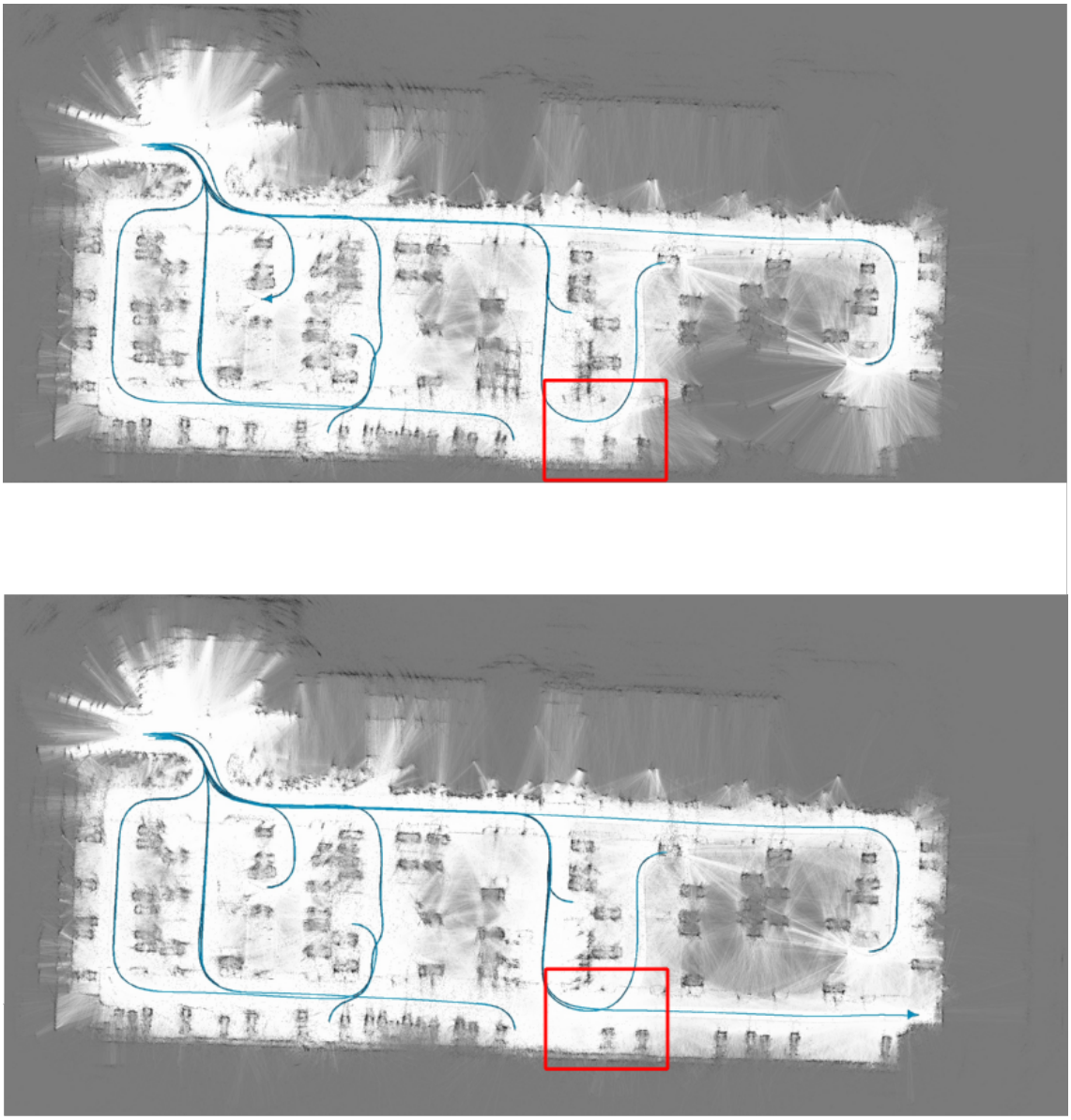


**Figure 4.3:** Update of the grid map on realistic parking maneuvers. The upper image (previous map) holds three parked cars in the red box, while in the lower image (current map) there are only two cars parked in the lower edge of the parking space.

## 4.2 Robust RADAR Cluster Map Representation

This section investigates the SLAM problem on RADAR data based on particle filters. While the previous approach focused on grid maps, which typically have high memory consumption and can only be updated with a very precise map matching. In this chapter, a novel approach is proposed. The RADAR sensor's physical properties are taken into account by applying a unique stream clustering based map representation.

The approach proposed in this section has several advantages over standard grid-based or feature-based maps. Besides having a noise reducing influence on the RADAR data, the map representation is especially suited for the highly dynamic parking lot scenario. It is highly adaptive to dynamic environments and can be used for exploration[1]. The resulting map is stored in an R-tree data structure to maintain maximum flexibility and speeding up access to the data. Our experimental results indicate that the presented algorithm named ClusterSLAM takes a step towards lifelong navigation in urban scenarios, as it outperforms the classical grid representation in numerous aspects, as discussed in the following sections.

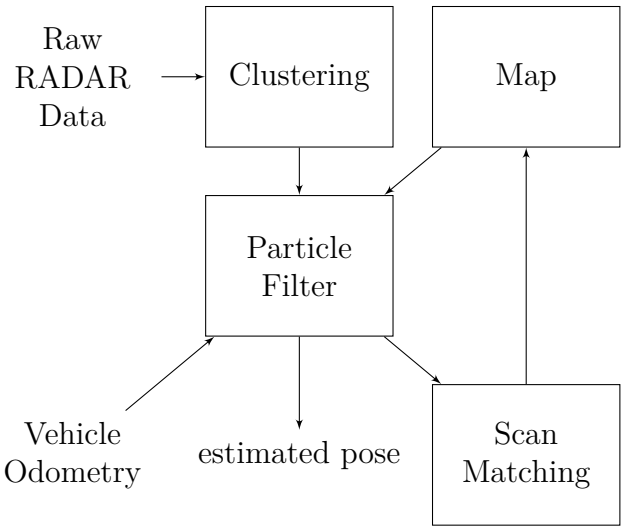



**Figure 4.4:** Overview of the ClusterSLAM localization system. The system inputs raw RADAR data, as well as the odometry and outputs an estimated pose.

[1]Exploration denotes adding previously unseen areas to the map.

ClusterSLAM is implemented as a particle filter, which is specifically designed to cope with RADAR data directly. By applying a density-based stream clustering algorithm to the incoming sensor readings, noisy measurements can be disregarded a priori and thus never reach the map.

The system architecture is displayed schematically in Figure 4.4. Raw data is clustered to form a scan of the environment. Using scan matching, this scan is compared and merged with a reference map to obtain the weights necessary for the particle filter utilized by ClusterSLAM. Each particle contains its own hypothesized version of the reference map, to which the current scan is compared.

### 4.2.1 Map Representation

The map representation is of particular importance in this approach, as it deals with three of the main sources of errors induced in the scenario at hand. Besides handling the high noise RADAR data, the test scenario is highly dynamic and thus requires a quick map update mechanism. Additionally, in the automotive industry, memory requirements are typically very strict. Thus, regular particle filter approaches, where each particle contains its own map, are impractical and costly to implement. Our approach thus uses a common map for all particles. The state space of each particle is only given by the pose and a particle weight.

Consider the RADAR data depicted in Figure 2.7. Despite the high noise floor, a human can easily discern certain shapes that correspond to physical objects, such as parked vehicles, rain gutters and curbs. A human is able to extract the essential information from the noisy data by paying attention to the *density* of the targets in certain locations and clusters forming specific shapes. The literature provides a variety of density-based clustering methods, among the most popular is Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [EKSX96]. In contrast to standard clustering algorithms such as $k$-means [KMN$^+$], density-based clustering algorithms do not require the number of clusters to be set beforehand and can be applied to arbitrary shapes. Originally these algorithms were developed in data mining applications for a large static data set with an unknown amount of search results. Due to the distributed nature of most large data sets, density-based clustering techniques have been applied to work on *data streams* rather than static data sets [ECQZ06], where the full information of the state space is not available at $t = 0$, but where more data is added over time. This data-driven approach can be applied to SLAM, as each new measurement update can be understood as a new sample of data being streamed into the data pool. The clustered data pool will represent the map.

There are different approaches to density-based stream clustering [AWS14]. The most suitable concept for the SLAM application is the micro cluster approach. It is utilized in the DenStream algorithm [ECQZ06]. As the RADAR data consists of two spatial coordinates and the amplitude ($r$, $\varphi$, $A$), as explained in Section 3.1, it

is clustered by the stream clustering algorithm in these three dimensions. Examples are depicted in Figure 4.5 and can be described by the following properties:

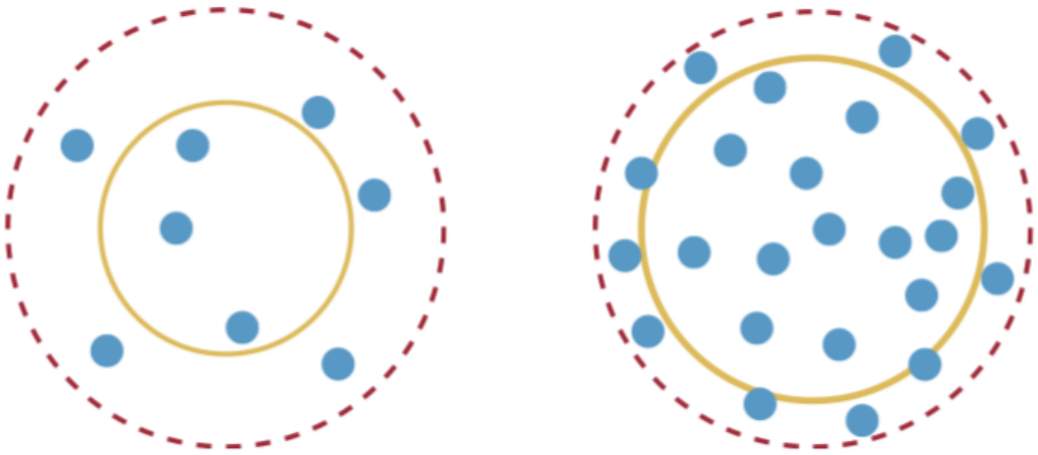
$$W_C = \sum_{i=1}^{n} w_i = \sum_{i=1}^{n} \prod_{j=1}^{m-1} (1 - \mathcal{N}(d_{ij}|C_j, R_j)), \tag{4.12}$$

$$C_i = \frac{1}{W_C} \sum_{i=1}^{n} c_i = \frac{1}{W_C} \sum_{i=1}^{n} p_i w_i, \tag{4.13}$$

$$C_{sq} = \frac{1}{W_C} \sum_{i=1}^{n} p_i^2 w_i, \tag{4.14}$$

$$R_i = \sqrt{C_{sq} - C^2}. \tag{4.15}$$

They describe the cluster weight $W_C$, the cluster center $C_i$, the center of mass $C_{sq}$ and the the cluster radius $R_i$, respectively. Hereby $i \in 0, ..., n$ iterates over the number of clusters and $j \in 0, ..., m$ iterates over the number of points of each cluster. The cluster weight is thus determined by the Gaussian probability of each member point being part of the cluster. The center of the cluster $C_i$ represents the center of mass of the cluster as determined by their members' weight. $C_{sq}$ represents the variance of points within the cluster and the cluster radius $R_i$ represents the one sigma range of the cluster. These parameters are generated in two update steps for each cluster, the cluster generation, measuring the center and radius and the cluster weight determination.



(a) O-Micro-Cluster ($W_C < W_{\text{thresh}}$). (b) P-Micro-Cluster ($W_C \geq W_{\text{thresh}}$).

**Figure 4.5:** Different types of micro clusters. (a) represents outlier micro clusters with insufficient density. (b) potential micro cluster. The RADAR detections are shown in blue, the standard deviation is shown in yellow. The dashed line indicates the maximum size of each cluster before it is separated into two.

**The Cluster Formation** takes place in each time step, when new RADAR data is processed. The center point $C$ is given by the center of mass of the RADAR targets according to their weights. The radius $R$ is defined as the standard deviation of the target distribution, whereas the squared weighted sum $C_{sq}$ is used for simplification. For each new incoming data point, these properties can be obtained iteratively. Thus it is not required to store all RADAR data for the measurement update and the map will only contain clusters described by these properties. After the update step, the clusters are ranked with respect to their weight $W_C$. It determines whether a cluster is ranked as a p-micro-cluster or as an outlier. In each measurement update, clusters can be promoted to p- or demoted into o-micro-clusters. Outliers are not considered for the weight calculation or for for the particle filter update. A new o-micro-cluster can be formed from a single RADAR point, if no other clusters are in its surroundings.

**The Cluster Weight** is computed each time a new RADAR target is processed. The weight calculation also encodes the RADAR-specific behavior of the algorithm. In the context of this work, the weight $w_i$ for all $n$ RADAR detections is calculated from the Gaussian probability $\mathcal{N}(d_{ij}|C_j, R_j))$ that a RADAR target is visible through the $m-1$ other clusters in the map, as illustrated in Figure 4.6. This means that clusters can form behind other clusters, even though they are obstructed from view. Due to the effects in Chapter 2, this is a feature unique to the RADAR sensor. To approximate this behavior, i.e. the multiple scattering and penetration depth, we assume that another cluster does not completely obstruct the line of sight, but that a cluster is only fully opaque at the center of a cluster where the Gaussian distribution is maximal. This is indicated in Figure 4.6.

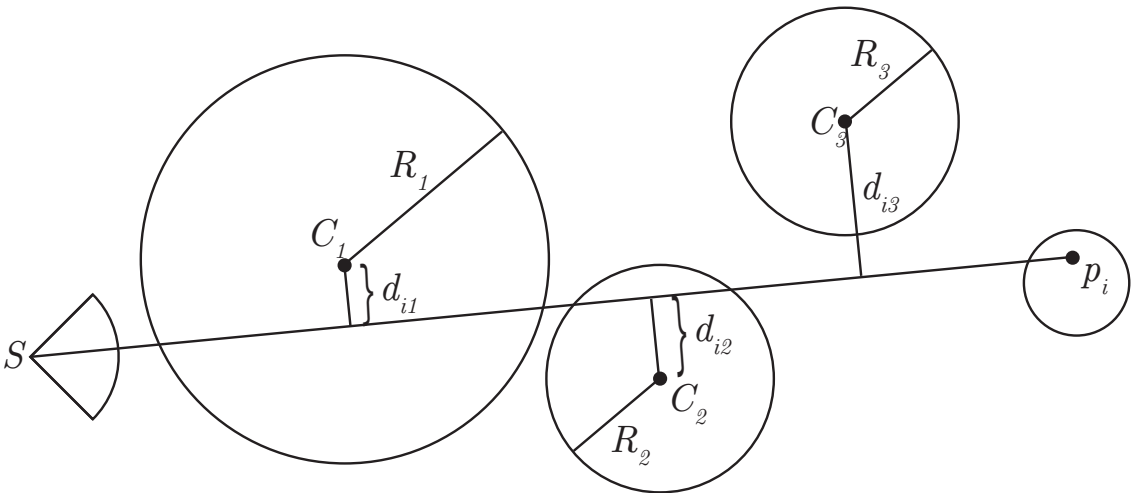



**Figure 4.6:** The cluster weight for a given point $p_i$ is calculated using the normal distance $d_{ij}$ of the center points $C_j$ from the line of sight originating from the sensor $S$ for each $p_i$. The radius $R_j$ indicates the $\sigma_j$ environment.

Hence, the total weight $W_C$ of the cluster is the sum of the individual containing target weights $w_i$. All clusters exceeding a threshold $W_C \geq W_{\text{thresh}}$ are called $p$-micro-clusters. In practice, we limit the weight by a parameter $W < W_{\max}$ to define a saturation limit.

These micro-clusters can be maintained incrementally allowing temporal accumulation of the RADAR targets [ECQZ06]. Any new measurement $x_{n+1}$ is added to the nearest cluster according to the Mahalanobis distance,

$$d(x, C) = \sqrt{(C - x)^T \Sigma^{-1} (C - x)}, \tag{4.16}$$

where $x$ represents the location of a new RADAR target, $C$ the center of the cluster and $\Sigma$ the covariance matrix of the measurement of $x$. If the target is within the maximum radius $r_{\max}$ of the cluster, a measurement is added to the cluster and its properties $(C_j, R)$ are updated. Note that adding another data point to a cluster can be done iteratively, such that only a single pass is necessary.

Since the RADAR sensor described in Section 3.1 only yields 64 points per sensor, significant cluster formation cannot be calculated from one sample of data. In order to have enough radar points available, the RADAR data is aggregated for $t = 0.5\,\text{s}$ before a new measurement update is performed.

Thus, the map consists of a set of clusters $\mathcal{M}_p$ containing the accumulated RADAR targets with properties (Equation 4.12−4.15). Unlike in grid structures, clusters can be placed arbitrarily in 3D space containing the spatial coordinates $x$, $y$ and amplitude $A$, which can easily be associated to the polar coordinates of the RADAR sensor and vice-versa by a simple coordinate transformation. Individual clusters are stored in an R-tree data structure for the purpose of efficient spatial search, as well as fast insertion and deletion.

In Figure 4.7, a cluster map is displayed in various levels of detail. Let the reader be reminded that only p-micro-clusters are kept in the final map, thus leaving regions of low measurement density free of clusters. In image Figure 4.7 (c), one can see the outline of four parked cars grouped in pairs leaving a space between them in the cluster representation. A common problem when dealing with measurements from automotive RADAR sensors are reflections caused by metal surfaces. The single cluster (light blue box labeled II.) is created in a strong reflection point from one of the cars. Similarly, in Figure 4.7 (b), one can see a few clusters (light blue box labeled I.) blocking the trajectory of the car. This is caused by reflections emitted from the aforementioned rain gutter embedded in the road's surface.

To further increase the robustness of the map and to handle dynamic environments, each micro-cluster decays with time. The cluster weight $W_C$ is adjusted such that the cluster has to be confirmed constantly by measurements (if the cluster is in the FOV of the sensors). If it is not measured any more, the cluster decays into

an o-micro cluster, which is disregarded in the measurement update of the SLAM algorithm. The adjusted cluster weight thus follows

$$W_{C,\text{decay}} = \sum_{i=1}^{n} \prod_{j=1}^{m-1} 1 - \mathcal{N}(d_{ij}|C_j, R_j) - \beta w_m. \tag{4.17}$$

The free parameter $\beta$ is a decay constant and $w_m$ is the weight of a unit cluster, having the weight of one RADAR target with zero occlusion. Thus, each cluster has to be observed frequently in order to remain a p-micro-cluster. If the weight

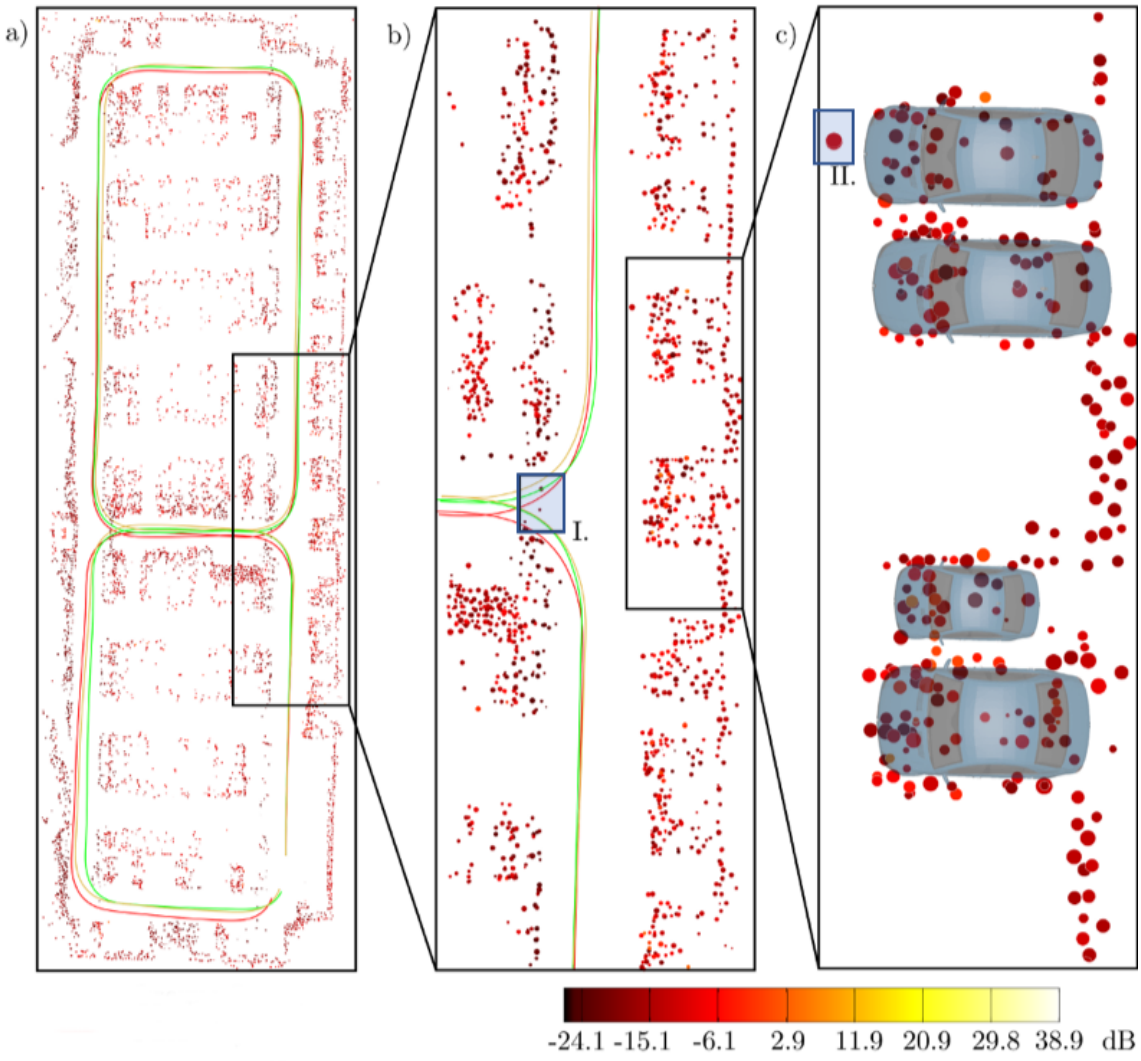



**Figure 4.7:** Map representation using $p$-micro-clusters in three levels of detail. As before, the applied color scheme represents the mean amplitude of the clusters. The ground truth trajectory is shown in green, odometry in red and the best pose estimate in orange. The cars parked in the parking lot environment are schematically overlayed in blue.

drops below the above mentioned threshold $W_{\text{thresh}}$, it is degraded to an outlier and disregarded for the pose estimation in the particle filter. On the other hand, if a cluster has not been observed recently but is present once again, an outlier will be promoted to p-micro-cluster within a few observations. The threshold $W_{\text{thresh}}$ is found by determining the average number of sightings of a highly reflective object.

The reader may notice that the disturbing factors (i.e. cars, pedestrians, reflections and noise) are usually non-stationary, thus they can be suppressed by a decay on the individual clusters. Therefore, only the stationary objects will remain in the map. Reflections are usually non-stationary if the vehicle is moving, as reflecting objects are measured from different viewing angles. Hence, when a car moves past the reflecting object, the decay eliminates the clusters produced by reflection. For the same reason, the decay helps eliminate moving objects while mapping, because they are not detected in the same location every time. Lastly, the decay is used to update the map due to environmental changes, such as cars not being parked in the same location as before. Even though such clusters usually have a large weight, they will be eliminated eventually, in case they are not confirmed by new measurements.

### 4.2.2 Particle Filter

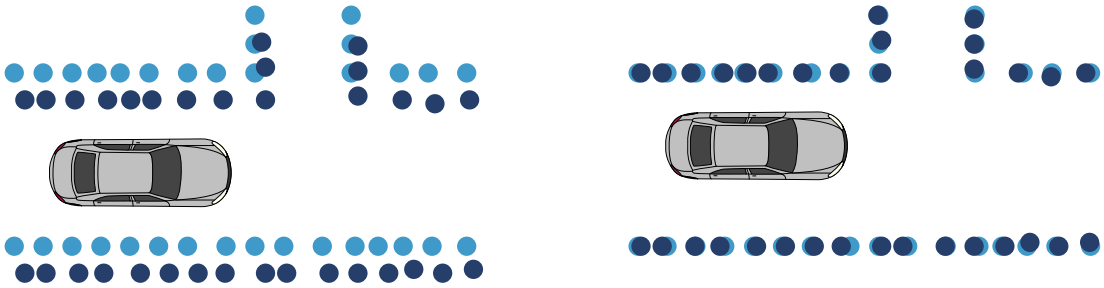
The particle filter used in ClusterSLAM to determine the posterior distribution closely resembles the algorithm explained in Algorithm 1. For the motion update, the odometry information from the vehicle is used, while the measurement model is based on the clustering algorithm described in the previous section.

**The motion update** is calculated based on the Ackermann steering geometry described in Section 2.1.1 and displayed in Algorithm 5. The velocity $v$ and yaw rate $\psi_v$ are extracted from the CAN and interpreted in a single track model. We assume that within two consecutive time steps $\Delta t = t_2 - t_1$, the car translates by $\Delta x = v\Delta t$ and is rotated by $\Delta\theta$. It is assumed that $v$ is constant within $\Delta t$. To update the particle filter probabilistically, the equations of Section 2.1.1 need to be sampled. As proposed in [TBF05b], a suitable sampling algorithm is to assume a normal distribution for $\Delta x$ to approximate the error introduced by the constant velocity assumption. It introduces four free parameters $\alpha_i$ to model the offset measured in translational and rotational direction. They were estimated for the specific parking lot scenario, where low velocities and tight turning radii introduce the most drift into the odometry model. They were estimated in a simple gradient descent to determine the least odometry drift on an eight shaped dataset after a full double loop closure.

**The measurement update** is performed using scan matching on clustered RADAR data to determine the posterior particle distribution and the particle weight. In this case, a scan from current measurements $\mathcal{S}$ containing $n_{\mathcal{S}}$ clusters are supposed to be best aligned with the already matched map data $\mathcal{M}$, containing $n_{\mathcal{M}}$ RADAR clusters.

**Algorithm 5** Odometry Motion Model

**procedure** MOTION UPDATE
  **Input:** Prior estimated odometry pose $\mathbf{x}_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})$, control $u_t$
  **Output:** Sample from the proposal distribution $\mathbf{x}_t = (x_t, y_t, \theta_t)$
  $\delta_{\text{rot1}} = \text{atan2}(\text{y}_\text{t} - \text{y}_{\text{t}-1}, \text{x}_\text{t} - \text{x}_{\text{t}-1})$
  $\delta_{\text{trans}} = \sqrt{(y_t - y_{t-1})^2 + (x_t - x_{t-1})^2}$
  $\delta_{\text{rot2}} = \theta_t - \theta_{t-1} - \delta_{\text{rot1}}$
  $\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} + \textbf{sample\_normal\_distribution}(\alpha_1\delta^2_{\text{rot1}} + \alpha_2\delta^2_{\text{trans}})$
  $\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} + \textbf{sample\_normal\_distribution}(\alpha_3\delta^2_{\text{trans}} + \alpha_4\delta^2_{\text{rot1}} + \alpha_4\delta^2_{\text{rot2}})$
  $\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} + \textbf{sample\_normal\_distribution}(\alpha_1\delta^2_{\text{rot1}} + \alpha_2\delta^2_{\text{trans}})$
  $x_t = x_{t-1} + \hat{\delta}_{\text{trans}} \cos(\theta_{t-1} + \hat{\delta}_{\text{rot1}})$
  $y_t = y_{t-1} + \hat{\delta}_{\text{trans}} \sin(\theta_{t-1} + \hat{\delta}_{\text{rot1}})$
  $\theta_t = \theta_{t-1} + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$
**return** $\mathbf{x}_t$



(a) Poor match (b) Good match

**Figure 4.8:** Illustration of scan matching. The scan $\mathcal{S}$ (dark blue) is overlaid onto the map $\mathcal{M}$. The vehicle pose is estimated based on the matching of $\mathcal{S}$ to $\mathcal{M}$.

The objective is then to find the best match between $\mathcal{M}$ and $\mathcal{S}$, such that the Euclidean distance between all matched points is minimized. Figure 4.8 illustrates this. There are a number of methods to determine and minimize the distance between scans. Among the most prominent is the Iterative Closest Point (ICP) algorithm [BM92]. However, since ICP does not model uncertainties, a more advanced method is needed, which is based on NDT [SML12]. The NDT **??** represents the scan and map as a mixture of Gaussians and use the $L_2$ norm to determine the distance between the sets $\mathcal{S}, \mathcal{M}$ represented as

$$D(\mathcal{S}, \mathcal{M}, \Theta) = \sum_{s \in \mathcal{S}} \sum_{m \in \mathcal{M}} \mathcal{N}(0 | T_{\mathcal{M}}(\mu_s, \Theta) - \mu_m, T_{\mathcal{M}}(\Sigma_s, \Theta) - \Sigma_m), \tag{4.18}$$

where $T_{\mathcal{M}}(\cdot, \Theta)$ denotes the transformation from the sensor coordinate system to the map coordinate system given the set of parameters $\Theta$. To evaluate the sum in a SLAM approach, the equation can be simplified to

$$d(\mathcal{S}, \mathcal{M}) = d_1 \sum_{i=1}^{n_{\mathcal{S}}} \sum_{j=1}^{n_{\mathcal{M}}} \exp\left( -\frac{d_2}{2} \mu_{ij}^T (R^T \Sigma_i R + \Sigma_j)^{-1} \mu_{ij} \right), \tag{4.19}$$

with $\mu_{ij} = R\mu_i - \mu_j + t$ and $R, t$ the rotation and translation component of $T_{\mathcal{M}}$, respectively. The constants $d_1$ and $d_2$ are constants from the Gaussian distribution.

Thus, if each particle $p$ holds its own map $\mathcal{M}_p$ and processes the same set of clusters $\mathcal{S}_t$ at any given time, we can determine the particle weight based on the scan matching distance given by Equation 4.19. Each micro-cluster $s_i \in \mathcal{S}_t$ is assigned to its nearest neighbor $m_j \in \mathcal{M}_i$ in terms of the Mahalanobis distance Equation 4.19 as described above. From the distance function, the particle weight for each of the $n$ particles in a set $\mathcal{P}$ is calculated with the following approach

$$W_i = \frac{\max_{i \in \mathcal{P}}(d(\mathcal{S}_t, \mathcal{M}_i)) - d(\mathcal{S}_t, \mathcal{M}_i)}{n \max_{i \in \mathcal{P}}(d(\mathcal{S}_t, \mathcal{M}_i)) - \sum_{i \in \mathcal{P}} d(\mathcal{S}_t, \mathcal{M}_i)}. \tag{4.20}$$

**Algorithm 6** Low Variance Resampling

```
1: procedure RESAMPLING
2:     Input: Particle set X_t, weights w_i ∈ W_p, number of particles M
3:     Output: Resampled particle set X̄_t
4:     X̄_t = 0
5:     r = rand(0, M^-1)
6:     c = w_1
7:     i = 1
8:     for m = 1 to M do
9:         U = r + (m − 1)M^-1
10:        while U > c do
11:            i = i + 1
12:            c = c + w_i
13:        add x_i to X̄_t
14: return X̄_t
```

**Resampling** of the particle filter yields the desired particle distribution. In total, this procedure is called the low-variance resampling algorithm [TBF05b] and is described in Algorithm 6. The procedure is visualized in Figure 4.9 and is based on a random number $r$ being chosen at the start of the resampling process. From this number, a set of equidistant samples is chosen. Thus particles with a high weight will be drawn multiple times, while particles with low weight have a high chance of being skipped. Without the particle resampling, a lot of particles would end up in places of low probability, so they have to be contracted to regions with high importance weight again. However particle resampling also introduces errors, if a wrong patch of sensor data causes a faulty measurement update. Thus resampling has to be performed with care.

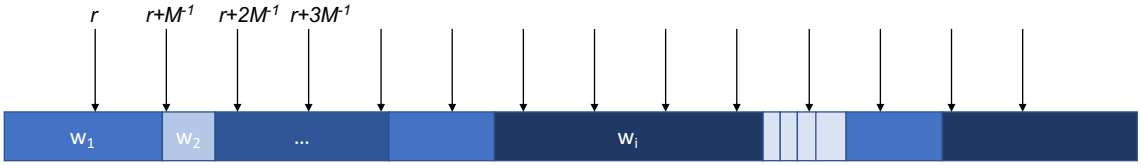



**Figure 4.9:** Resampling of differently weighed particles. A random number $r$ is selected once. Then $M$ particles are equidistantly sampled from the particles. The color and length of the sticks indicates the weight. Darker colors and larger segments correspond to higher weights. Thus it is ensured, that higher weighted particles get redrawn more often. Based on [TBF05b].

This standard approach described in Algorithm 6 is initialized in lines 1–4 by drawing a random number $r$ from the interval $[0, M^{-1}]$ with $M$ being the number of desired samples. From the random seed $r$ a sample for each of the $M$ particles is drawn ($U$). Then particles are selected based on their importance weight $w_i$ for the resampled particle set $\bar{\mathcal{X}}_t$. Thereby particles with large importance weight will be selected proportionately more often than particle weights with small $w_i$.

**The initial conditions** are based on the lost robot problem. With no prior information, the particles would technically need to be scattered across the entire map in a uniform distribution. In this case, we assume that we know the position of the robot within the accuracy of a standard vehicle GNSS system. Hence, we scatter the initial pose of the vehicle over an area of 16 m and 90° in heading around the origin of the map, as this describes roughly the worst case of a low-cost GPS localization as can be typically found in the production vehicles. This is merely done to reduce the initial size of the state space such that the particles converge quickly to the true pose.

### 4.2.3 Evaluation of ClusterSLAM

Two main situations have to be considered independently to evaluate the presented algorithm. The initial mapping process with the vehicle in exploration mode is discussed. Second, the localization on the prerecorded map and the map update is discussed in detail.

The results have been produced on a subset of about 32 sequences of data collected on parking drives from three different days from the *eight shaped* dataset introduced in Section 3.3 because at the time of publication [SWK+16], only a subset of data was available for processing. To remain consistent, the results are shown as published.

**The Initial Mapping Process** is completely relying on the sensor readings and tries to construct a map from the given sensor data. The particle filter injects new sensor data into a map from previous measurements. The corresponding initial mapping thus only compares new sensor data to prior information and has no static map in the background. In the autonomous parking scenario, this is used to perform the initial manual recording of the parking lot. Further drives can then rely on the map for pose estimation. Both are being presented in the following section.

Therefore, it is essential that the particle filter produces a consistent map. Most importantly, loop closures have to be detected and the error at the end of the trajectory has to be minimal. In Figure 4.10, the pose estimate error of the SLAM algorithm is compared to the odometry error for a fully occupied parking lot. The detection of the loop closure at the end of the trajectory results in a low error of about 1 m at the end point. For a sequence recorded at 16:00 o'clock, the full

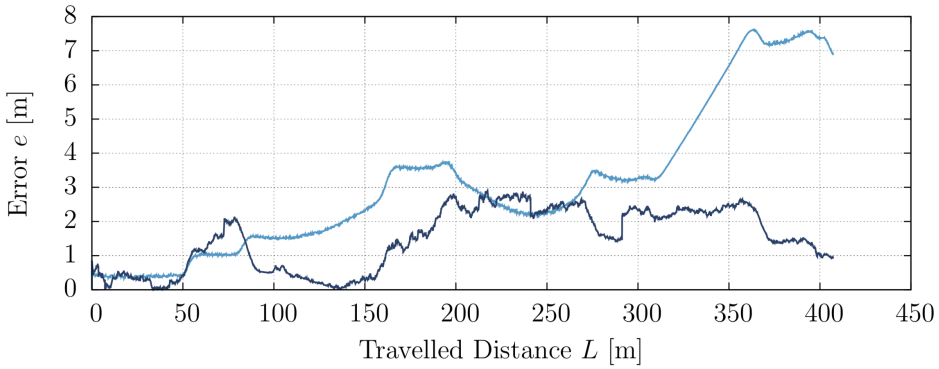



**Figure 4.10:** Deviation from ground truth of the pose error during the map creation phase of the best particle[2](dark blue) and the odometry (light blue) at 16:00 when the parking lot is highly occupied. The loop closure has been detected with an error of about 1 m at the end point.

trajectory has been displayed in Figure 4.7. In Figure 4.10 an optimized trajectory from the particle filter is shown. The particle trajectory is very close to the ground truth in most cases. One of the most important criteria for map quality is whether loop closes can be detected. Since the trajectory is closed, we compare the end point error for mapping runs at each time of the day in Figure 4.11. The main thing to note is that the error is basically independent of data collection time and thus of parking configuration. At 9:00 the parking space was usually only partially filled,

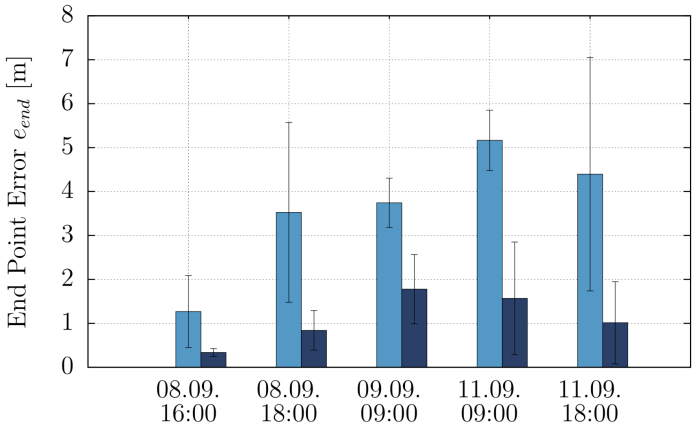



**Figure 4.11:** The mean error of the mapping process for all sequences in the data set sorted by time of the day, which is an indicator for parking lot occupancy. The error bars indicate a $1\sigma$ environment over all sequences, comparing the odometry (light blue) to the ClusterSLAM pose estimation (dark blue).

while at 16:00 almost every parking lot was occupied. By 18:00, almost all cars had disappeared.

**The Pose Estimation and Map Update** is performed as soon as a map exists. Then we apply the online localization and decay based map update described above to model environmental changes in the measurement update. The map update is graphically shown in Figure 4.12 and analyzed quantitatively in Figure 4.13, which shows that despite changing environments, the localization of ClusterSLAM yields consistent results. Even though at 18:00, the parking space was almost empty. For the localization within the map, we assumed that the starting position is known with standard vehicle GPS accuracy of about 16 m in radius and 20° in heading. After the initial convergence phase of the particle filter (about 50 m), it yields an accuracy less than 2 m despite highly dynamic environments while performing online map updates. In Figure 4.12, the map update using the decay mechanism is depicted. Comparing the clusters that refer to parked cars boxed in black, the map update while passing the objects can be seen. Each cluster that is not measured any more is demoted to an outlier and not used for the measurement update, but is retained in the map. Thus, as soon as the car passes the same location in a full parking lot again, within very few scans, the clusters can be reinstated and will positively affect the localization result. For this reason the localization performance is consistent throughout massive changes in the surroundings.

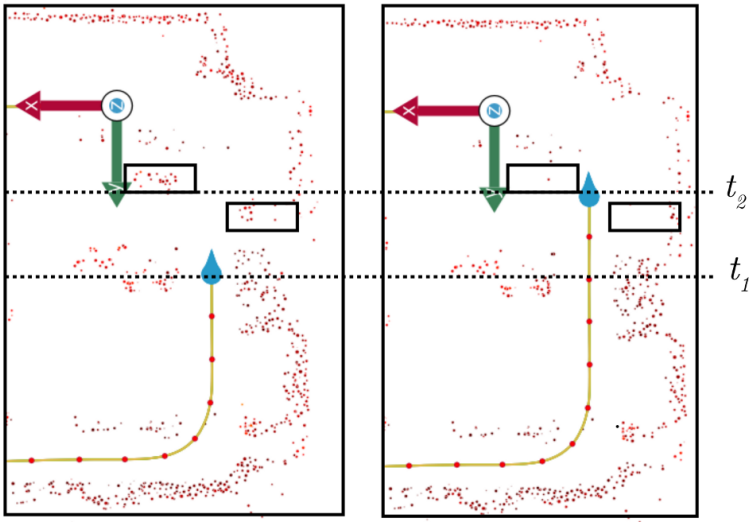



**Figure 4.12:** The vehicle (blue drop shape) passes a region that changed since the initial mapping. Between the time steps $t_1$ and $t_2$, the map update removes the two indicated cars (black boxes), while preserving the immobile structures.

[2]The best particle is described as the particle with the highest average weight across the entire trajectory. It thus has the most consistent map to the sensor data.

A localization step adds new clusters to the map, while others decay, resulting in a map size of approx. 200 kB for a sequence from the given data set. This is several orders of magnitude smaller than grid-based map representations, which typically amount to several MB for the same area. The memory and computing requirements thus allow this algorithm to be real time capable.

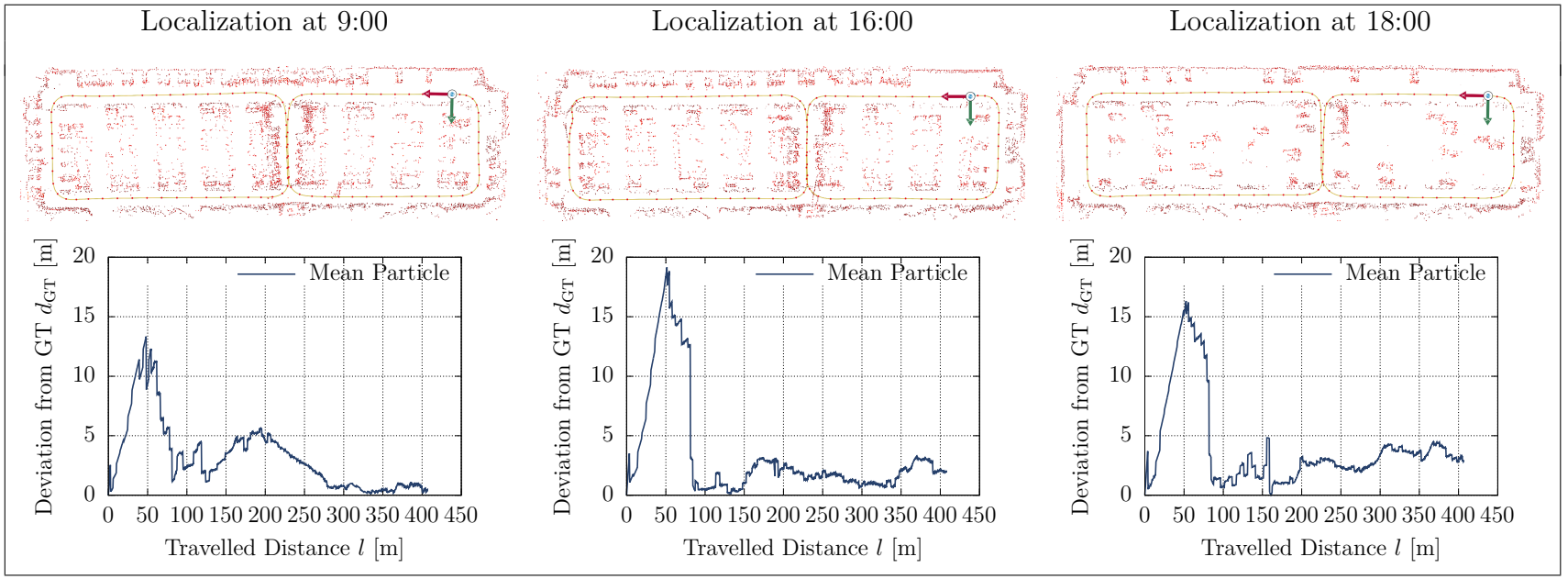



**Figure 4.13:** Localization results after initial mapping at 16:00. The respective vehicle configurations are illustrated as ground truth maps at the top. Each localization error with respect to the traveled distance compared to the odometry result is shown directly below.

### 4.2.4 Conclusion

In this chapter, a SLAM technique using automotive RADAR targets and odometric sensors, developing a map from a stream clustering algorithm. The map representation is especially suited for dynamic environments, which was validated using an extensive data set of a public parking space with different configurations of parked cars. Environmental changes are managed by cluster decay, which adds robustness to the system. We have shown that the presented map is suitable for localization using ClusterSLAM. To suit the cluster map, a specially designed weight function has been conceived to produce an accurate localization result despite the noisy and inaccurate sensor data. The map representation combines the advantages of feature and grid based algorithms alike, containing a high information density, while consuming little memory. Thus, our approach constitutes a significant step towards lifelong mapping in unexplored scenarios.

## 4.3 Graph Based RADAR SLAM

In this section, an alternative solution to the RADAR based SLAM problem is proposed. While the ClusterSLAM approach is providing a solid map update process and decent localization accuracy, a GraphSLAM based approach can outperform the ClusterSLAM approach in any category.

The localization can be improved compared to ClusterSLAM by generating landmarks from the raw RADAR signal [RGH+13] and constructing a landmark map with a GraphSLAM approach [TM05]. In contrast to clusters, the landmarks can be distinguished from each other using a feature descriptor. This severely improves the map matching performance and enables a high accuracy with a deterministic graph based optimization. It can be shown that high accuracy in both mapping and localization can be achieved, while keeping computation complexity and memory usage to a minimum. The map is optimized and is stored in an R-tree [Gut84] data structure to maintain maximum flexibility, while keeping lookup times low.

Due to the simplicity and compactness of a landmark map, this approach is not only limited to the parking scenario. It can easily be scaled in a cloud based application to perform at large scale in order to maintain the map across multiple vehicles. Multiple cars can contribute to mapping a large scale environment, such as a multi layer parking garage or even a country's highway network. This is further discussed in Section 4.4. The parking lot scenario, however, is among the most challenging ones due to the different vehicle configurations, sensor occlusion in tight spaces and moving objects.

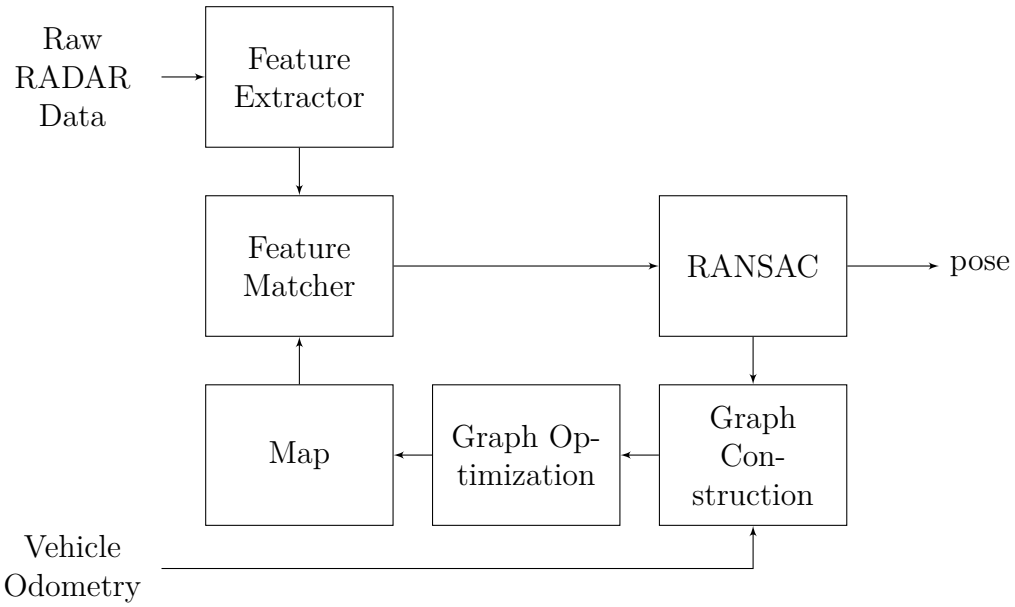



**Figure 4.14:** Overview of the GraphSLAM localization system.

The localization system is displayed in Figure 4.14. The raw RADAR data is first passed through the feature detector to extract the features later used in the GraphSLAM algorithm. The detected features are matched to the according map data and outliers are detected through a RANdom SAmple Consensus (RANSAC) algorithm drawing a subset of points from the matches to estimate the most likely transformation between the current scan and the already stored map. The good matches are assembled to a graph considering the vehicle odometry information to form a graph, which is then optimized in the Graph Optimizer. The resulting map is stored and used for further incoming measurements.

Aside from determining the inliers of the found matches, the RANSAC algorithm also yields a transformation between the feature points and map points, effectively localizing the vehicle in the map according to the detected landmarks. The following sections describe the functionality of the individual building blocks of the system.

### 4.3.1 RADAR Based Landmarks

The selection of appropriate robust features is essential for landmark based localization algorithms and requires special adaptations in the case of the RADAR sensor. Landmarks are generated in two basic steps, the feature *detection* and the feature *description*. While various methods are abundant in the image processing community, this thesis implements both these steps based on [RDH+16b], which was developed specifically for the RADAR in collaboration with Rapp et. al. For LiDAR sensors, a method for feature extraction and description has been suggested in [TA10b]. It uses a hierarchical clustering approach to detect individual features in dense LiDAR data. However, this approach is not applicable to RADAR measurements because the point density of 64 points per scan does not yield a high enough data density for the hierarchical clustering.

The feature detector finds relevant points in the RADAR data and thus yields the position of potential landmarks. In the RADAR signal domain, prominent points are generally located around scatter centers, which are specific points with high reflectivity and thus a low SNR. Furthermore, a scatter center can be tracked very well, as the RADAR sensor continuously receives points from scatter centers. In practice, scatter centers of RADAR data are typically e.g. metallic pole like objects or metal fences. This is beneficial for the dynamic environments of the parking lot, because the probability of finding the same object again after some time is significantly higher than for vegetation.

The descriptor on the other hand analyzes the environment around a landmark to generate a unique signature that corresponds to the particular landmark shape. The signature must be sufficient to solve the feature association problem, since similarity of descriptors corresponds to a similar surrounding of landmarks. In the described approach it is thus necessary to aggregate the RADAR data over time to extract the features. This can e.g. be achieved by constructing a grid map as presented in Section 4.1 to detect the scatter centers on a more dense representation of the data.

**The Fast Scatter Center Detection (FSCD)** algorithm Algorithm 7 was specially developed to incorporate the RADAR-specific physical behavior into a feature detector. It operates on the RADAR grid map approach presented in Section 4.1 and considers point like scatter centers, because those typically refer to human made structures like lamp post or signs. Additionally, in Figure 2.7 multiple reflections from vehicle bumpers typically form an ellipse rather than a circle and would thus be filtered out. Similar to the Features from Accelerated Segment Test (FAST) detector, FSCD applies a radial scheme to generate rotation invariant features, which allows for physical objects to be seen from different directions and still yield the same detection, which adds more robustness in complex scenarios.

**Algorithm 7** FSCD algorithm on an occupancy grid map

**procedure** FSCD
  **Input:** Occupancy grid map $\mathcal{M}$
  **Output:** Feature detections $\mathcal{P}$
  $\mathcal{P} = \emptyset$
  **for all** cells $c$ **do**
    **if** $\mathcal{M}(c) > \kappa_{\text{FSCD}}$ **then**
      check = true
      **for** scheme index $j = 1$ ... 16 **do**
        **if** $\mathcal{M}(c) \leq \mathcal{M}(c_j)$ **then**
          check = false
          **break**
      **if** check **then**
        $\mathcal{P}' = c \cup \mathcal{P}'$
  $\mathcal{P} =$ **FSCD_clustering**$(\mathcal{P})$
  **return** $\mathcal{P}$

**procedure FSCD_clustering**$(\mathcal{P})$
  $\mathcal{P}' = \emptyset$
  **for all** $p \in \mathcal{P}$ **do**
    $\mathcal{A} = \{p' \in \mathcal{P} : |p - p'| < \eta_{\text{FSCD}}\}$
    remove $\mathcal{A}$ from $\mathcal{P}$
    $p' = 1/n_{\mathcal{A}} \sum_{a \in \mathcal{A}} a$
    $\mathcal{P}' = p' \cup \mathcal{P}'$
  **return** $\mathcal{P}'$

The FSCD algorithm is described in Algorithm 7. Consider an occupancy grid map $\mathcal{M}$ generated like described in Section 4.1. Let $c$ be the cell index that is being tested for the existence of a scatter center and $\mathcal{M}(c) \geq \kappa_{\text{FSCD}}$ the occupancy of the tested cell. Then $c$ is a scatter center as defined by Algorithm 7, if

$$\mathcal{M}(c) > \mathcal{M}(c_j), \forall j = 1...16. \tag{4.21}$$

In other words: The grid cell $c$ is a scatter center if the occupancy peaks above a threshold $\kappa_{\text{FSCD}}$ and if there is no other local maximum within a radius of two grid cells around the detection. A visualization of the surrounding cells $c_j$ is given in Figure 4.15. The parameter $\kappa_{\text{FSCD}}$ ensures the stability of the detection as it is chosen such that a high number of detections have to be measured in $\mathcal{M}(c)$ in order for it to be a scatter center.

In Algorithm 7 each cell is tested for the condition Equation 4.21. Because a maximum of 16 compare operations are executed, the detector has linear complexity with regards to the number of cells in the map $\mathcal{M}$.

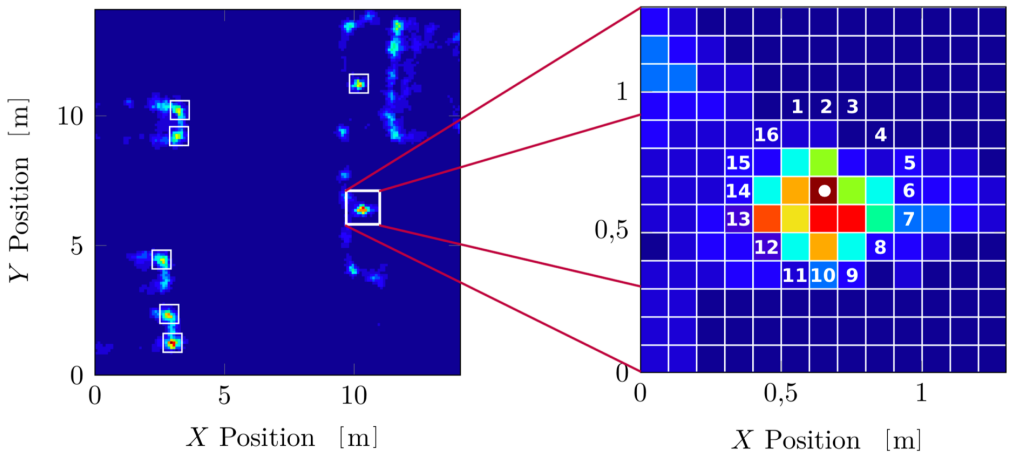



**Figure 4.15:** The radial scheme utilized in FSCD. The features are marked in white (boxes in left image) and the relevant radial points are numbered from 1 to 16 (right) [RDH+16b].

If each detection fulfilling the condition of Equation 4.21 would be returned as a landmark, there would be a multitude of detections on a single physical object. This is due to the discretization error of the grid map. In Figure 4.15, at least four grid cells fulfill Equation 4.21, but they all correspond to a single lamp post. To compensate for the multiple associations, a cluster based pruning of the landmarks is performed removing all double landmarks in a radius of $\eta_{\text{FSCD}}$ and correcting the position of the landmark to the cluster center. This behavior is displayed in the **FSCD_clustering** procedure of Algorithm 7.

**Description** For a landmark based SLAM approach, each landmark that is only described by a position in space, is not sufficient to solve the feature association problem, especially between the map and the current scan, because the coordinate systems are different and the transformation is unknown. Thus each feature is augmented using a *descriptor*. A descriptor encodes characteristic information about the detected feature $p$ in a feature vector $f_p$. The pair $(p, f_p)$ is called a *landmark*.

There are a number of description techniques for feature points in literature, typically in image processing settings, i.e. Fast Retina Keypoints (FREAK) [AOV12], Scale-Invariant Feature Transform (SIFT) [Lin12], Speeded Up Robust Features (SURF) [BTV06] or Oriented FAST and rotated BRIEF (ORB) [RRKB11]. Each feature detection utilizes similar methods to generate a maximally unique fingerprint of the surrounding pixels to obtain very recognizable landmarks with few mismatches.

In this thesis, the Binary Annular Statistics Descriptor (BASD) has been used [RDH+16b] to generate a feature vector for each detection. The BASD was specifically developed for map matching of RADAR grid maps and is fully rotation invariant [RDH+16b]. The method uses an annular partition of the surrounding[3] of the feature to provide expressive descriptors based on occupancy statistics of each circle. For each ring, the following information is encoded in the descriptor $f_p$:

$$\mu_i = \frac{1}{n_{\mathcal{C}_i}} \sum_{c \in \mathcal{C}_i \cap \mathcal{M}} \mathcal{M}(c) \tag{4.22}$$

$$\sigma_i = \sqrt{\sum_{c \in \mathcal{C}_i \cap \mathcal{M}} \mu_i - \mathcal{M}(c)} \tag{4.23}$$

$$med_i = \text{median}(\text{c} \in \mathcal{C}_\text{i} \cap \mathcal{M}) \tag{4.24}$$

$$min_i = \min(\text{c} \in \mathcal{C}_\text{i} \cap \mathcal{M}) \tag{4.25}$$

$$max_i = \max(\text{c} \in \mathcal{C}_\text{i} \cap \mathcal{M}). \tag{4.26}$$

These statistics are computed for each circle around the extracted feature point characterizing the surroundings of the point with statistical means. To compare features efficiently, the feature vectors are first binarized.

In image processing, binary descriptors are especially useful, because of two main advantages: On the one hand, the comparison of binary descriptors can be very efficiently realized using the *Hamming distance* [Mal10]

$$\Delta(x, y) = |\{i \in \{1, \ldots, n\} : x_i \neq y_i\}|. \tag{4.27}$$

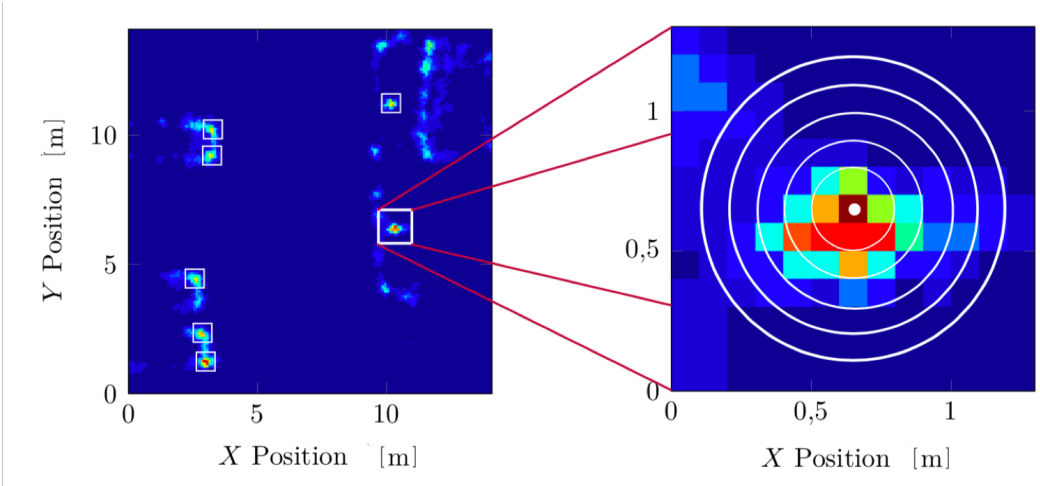



**Figure 4.16:** The same scenario as in Figure 4.15 is shown with the full descriptor rings (right). To binarize the feature vector, the sign of the difference between the statistical values of 4.22 to 4.26 are used [RDH+16b].

[3]The algorithm generates concentric circles around the detection

On the other hand, the descriptors do not take up much memory. Both advantages are especially relevant for large numbers of features or for real time applications, such as the automotive SLAM algorithms.

Thus in this thesis, the properties of Equations 4.22 to 4.26 are encoded into a binary vector by considering all rings $n_r$ surrounding a feature as indicated in Figure 4.16. In the $i$th slot of the descriptor vector the binary information is stored, whether the properties in the $i$th ring are smaller than the ones in the outer rings $j = i + 1...n_r$. The comparison yields the following number of binary entries $n_e$ in the feature vector:

$$n_e = 5 \sum_{i=1}^{n_r-1} i = \frac{5}{2}(n_r - 1)n_r. \tag{4.28}$$

This approach has proven to be very reliable and repeatable among many different RADAR grids, since it can be applied in grid registration [RGH+13]. The descriptors calculated by BASD in combination with the landmark detection [RDH+16b] is used to provide robust landmarks for the GraphSLAM approach. Beside the discussed descriptor described above, the FREAK [AOV12] descriptor, commonly known in image processing, is used as a reference implementation.

### 4.3.2 Graph Construction

The graph generation process consists of three different parts, which are displayed in Figure 4.14: feature association, outlier detection and graph assembly. While the feature association provides a set of landmark matches between the current vehicle detections and the map, the outlier detection identifies and eliminates the mismatches induced by noise and the dynamic environment. In the graph construction, the good matches are given a unique ID and are added to the graph as nodes.

**Feature Association** of the current set of landmarks $\mathcal{S}$ and the map $\mathcal{M}$ is obtained as described in Algorithm 8. It yields a set of matches $\mathcal{A}$ stored as a list of tuples $(s_i, m_i)$. Each tuple is called an *association*. Algorithm 8 iterates over all found landmarks in the current scan $\mathcal{S}$ and over all landmarks within a certain bounding box $\mathcal{B}$ to find all associations $a$ by calculating the Hamming distance, which counts all differing bits of the binary descriptors of $s$ and $m$. If the distance is below a certain threshold $d_{\text{thresh}}$, the association is added to $\mathcal{A}$.

Each observed landmark $s \in \mathcal{S}$ contains a relative position to the vehicle's rear axis and each landmark $m_i \in \mathcal{M}$ contains a position of the landmark in the map coordinate system. Both retain a binary descriptor $d$, which consists of $n_e = 512$ boolean elements. This number of elements has been proven effective detailing enough of the surroundings on a RADAR grid with $20 \times 20$ cm resolution. In practice, about sixty landmarks have to be compared to a local subsection $\mathcal{B}$ of the map,

**Algorithm 8** Binary feature matching

```
1: procedure MATCH
2:     Input: set of detected landmarks 𝒮, map ℳ
3:     Output: feature matches 𝒜
4:     𝒜 = ∅
5:     for all landmarks s in 𝒮, box ℬ do
6:         for all landmarks m in ℳ ∩ ℬ do
7:             if Δ(s, m) < d_thresh then
8:                 a = (s, m)
9:                 𝒜 = a ∪ 𝒜
10:    return 𝒜
```

which contains on average about 300 landmarks. Thus the matching can be done by a brute force matcher and still run in real time, as can be seen in the evaluation Section 4.3.3.

**Outlier Detection** is necessary, because the landmark detection and matching process are susceptible to outliers that can cause problems during graph optimization. Hence outliers are geometrically identified using a RANSAC based technique. The RANSAC algorithm follows the approach described in [RGH+14].

The desired variable is the rigid transformation $T = (R, t)$ consisting of the rotation and translation of the landmark locations in the vehicle frame to the map frame. If all matches are consistent, any random subset of three matches could be used to calculate $T$. When transforming all other landmarks in $\mathcal{S}$ to the map coordinate system, they should be perfectly aligned with their counterpart in $\mathcal{A}$. If there are mismatches, the probability that three random matches cause a suitable transformation decreases. The RANSAC algorithm utilizes this technique to determine the most likely transformation between vehicle frame and map frame, as well as a list of inlier matches that are represented by the transformation best. The algorithm is shown in Algorithm 9.

In each iteration $i$, a sample subset of three matches $\mathcal{A}_3$ are selected randomly. Based on this subset, the 2D rigid transformation parameters $T$ are determined. If matching features are within distance $d_{\text{thresh}}$ after transformation, it can be determined as an inlier. This is repeated $n_i$ times, while the transformation parameters which produced the most inliers is returned as the best fitting transformation. In practice, the number of RANSAC iterations $n_i$ varies with the percentage of outliers. In this thesis, $n_i = 100$ is chosen to ensure good transformation estimation in noisy RADAR data.

The estimation of the transformation parameters is performed using Singular Value Decomposition (SVD). According to the results of [SHR17], the problem of finding

**Algorithm 9** RANSAC

**procedure** RANSAC
  **Input:** set of matches $\mathcal{A}$, number of iterations $n_i$, threshold $d_{\text{thresh}}$
  **Output:** set of inliers $\mathcal{A}'$, transformation $T$
  $i = 0$
  $\mathcal{A}_{\text{inliers}} = \emptyset$
  **while** $i < n_i$ **do**
    $\mathcal{A}_3 =$ **select_random_subset**$(\mathcal{A}, 3)$
    $T =$ **compute_transformation**$(\mathcal{A}_3)$
    $\mathcal{A}^i_{\text{inliers}} =$ **compute_inliers**$(T, d_{\text{thresh}})$
    **if** $|\mathcal{A}_{\text{inliers}}| > |\mathcal{A}^i_{\text{inlier}}|$ **then**
      $\mathcal{A}_{\text{inlier}} = \mathcal{A}^i_{\text{inlier}}$
      $T_{\text{inliers}} = T$
    $i = i + 1$
  **return** $\mathcal{X}_{\text{inliers}}, T_{\text{inliers}}$

the best fitting transformation that aligns two sets of corresponding points can be written as a least square problem:

$$T = \arg\min_{R,t\in\mathbb{R}} \sum_{i=1}^{N} ||(Rp_i + t) - q_i||^2, \tag{4.29}$$

where $p_i$ and $q_i$ are associated points. An approach for minimizing the term was proposed by [AHB87] and is applied here for two dimensional problems. To obtain the optimal transformation, first consider the centroids of both sets:

$$\bar{p} = \frac{1}{n} \sum_{p\in\mathcal{S}_n} p_i \tag{4.30}$$

$$\bar{q} = \frac{1}{n} \sum_{p\in\mathcal{M}_n} q_i, \tag{4.31}$$

where $\mathcal{S}_n, \mathcal{M}_n$ are the $n$ selected landmarks from the current scan and from the map, as selected $\mathcal{A}_3$. We further define the deviations from the centroids as:

$$x_i := p_i - \bar{p} \tag{4.32}$$

$$y_i := q_i - \bar{q} \tag{4.33}$$

Let $X$ and $Y$ be matrices with columns $x_i$ and $y_i$, respectively with size $2 \times n$. Then we define the covariance matrix $S = XY^T$ of size $n^2$, with its singular value decomposition

$$SVD(S) = U\Sigma V^T, \tag{4.34}$$

where $U$ and $V$ are unitary matrices that contain the singular vectors of the covariance matrix $S$. Then the optimal rotation is given by

$$R_{\text{best}} = V \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & \det(VU^T) \end{pmatrix} U^T \tag{4.35}$$

and the optimal translation can be obtained by

$$t_{\text{best}} = \bar{q} - R\bar{p}. \tag{4.36}$$

Since the SVD method finds the best transformation analytically, the RANSAC algorithm can be performed very efficiently using this estimation function. The resulting transformation is then used to add the current scan to the map by assigning a unique identifier to each landmark in the map. Outliers and unmatched landmarks get new identifiers, while matched landmarks adopt the ID of their matched counterparts.

**The Graph Construction** is based on Section 4.3 and uses the unique identifiers for each landmark and the estimated vehicle poses from wheel odometry that outputs an updated pose estimation about five times as often as new landmark data comes in. Each odometry measurement and landmark observation is added to the graph as vertex, whereas the observation of measurement accuracy as inferred from the covariance matrix from each pose is stored in the graph's edges, both for odometry and landmark measurements. The finalized graph is partially depicted in Figure 4.17.

Since the initial graph is only constructed using odometry, the map contains a large amount of drift error (Figure 4.17 top left). To remove drifts and measurement errors, a graph optimization is performed. This allows the system to solve the full SLAM problem estimating the mapping trajectory as well as the landmark positions.

**Graph Optimization** is used to remove the introduced error from odometry and RADAR measurements. The optimized graph is the globally consistent solution to the SLAM problem. This is done in an offline process after the initial data has been collected, matched and brought into the graph structure. In practice the optimization is done after the car has been put to a stop in the parking garage. The optimized graph serves as the map for subsequent autonomous drives in that environment.

The cost function $J_{\text{Graph}}$ can be viewed as a set of interconnected springs under tension, where the landmark and odometry nodes are the junction points of the springs. The amount of tension is inversely proportional to the measurement accuracy. By releasing and relaxing all the springs, they will morph to an optimal state.

To determine the optimal solution, a Levenberg-Marquadt (LM) algorithm [Mar63] is performed, employing a robust Cauchy kernel of width 1.0 m to ensure that the LM algorithm does not choose an irrelevant local minimum far away from the desired solution. To perform the graph optimization, the g2o [KGS+11b] library is used. It is efficient for large optimization problems and the construction and optimization of the graph are well optimized for the full SLAM problem. The resulting map is stored in an R-tree [Gut84] data structure for fast access times during online localization to guarantee fast matching between incoming landmark data and the stored map.

As can be observed in Figure 4.17, the optimization not only closes the loop of the trajectory correctly, but corrects the position of the landmarks such that they are more clearly representing the physical object. It can be observed, that in the upper left, there are landmarks on a vehicle misaligned significantly before optimization, but after optimization (upper right) the vehicle can be discerned from the landmark

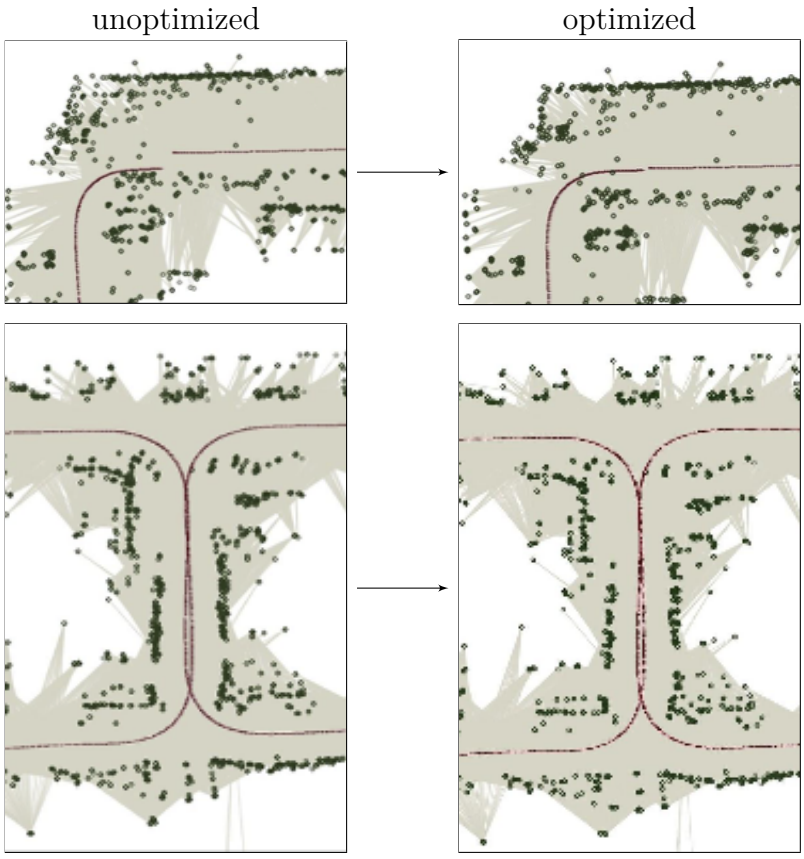



**Figure 4.17:** Graph of two portions of the parking lot before (left) and after (right) optimization. The graph consists of odometry and landmark measurements (vertices, green) and their observations (edges, gray). Before the optimization, the map contains drift error, as depicted in the upper left. The red line indicates the trajectory of the vehicle during mapping, landmarks are depicted in green.

data. Additionally in the lower left image, the landmarks appear to represent a continuous (wall like) structure. After optimization (lower right), one can see, that in fact the structure is not solid, but interrupted. These landmarks correspond to vehicle fronts of parked cars. The entire optimized map is depicted in Figure 4.18.

Having detailed the implementation of the RADAR GraphSLAM approach, in the following section, the practical performance and experimental results are presented in the initially given scenario of a parking lot.

### 4.3.3 Experimental Results

The experimental results are categorized in two main parts. Firstly, the mapping result is analyzed, comparing the optimized RADAR point cloud to the ground truth map generated using a total station. Secondly, the localization performance is evaluated, comparing the trajectory to a ground truth positioning system. The results were obtained on the full *figure of eight shaped* data set from Section 3.3. By performing the localization and mapping on different days, the stability of the implemented RANSAC and the optimization robustness are put to the test.

**The Mapping performance** can be measured based on the quality of the trajectory reconstruction. The mean trajectory reconstruction errors across all 41 sequences in the data set are given by Table 4.1 for two different description methods. The mean error is below 1 m, while errors were in the order of $5 - 10$ m for unoptimized maps, which purely rely on odometry information for trajectory reconstruction. Figure 4.19 shows the growth of the error along the trajectory both before and after the optimization. Without optimization, the error grows to around 7.5 m, while the optimized residual error is limited to around 0.59 m.

Note that the BASD data can be processed significantly faster than the FREAK descriptor, despite needing to process significantly more landmarks. This is due to the

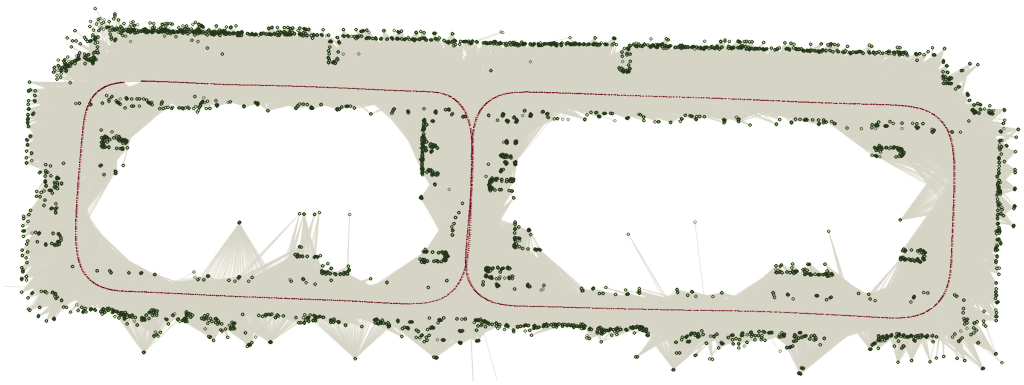


**Figure 4.18:** Fully optimized map using GraphSLAM.

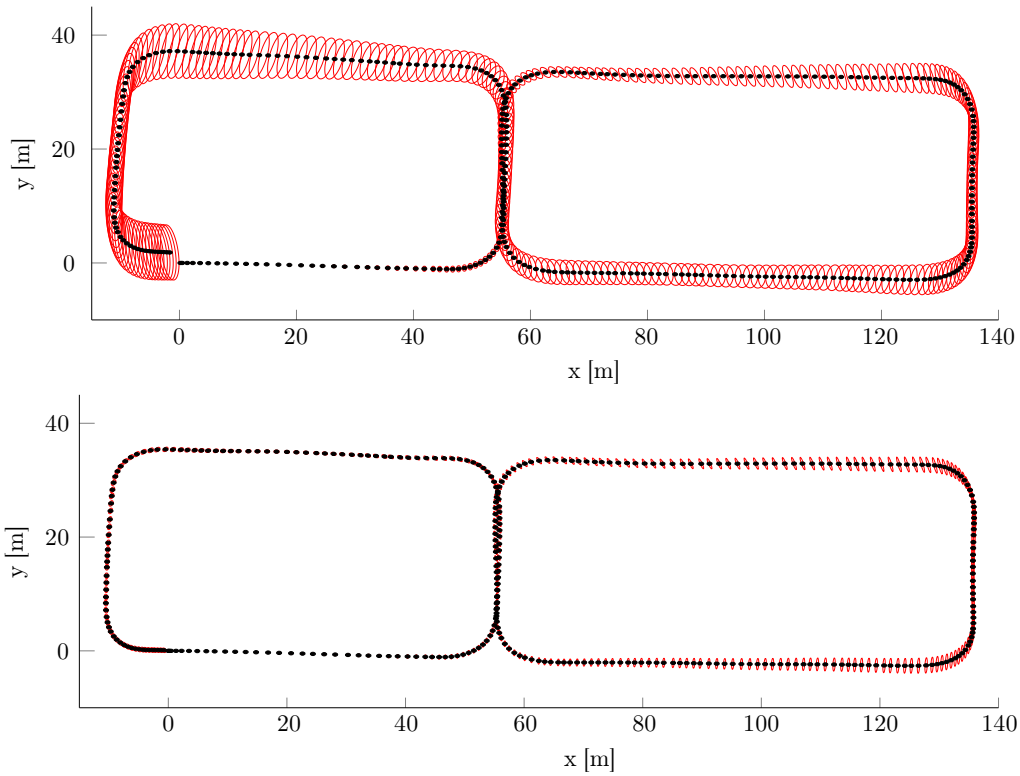



**Figure 4.19:** The trajectory of the vehicle as measured by odometry (upper) and calculated by the optimization (lower). Both plots include error ellipses, describing 95% confidence of the pose.

binary descriptor computing the distance between two descriptors with Hamming distance, which is more efficient.

In the end, both descriptor types yield high precision maps. This is mainly due to the outlier detection via RANSAC. Only a fraction of the features have to be matched between scans, while the rest is found to be outliers. On the other hand, the outlier detection costs processing time. One can see in Table 4.1 that while BASD has by far the most landmarks per map, the processing time is very low (39.2% of real time).

| descriptor | num. landmarks | mean error | processing time |
|---|---|---|---|
| FAST | 5082 | 0.87 m | 65.6 % real time |
| BASD | 7232 | 1.00 m | 39.2 % real time |

**Table 4.1:** Mapping quality and processing speed, as percentage of real time for different descriptors[4].

[4]Values below 100% mean the algorithms run faster than real time.

The position error of the landmarks can be inspected visually in Figure 4.20. The landmarks that fall onto stationary objects are very precisely on the reference map. The ground truth map is displayed in Figure 4.20 in red along with the mapping result in blue. Deviations are mainly caused by parked cars that had been pruned from the reference map. The mean errors per map are depicted in Figure 4.21 and indicate that the resulting mapping procedure produces a highly accurate position for the landmarks in the map. While only one map did not converge, which was due to a parameter error in the odometry, all other optimizations – day and night – resulted in consistently low errors of below 10 cm with respect to the ground truth. The overall mean landmark error over all landmarks and all measurements is $\mu = 6.8\,\text{cm}$. This is measured by finding the nearest neighbor for each landmark in the optimized map in the ground truth map given a rigid transformation between those two maps. The transformation has been determined by selecting 32 hand labeled correspondences between optimized map and ground truth.

**The Localization** result for a single sequence is depicted in Figure 4.22. The difference in maps (black vs. gray) in the background shows that the parking lot was almost fully occupied during mapping and only about half filled when localization was performed. In localization (blue) scenarios with a more recent map or on a completely filled parking lot with not much variation left, the localization accuracy is better than 1 m at all times because localization and mapping result agrees with each other in both lateral and longitudinal direction. In regions of drastic change, where lots of cars were moved, the localization fails due to insufficient number of good matches after the outlier detection. If the maps differ significantly, the localization output is considered unavailable. Figure 4.22 illustrates both the highly accurate localization in similarly shaped areas, as well as disabled localization output (gaps in blue line) due to few good matches in a drastically changed environment.

The overall localization system performed well for maps with minor changes in the environment, while relocating multiple vehicles lead to detection of failure in the outlier detection. In Section 4.4 and Chapter 5, ways to improve availability and performance of the localization will be discussed.

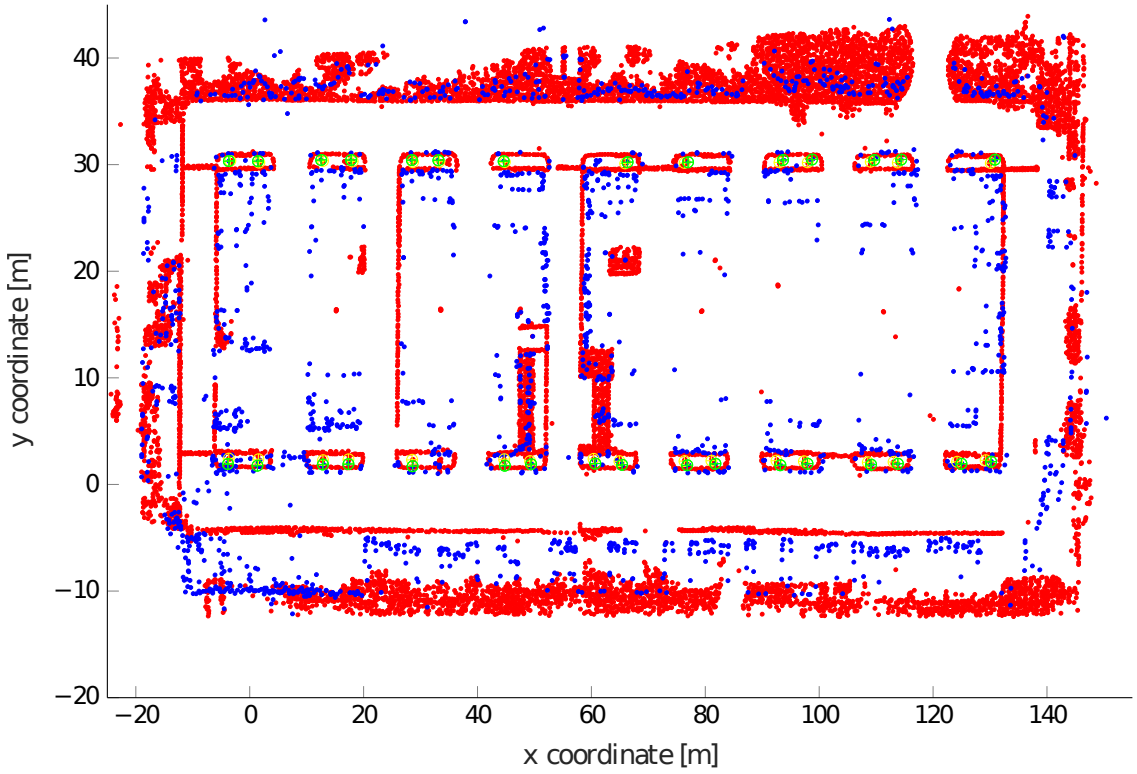



**Figure 4.20:** Overlay of the ground truth (red) and an optimized landmark map (blue), which have been aligned by matching a set of points (green) by hand between the ground truth and optimized landmark map.

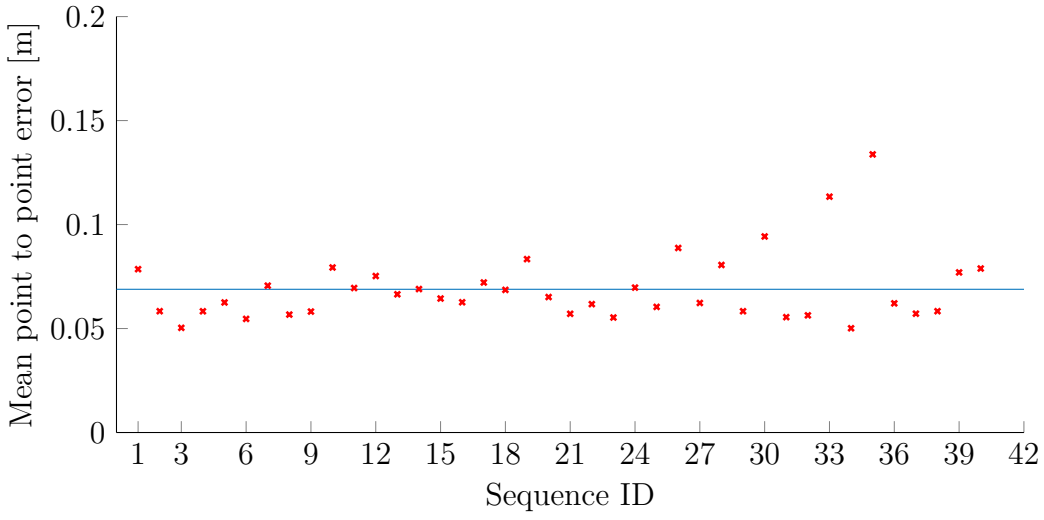



**Figure 4.21:** Mean point error of a selection of 32 hand labeled features. The blue horizontal line at $\mu = 6.8\,\text{cm}$ marks the mean error over all sequences.

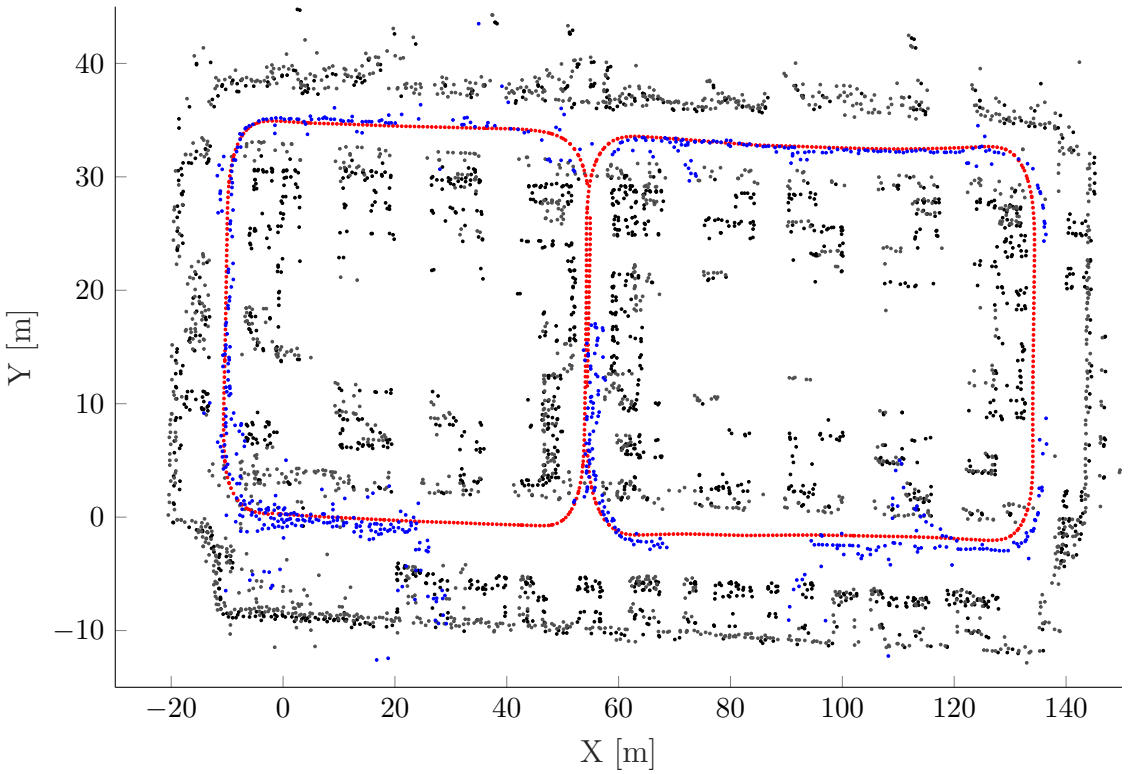



**Figure 4.22:** Estimated vehicle position (blue) vs. ground truth (red). The optimized maps of the mapping (black) and at the time of localization (gray) are depicted. Mapping was performed on an almost fully occupied parking lot, while localization was performed when it was half empty.

### 4.3.4 Conclusion

In this section a robust GraphSLAM system based on distinguishable features from noisy automotive RADAR sensors was introduced. It was shown that using a strong outlier detection and a robust map optimization, the maps become highly representative and the localization becomes robust to outliers. Additionally, the online pose estimation yields very accurate results in case of similar surroundings to the mapping trajectory. Further improvements, such as an online map update and the fusion with more sensors are followed up on in the upcoming chapters.

This section has set the foundation for the GraphSLAM framework used in this thesis that can be expanded in further applications in multiple directions. The most important goal is to add robustness to the system. Higher accuracy and robustness is expected to be obtained by using landmarks from various sources, such as LiDAR and camera, combining all in a common graph to build a map across multiple sensors. Figure 4.22 suggests that by incrementally updating the map during each localization run, one could model changing environments efficiently and close the gaps in localization availability. This is shown in Section 4.4.

## 4.4 Radar Map Updates using Joint Graph Optimization

In this section, a fundamental extension to Section 4.3 is discussed. While in the previous approach, the map is assumed to be static after initial mapping, this section proposes a continuous mapping process across multiple drives and even different cars to accumulate the mapping information of a large area into a crowd based optimization. In this context the parking lot scenario is merely meant as a proof of concept for a large scale cooperative mapping approach unifying maps of individual cars across large areas, such as entire city or highway networks. A landmark based GraphSLAM approach [SKR+16] is especially suitable to serve as a basis for crowd based mapping, because it uses distinguishable landmarks that can be matched independent of sensor model and software framework. The previous chapter thus provided the framework for the expansion into more complex scenarios and more precise and robust localization and mapping performance.

In a crowd based application, the RADAR sensor has an important advantage over LiDAR sensors. The abundance of the sensor in many different cars over many manufacturers makes RADAR input data for mapping a rich data source. Theoretically, every vehicle that has a mobile connection to a backend server and an active cruise control system (which is usually based on RADAR [MVAV11]) can be utilized to contribute to the mapping of large areas for autonomous vehicles.

In this section it is shown that utilizing Joint Graph Optimization, autonomous driving can be enabled in many environments by collaboratively mapping out vast areas very accurately even in areas with many semi-stationary objects, such as a parking area. The presented approach shows that cooperative, non centralized mapping alone – without the need for high precision sensors – is sufficient for autonomous driving. It can also be used to augment manufacturer made maps or eliminate the need for centralized mapping altogether, once enough vehicles are participating.

### 4.4.1 Joint Graph Optimization

The overview of the extended localization system is given in Figure 4.23. Each of the $n$ drives in the parking lot provide both *Raw RADAR Data* and *Vehicle Odometry*, just as in previous sections, which is used to generate the map data. The original part of the architecture (refer to Figure 4.14) was proposed in [SKR+16] and is represented by the gray boxes of Figure 4.23. In this thesis, graph optimization is adjusted to keep global consistency across multiple cars and drives. Each map is registered in a *Global Map Registration* to ensure that they are aligned and in the same coordinate system. The system does not rely on GPS for the registration, but rather on feature matching between the different maps to find common landmarks among the different drives. After the registration has been successful, landmark selection and improvement is performed in the *Map Refinement* step. Landmarks

are selected to keep the map up-to-date. In the following discussions, let the base map be the map on the server that is always updated with the latest information and let the local map be a map recorded by one vehicle at a specific time. The objective is then to consistently integrate all local maps to the base map.

Beside regular pose-to-pose and pose-to-landmark constraints, another type of constraint is introduced. It links the new map increment to the base map. These virtual constraints are calculated via keyframes that are selected equidistantly in space along the trajectory. First, the map trajectory of the local map and the base map are subsampled to form a set of key points $\mathcal{K}_{\text{local}}$ and $\mathcal{K}_{\text{base}}$. Since each map is based in its own coordinate system with a random origin, the key points have to be associated to the map. Thus, individual map increments around the key point are matched with the other map resulting in a set of constraints between the local map and the base map. The principle is shown in Figure 4.24. The key frames from both the base map and the increment map connecting the two maps wherever a match was found. In order to connect both maps, the corresponding features to each key frame from the increment map are matched to key frames from the base map. The key point method was mainly introduced to save processing time. It would have been possible to match both maps without selecting specific regions by just iterating over all landmarks in the map. This would not only require a large amount of processing power, but it also does not improve accuracy. Due to the equidistant sampling of key points, enough matching windows are selected to align

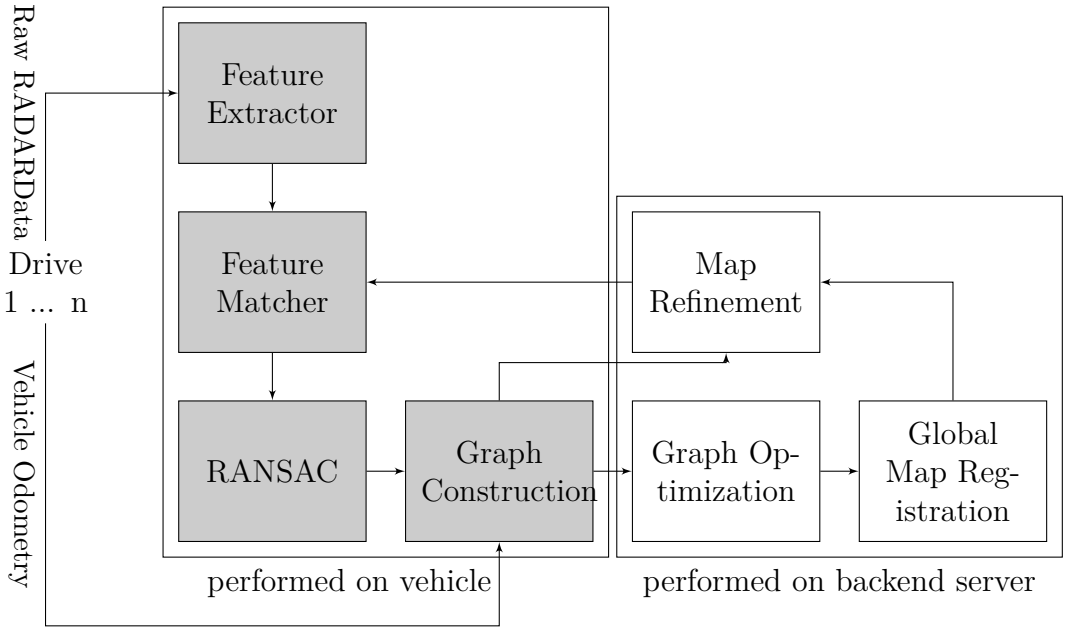



**Figure 4.23:** Overview of the extended GraphSLAM localization system. Gray elements depict the static localization system from [SKR+16], while the white ones show the crowd based addition to the system.

both maps with their corresponding counterparts without having to cross-reference every landmark with all maps.

In summary, the global optimization problem for joint graph optimization is an extension of Equation 2.36 and can be written as

$$J_{\text{JGO}} = J_{\text{Base}} + J_{\text{Local}} + \underbrace{\sum_{i \in K_{\text{local}} \cup \mathcal{K}_{\text{base}}} (x_i - g(x_{m(i)}))^T R_i^{-1} (x_i - g(x_{m(i)}))}_{\text{virtual constraints}}, \tag{4.37}$$

with $J_{\text{Base}}$ representing the constraints of the base map, $J_{\text{Local}}$ the constraints of the local map and $g(x_{m(i)}), R_i$ the transformation of the matching key point $(x_{m(i)})$ of the local map to the key points of the base map $x_i$.

In Figure 4.25, the result of the joint graph registration is displayed. From one frame to the next, another local map is added to the base map and then optimized together with the previous data by solving

$$\hat{x} = \text{argmin}(J_{\text{JGO}}). \tag{4.38}$$

To add robustness, the key frame matches are evaluated via RANSAC to exclude outliers and calculate the transformation between the base map and the local map. Similar to the original Feature Matcher from Section 4.3, a match will add a constraint connecting key frames from different maps. The edge connecting the node stores the confidence of the particular match as determined by RANSAC, utilizing a least squares estimation [Van08] to obtain a covariance to assign a contribution to the Jacobian for each virtual constraint. Additionally, an extension to the RANSAC algorithm proposed in Algorithm 9 is implemented. As shown in Algorithm 10, to improve performance, in each iteration the randomly drawn subset of matches $\mathcal{A}_3$ is rejected immediately, if the target and destination set do not form congruent triangles up to a certain tolerance $t$. This saves many least square estimates and many

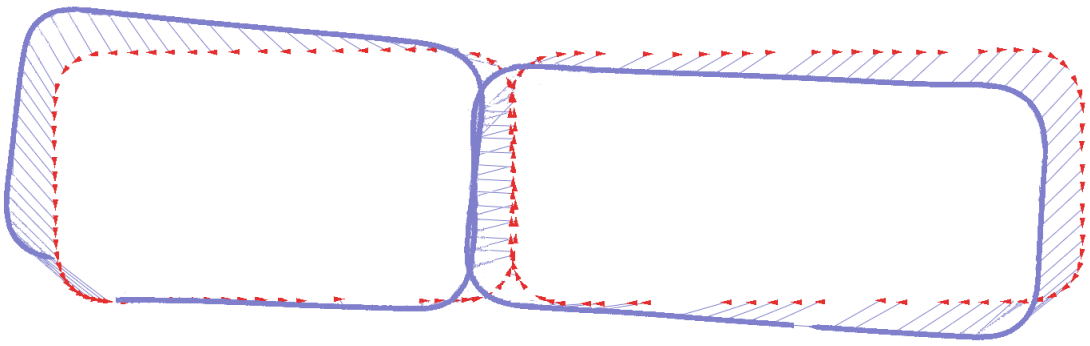
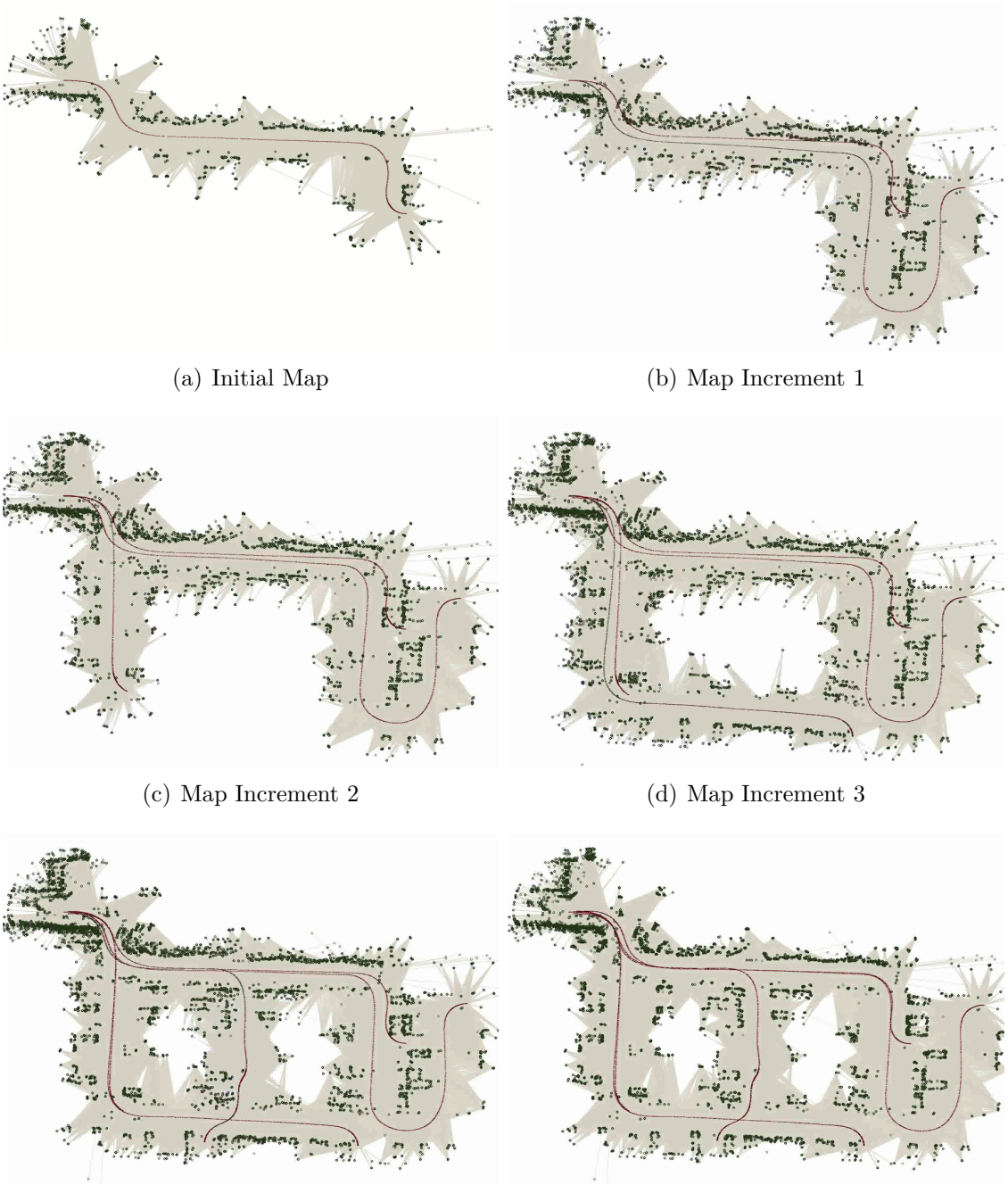


**Figure 4.24:** Local match based registration showing equidistant sampling of the base map (red). The key points are matched to the increment map (blue) and constraints between the two maps are determined.



(a) Initial Map

(b) Map Increment 1

(c) Map Increment 2

(d) Map Increment 3

(e) Map Increment 4

(f) Map Increment 4 – optimized

**Figure 4.25:** Joint optimization of six test drives. From (a) through (e) the graph is optimized before another drive is added. The map gets more consistent incorporating new data during each additional parking drive. In (f) all six drives have been optimized together to form a globally optimized map.

point transformations if the initial set was an outlier. This adjustment is required when matching the key point windows, because many combinations of windows do not have matching counterparts at all. Thus speeding up the RANSAC procedure is an integral part to allow Joint Graph Optimization.

Note that the update is done in an incremental fashion. The base map always stores the most current information. By graph optimization, the most consistent alignment among a new drive is determined and transformed into a common coordinate system. If an incoming increment cannot be matched right away because it does not have any overlapping region to the base map, the key points can be stored as metadata. Once the base map has grown to overlap with the increment, it is integrated. The overall map quality thus increases over the amount of information that is available about the mapping area. In the case of Figure 4.25, the first three trajectories did not provide sufficient information about the surroundings. Five different parking trajectories were needed to build a consistent map of about half the available parking space providing the best mapping quality with consistent information. This is especially supported by the loop closure that individual drives usually do not form, but the combination of drives oftentimes contains.

The more drives are added to the system and the larger the mapped area gets, the more computationally intensive the optimization becomes. The Joint Graph Optimization can be performed in a backend server platform that has virtually unlimited resources available compared to the small computing units in today's production vehicles. The completed map can then be streamed on-demand to each individual vehicle. Nevertheless the map data has to be refined regularly to limit computing time, but most importantly to prevent deterioration of the entire map.

**Algorithm 10** Modified RANSAC

**procedure** RANSAC
  **Input:** set of matches $\mathcal{A}$, tolerance $t$, number of iterations $n_i$
  **Output:** set of inliers $\mathcal{A}'$, transformation $T$, covariance $Cov(\mathcal{X}_{\text{inliers}})$
  $i = 0$
  $\mathcal{A}_{\text{inlier}} = \emptyset$
  **while** $i < n_i$ **do**
    $\mathcal{A}_3 =$ **select_random_subset**$(\mathcal{A}, 3)$
    **if** !**subset_is_congruent**$(\mathcal{A}_3, t)$ **then**
      continue;
    $T =$ **compute_transformation**$(\mathcal{A}_3)$
    $\mathcal{A}^i_{\text{inliers}} =$ **compute_inliers**$(T, d_{\text{thresh}})$
    **if** $|\mathcal{A}_{\text{inliers}}| < |\mathcal{A}^i_{\text{inlier}}|$ **then**
      $\mathcal{A}_{\text{inlier}} = \mathcal{A}^i_{\text{inlier}}$
    $i = i + 1$
  $Cov(\mathcal{X}_{\text{inliers}}) =$ **compute_covariance**$(\mathcal{X}_{\text{inliers}})$
  **return** $\mathcal{X}_{\text{inliers}}, T, Cov(\mathcal{X}_{\text{inliers}})$

### 4.4.2 Map Refinement

Transmitting the map from a server to local vehicles may remove the need for the algorithm to run in real time, but it does not allow the map to become arbitrarily large. As can be seen in Figure 4.25 in the upper left of each image, at the entrance to the parking area, where the local maps overlap the most, the landmark density is growing in the base map with each update, but providing no new information about the surroundings. These landmarks have similar descriptors [RHT+15] and might all be valid matches for new key frames. Besides increasing the map size, this also introduces mapping errors, because many features only vary subtly in descriptor and position provides many potential matches for a new incoming key frame. RANSAC might also not detect the outlier because of the slight change. Thus, in this section a feature rating and selection method is presented after the common registration of multiple maps have been accumulated.

In order to control the local density of landmarks, a clustering algorithm is applied to group landmarks locally. There are a number of ways to cluster a set of two dimensional points in Cartesian space. In this thesis, one of the most basic algorithms was used, the mean shift algorithm with a Gaussian kernel, originally introduced by [FH75] which does not require to preselect the number of clusters.

In Algorithm 11 the algorithm is displayed. The shift point $m(x)$ is the weighted mean based on the neighborhood of points $x_i$ of $x$ and can be computed as

$$m(x) = \frac{\sum_{x_i \in N(x)} \mathcal{N}(x_i - x, \sigma)\, x_i}{\sum_{x_i \in N(x)} \mathcal{N}(x_i - x, \sigma)}, \tag{4.39}$$

with the Gaussian distribution $\mathcal{N}$ and an isotropic kernel width $\sigma \in \mathbb{R}$. In Figure 4.26 the clustered feature points are color coded. For all landmarks per cluster, a selection algorithm is performed removing the most outdated information until the density of a cluster is reduced below a threshold. In order to select the best features, each

**Algorithm 11** Mean Shift Clustering

```
1:  procedure MEAN SHIFT
2:      Input: set of points A, threshold ε
3:      Output: set of clusters C
4:      C = ∅
5:      while A ≠ ∅ do
6:          for all x ∈ A do
7:              m(x) = compute_shift(x)
8:              if ||m(x) − x|| < ε then
9:                  C = C ∪ x
10:                 A = remove_element(A, x)
11:     return C
```

landmark in the cluster is assigned an age $\omega$ and a degree $\delta$. The age relative to the current map increment $m_0$ is given by

$$\omega = m_0 - \max\{m \leq m_0 : \text{landmark not observed in } m\}. \tag{4.40}$$

The age formula describes the number of consecutive local maps, where the feature has not been found in. Thus, only landmarks that have been recently observed will end up in the map. Note that this algorithm does not remove the parked vehicles in the parking lot, but rather empties parking spots that are no longer in use. The degree $\omega$ denotes the amount of maps, a landmark was observed in the total set of maps that form the base map $\mathcal{M}$:

$$\delta = |\{m \in \mathcal{M} | \text{landmark observed in } m\}|. \tag{4.41}$$

The features in each cluster are first sorted by their age starting with the youngest. In the case that multiple landmarks have the same age, the degree is considered. Landmarks that are oldest or observed least will be removed from the map only leaving the most current and stable landmarks.

The ranking is performed for each cluster that exceeds a certain maximum density. Landmarks are removed according to the above sorting until the density is beneath the maximum density. Each time another increment is added to the base map, the clusters are re-evaluated. Thus, the map size saturates when a lot of cars pass a specific region frequently.

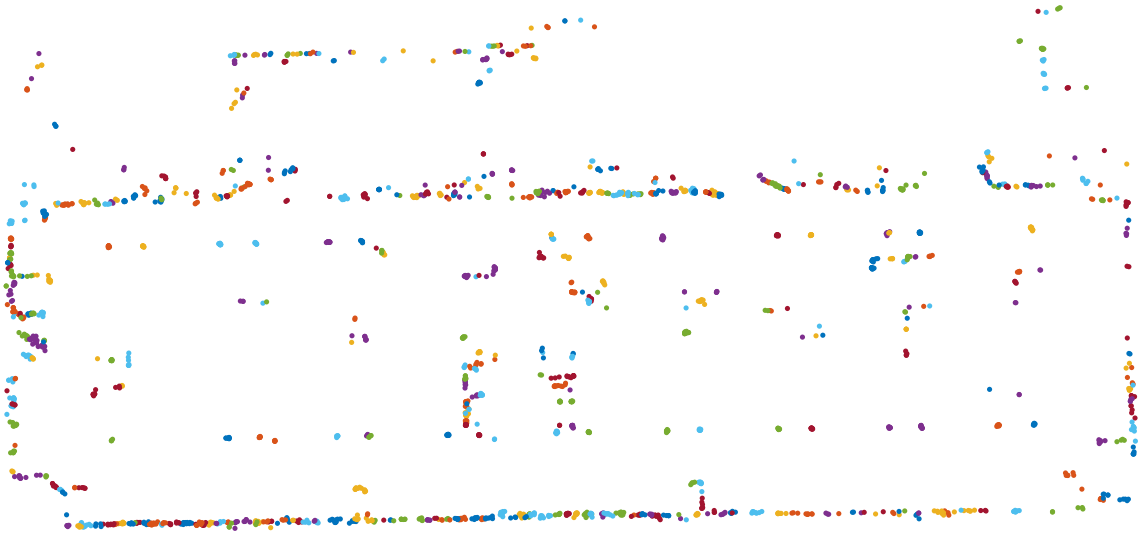


**Figure 4.26:** Clustering of landmarks in a map of the parking area. Different colors indicate different clusters.

### 4.4.3 Experimental Results

The experimental results are categorized into two main parts. The mapping accuracy of the joint graph optimization is analyzed by comparing the mapping result to a highly accurate reference map generated by a total station [ZLN09]. Localization accuracy is measured by comparing the trajectories both in map refinement and during localization to a DGNSS/INS system. Additionally, the map size improvement due to the map refinement is studied. In the following sections, both the mapping and localization performance is evaluated on both data sets. Finally, the effects of the feature selection are evaluated.

The experiments were conducted on the parking area in Figure 3.4 using 16 *figure of eight shaped datasets* at different times, during daytimes and across seasons ranging from June until December and about 15 *real parking maneuvers* from two different vehicles equipped with RADAR and odometry sensors.

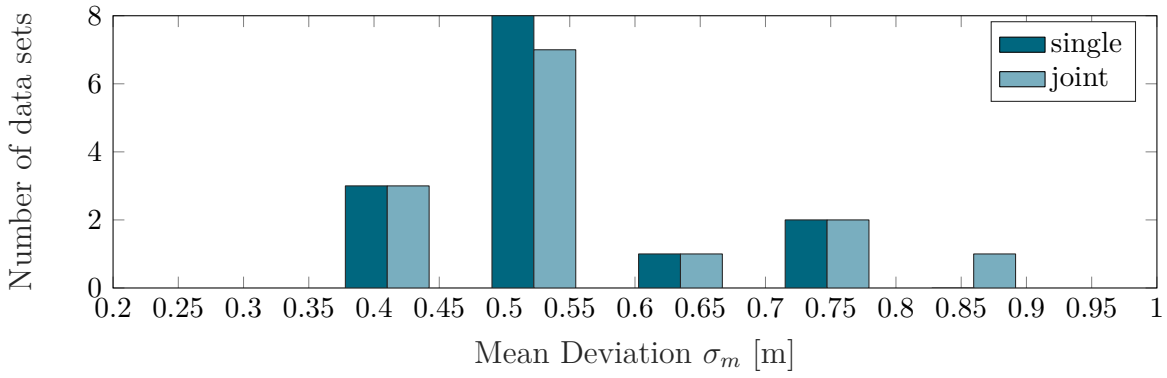
**The accuracy of Joint Graph Optimization** is shown in Figure 4.27. It compares each optimized trajectory individually to the ground truth via Root Mean-Squared Error (RMSE). The deviation can be expressed as the root mean squared error:

$$\sigma_m^2 = \frac{1}{|T|n_t} \sum_{p \in T} \sum_t ||x_p(t) - x_g(t)||^2,$$

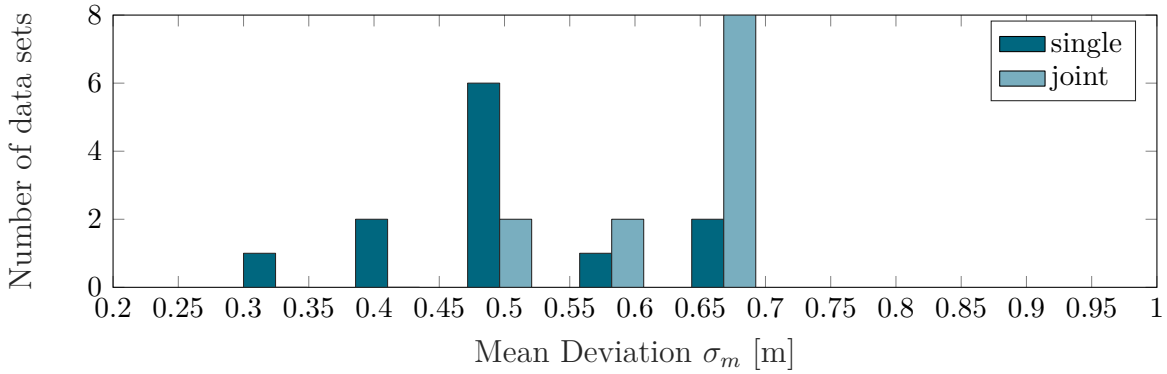
where $p \in T$ are all poses in the trajectory $T$ and $n_t$ are all time stamps for which a localization output was computed. Each position $x_p$ is compared to the corresponding ground truth position $x_g$. The mean deviation $\sigma_m$ is then computed for each trajectory in the data set. Figure 4.27 (a) and (b) show the distribution of the mean error over all the trajectories.

The comparison between the joint graph registration and the individual processing does not seem to show any improvement in mapping quality at first glance. In Figure 4.27 (b) even, the overall mapping performance seems to decrease slightly. This effect can be explained by two influencing factors: While in Figure 4.27 (a) the eight shaped dataset is mostly covering the same area, landmarks are usually shared between the individual drives. The positions of these well known landmarks are static and do not change relatively to each other. Thus the optimization error remains constant even throughout visiting the area multiple times. The residual error results mostly from calibration offsets and the natural limitation of the RADAR sensor, which is discussed in Section 3.3. Another effect might be the different conditions during each drive. Different cars and different temperatures might affect the odometry offset estimation and thus cause inconsistent drifts through different sequences. In the real parking maneuvers of Figure 4.27 (b) on the other hand, each sequence explores mostly new regions of the parking lot. Thus there are only very few common landmarks visible in all the different passings. Therefore, individual landmark observations have a larger impact and slightly deteriorate the

mapping results. With more passings of the same environment though, the error will converge to the best possible results with the given sensor setup. In the real parking maneuvers, a larger area has been covered, as visualized in Figure 4.25. Additionally the base map stores the most recent information. This has an impact on localization accuracy, since each car can benefit from the joint information of all previous drives. Thus in Figure 4.28, the localization accuracy in the overlapping eight shaped dataset has been analyzed. The parking lot is prone to a lot of changes in the environment due to cars entering and leaving. In a complicated scenario like this, the map update provides a significant improvement in localization accuracy. This holds true for both the mean performance, improving from 0.79 m to 0.55 m with respect to an outdated map, but most importantly the outlier trajectories have improved significantly. We can see by the tail of the distribution in Figure 4.28 that even in worst case scenarios, the localization performance is improved significantly. The former will have a direct impact on the parking performance of an autonomous system, while the latter will raise localization availability significantly.




(a) The *eight shaped dataset.*




(b) The *real parking maneuvers.*

**Figure 4.27:** Histogram of deviations from the ground truth. The dark blue color indicates the individually processed maps, the light blue the joint optimization.

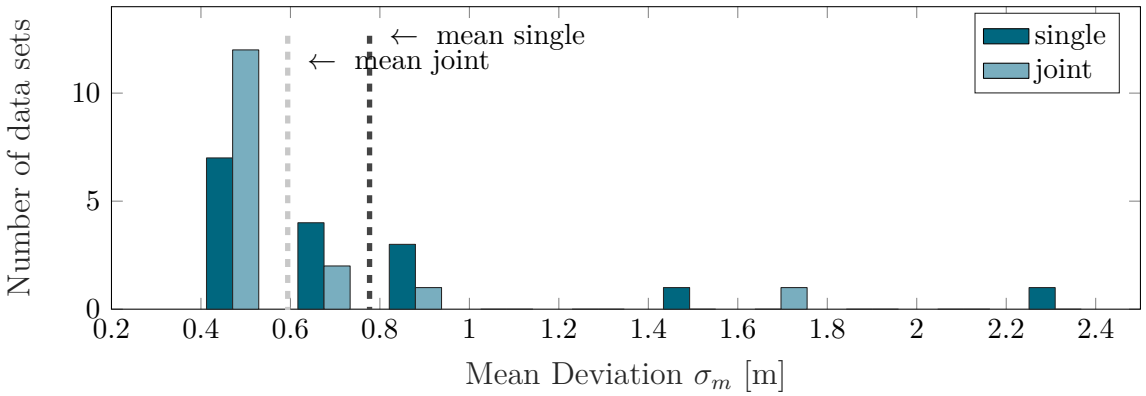
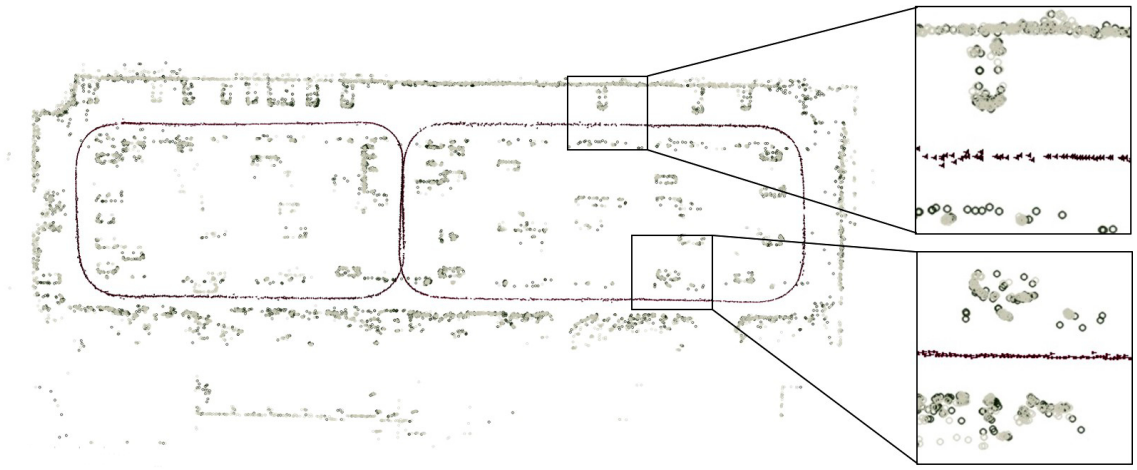
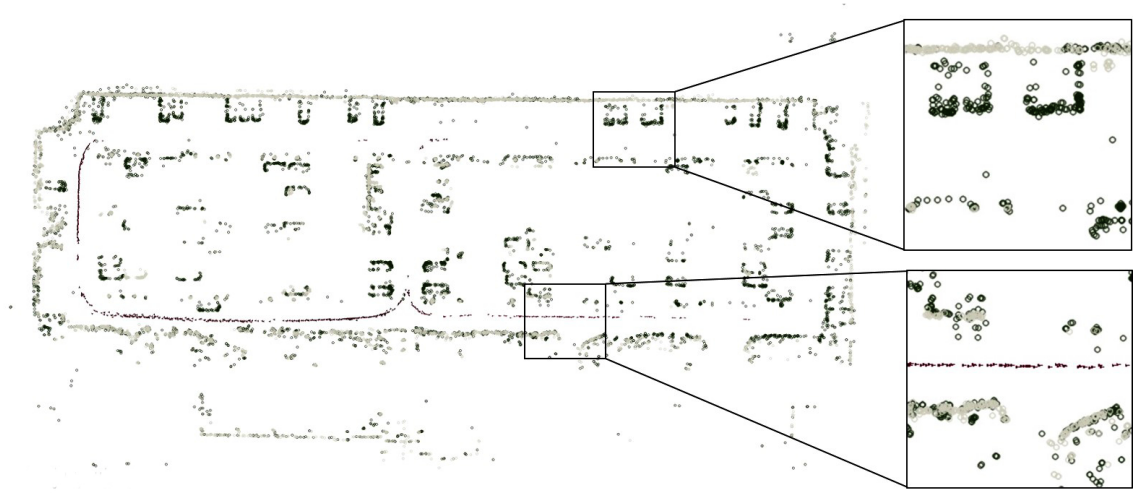



**Figure 4.28:** Localization accuracy both with and without joint graph registration. On average, localization accuracy with single optimization is 0.79 m. The localization accuracy with joint graph registration is 0.55 m.

As visualization, the different map update steps are shown in Figure 4.29. It can be seen that the localization performance deteriorates in places with large differences between the reference map (light gray) and the current map (dark gray). This can be observed in the cutout windows of Figure 4.29. One can observe that the localization system is robust against small changes in landmark configurations, but with big differences in configuration, the localization result breaks down due to insufficient matches. Nevertheless Joint Graph Optimization produces an up-to-date and highly precise map increment that maintains the map independent from the performance of the online localization, because the Joint Graph Optimization can be performed at the end of the drive with all relevant information.



(a) Localization on an up-to-date map.



(b) Localization on an outdated map.

**Figure 4.29:** Comparison of the localization performance and availability for the updated map (a) and the outdated map (b). The grey dots represent the map that was used for reference, while the black dots indicate landmarks measured during localization. The triangles indicate the localization result.

**Availability of the localization** typically describes the percentage of time or distance where the localization algorithm was able to produce a useful result at all. In Figure 4.29, the availability when localizing on an outdated map is significantly lower than localizing on an up-to-date map, as can be seen when comparing the amount of estimated positions (triangles) along the trajectory between Figure 4.29 (a) and Figure 4.29 (b).

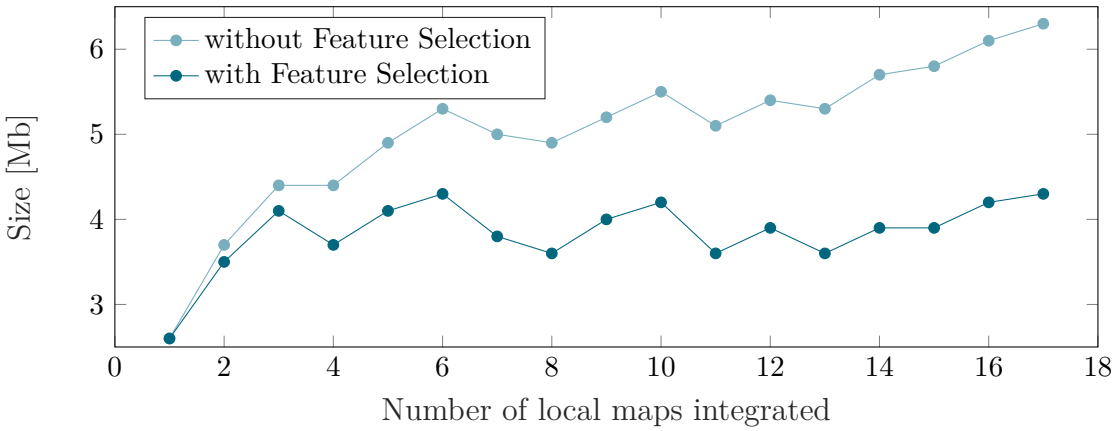



**Figure 4.30:** Feature map size on the *eight shaped dataset*, where all trajectories overlap both with feature selection (dark blue) and without (light blue).

**The Size of the Feature Map** is illustrated in Figure 4.30 with respect to the *figure of eight shaped dataset*. Since all trajectories on this dataset cover roughly the same area, the map stores a similar amount of information after each update. Without applying feature selection, the map size grows linearly with each additional local map contribution, increasing by 160 % after only 16 updates. This is impractical for large scale applications and deteriorates the map quality significantly. With the result of the pruning based on mean shift clustering described in Section 4.4.2, the size of the feature map is maintained at a near constant level, while improving the localization accuracy, as could be seen in Figure 4.28. With this important addition the crowd based approach scales well over time and avoids adding redundant information. The number of landmarks and thus the size of the map thus merely depends on the area covered by the map. In the case of the eight shaped dataset with roughly 140 m by 50 m of mapped area the map size would be about 571 MB/km$^2$.

### 4.4.4 Conclusion

In Section 4.4, a unique map update mechanism for crowd based localization applications was presented. It was shown how multiple cars with different sensor setups contribute to a global map via Joint Graph Optimization. Removing redundant information from the map by clustering and ranking landmarks, both improved localization performance and large scale applicability.

Summarizing the presented algorithm in Section 4.4, the initially introduced RADAR based GraphSLAM idea [SKR+16] is extended to crowd based mapping for multiple cars in a landmark based approach by joint graph registration and optimization. The interconnection of landmarks is achieved via key points that link individual local maps to the base map. To remove outdated landmarks that are no longer seen, because they belong to semi-static objects, a pruning based on mean-shift clustering and feature rating is implemented.

The evaluation shows three main benefits of this approach:

- Mapping quality remains constant even when adding unexplored areas due to joint graph optimization.
- Localization quality is improved by keeping the map up-to-date.
- The data size per mapped area remains constant due to feature selection.

This section concludes the elaboration on the RADAR GraphSLAM framework. In the following chapters the applications of the framework to more general approaches in localization are discussed. This will include adding different sensor types in addition to RADAR to the localization system. Especially automotive LiDAR sensors are becoming more popular among car manufacturers for autonomous driving. This might add further accuracy and robustness because of the higher resolution and range. Furthermore, it is investigated whether parked cars can be classified in the feature maps based on the descriptors of the RADAR landmarks.

# 5 Sensor Fusion in GraphSLAM

In this chapter the sensor fusion capability of the GraphSLAM framework is introduced. It was shown in previous chapters that using RADAR data for localization on a parking lot is sufficiently accurate, but in order to develop fail-operational autonomous driving solutions, the robustness and reliability are especially important. To establish a more reliable system that includes any subset of sensors, the GraphSLAM framework introduced in Section 4.4 serves as a platform.

The RADAR GraphSLAM framework in Chapter 4 was developed such that it can be extended easily by different modules to integrate landmarks from different sensors. As the architecture is already separated into a front end and back end part, the framework was modularized even further to enable multiple Feature Matcher modules for multiple sensors to contribute to the online Graph construction. The multi-sensor adaptation of the framework is displayed in Figure 5.1. The system can

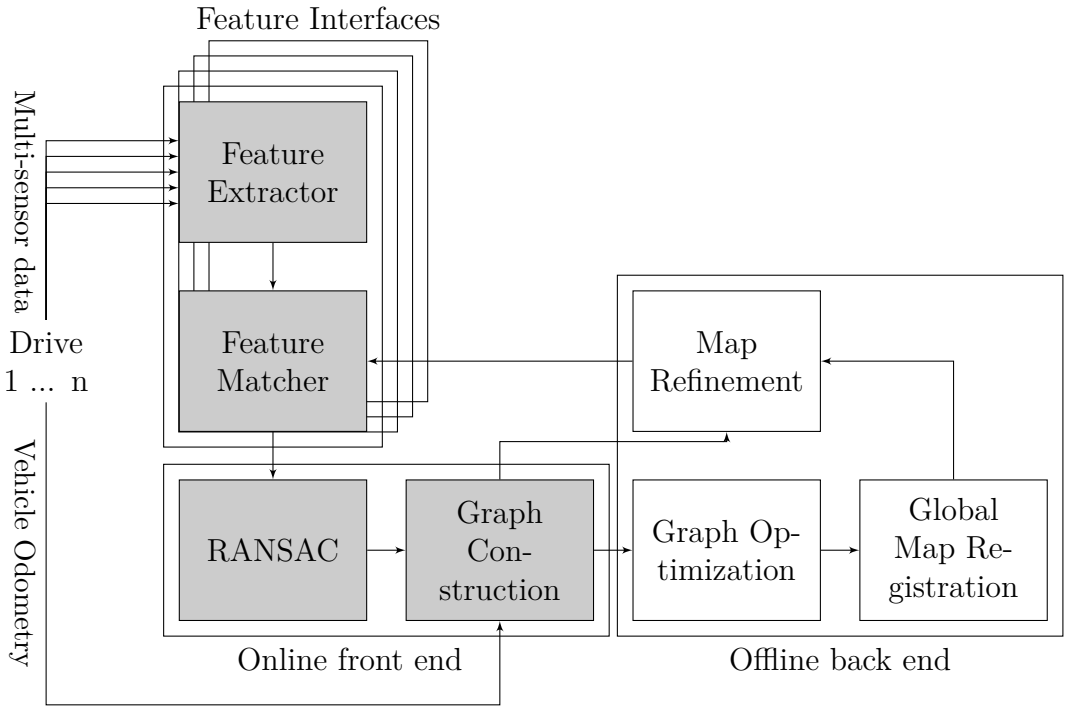



**Figure 5.1:** Overview of the GraphSLAM localization system. The gray elements depict the static localization system from [SKR+16], while the white ones show the crowd based addition to the system.

process multiple input streams of sensor data by adding different Feature Extractor and Feature Matcher components for each sensor. The RANSAC and Graph Construction then filter the outliers from all sensors and construct a graph that contains nodes and edges from multiple sensors, all connected to the same backbone of odometry poses. In the GraphSLAM back end the graph optimization remains completely unchanged. This architecture allows for all sensor dependability to be hidden in the feature extraction part of the SLAM algorithm. Most of the other code parts remain modular and are unchanged when extending to multiple sensors.

The cost function that contains all constraints of the graph optimization Equation 4.37 must then consequently be extended to

$$J_{\text{Graph,multiSens}} = \underbrace{x_0^T \Omega_0 x_0}_{\text{map origin constraint}} + \underbrace{\sum_{ij} e_{ij}^{\text{odo}}(x_i, x_j)^T \Omega_{ij}^{\text{odo}} e_{ij}^{\text{odo}}(x_i, x_j)}_{\text{odometry constraints}} + \underbrace{\sum_{s_i \in \mathcal{S}} \sum_{i,l} e_{il}^{s_i}(x_i, m_l)^T \Omega_{il}^{s_i} e_{il}^{s_i}(x_i, m_l)}_{\text{landmark measurement constraints for different sensors}}, \tag{5.1}$$

where $s_i \in \mathcal{S}$ are the different sensors and each constraint $e^T \Omega e$ is now sensor dependent. The correct parameters are determined by the Feature Matcher and the RANSAC. Note that there are no empirical weighting factors that prioritize the sensor inputs among each other. The graph optimization determines the most consistent result based on the quality of the input data alone.

Now that data from multiple sensors have to be processed, the time synchronization is important. Since the vehicle in Figure 3.1 has free running sensors that are not triggered externally, latency effects come into play. Fortunately the ADTF framework described in Section 3.1 handles the time synchronization and assigns timestamps according to their arrival time at the PC. It also assures that the data reaches the Feature Matcher with the correct latency when simulating a measurement. For testing outside the framework, a simple routine was built that can sort different sensor inputs based on their timestamps. The final evaluation was performed in the ADTF framework in order to have the best timing constraints possible.

In the following, two different sensor fusion setups are discussed. An extensive study has been performed on the integration of the LiDAR sensor and a Feature Extractor based on the Hough transform. Additionally in collaboration with [Muf18], a stereo camera based GraphSLAM algorithm was performed on Stixel landmarks. This chapter discusses the benefits of sensor fusion in the GraphSLAM framework with respect to the localization accuracy and availability in the parking lot scenario and on urban routes.

## 5.1 LiDAR Landmark Integration

The LiDAR sensor is complementary to the RADAR sensor in many ways. The accuracy and angular resolution of the LiDAR is significantly better and noise is not an issue in ideal conditions. But the LiDAR sensor requires line of sight, cannot easily be mounted in a desirable spot in a production vehicle and becomes increasingly noisy in unfavorable weather conditions. Because RADAR and LiDAR are complementary in many ways, fusing the two in a GraphSLAM is highly desirable. In order to successfully integrate the LiDAR into the GraphSLAM framework, first features have to be extracted from the point cloud and then the features have to be integrated into the graph.

### 5.1.1 Generation of LiDAR landmarks

The landmark generation from LiDAR has been studied for various use cases like selecting regions of interest [TA10a]. They showed that LiDAR data can be clustered by a hierarchical clustering algorithm in order to determine regions of interest for classification. Each cluster could be identified by geometric features. For the purpose of this thesis, a simple feature extractor compatible with the established RADAR GraphSLAM framework, was developed. This includes the generation of a descriptor for each detected feature to enable the detection of loop closures. It should operate in real time, use the information from a single scan only and detect the same features from any view point (rotation invariance).

One possible approach analogous to Section 4.2 requires clustering LiDAR data. This has been performed by [TA10a] to combine LiDAR data into a hierarchical clustering of LiDAR point clouds and describing them by a bag of words approach. However, in this thesis another, simpler approach is taken that takes the measurement model into account. While RADAR data tends to accumulate in scatter centers that can be found by a clustering algorithm, LiDAR data follows an angular sweeping and thus forms a particular line shape that can be analyzed for features suitable for localization. Similar approaches [BZ09] use the Hough transform to estimate the shape of the surroundings, which provides excellent results for line and circular features, but breaks down for small diameter features. For the laser scanner used in Section 3.3, the point density of the automotive LiDAR scanner is oftentimes not dense enough to find small features this way. Another approach [YO10] uses polar histograms in a neighborhood of points to create scale invariant features, which is a suitable method to handle scale invariance. However the algorithm does not propagate the geometry provided by the scan as the Hough transformation would and is thus susceptible for occlusions if only using a single scan. Thus in this chapter, another approach based on the curvature of the extracted geometry information is proposed.

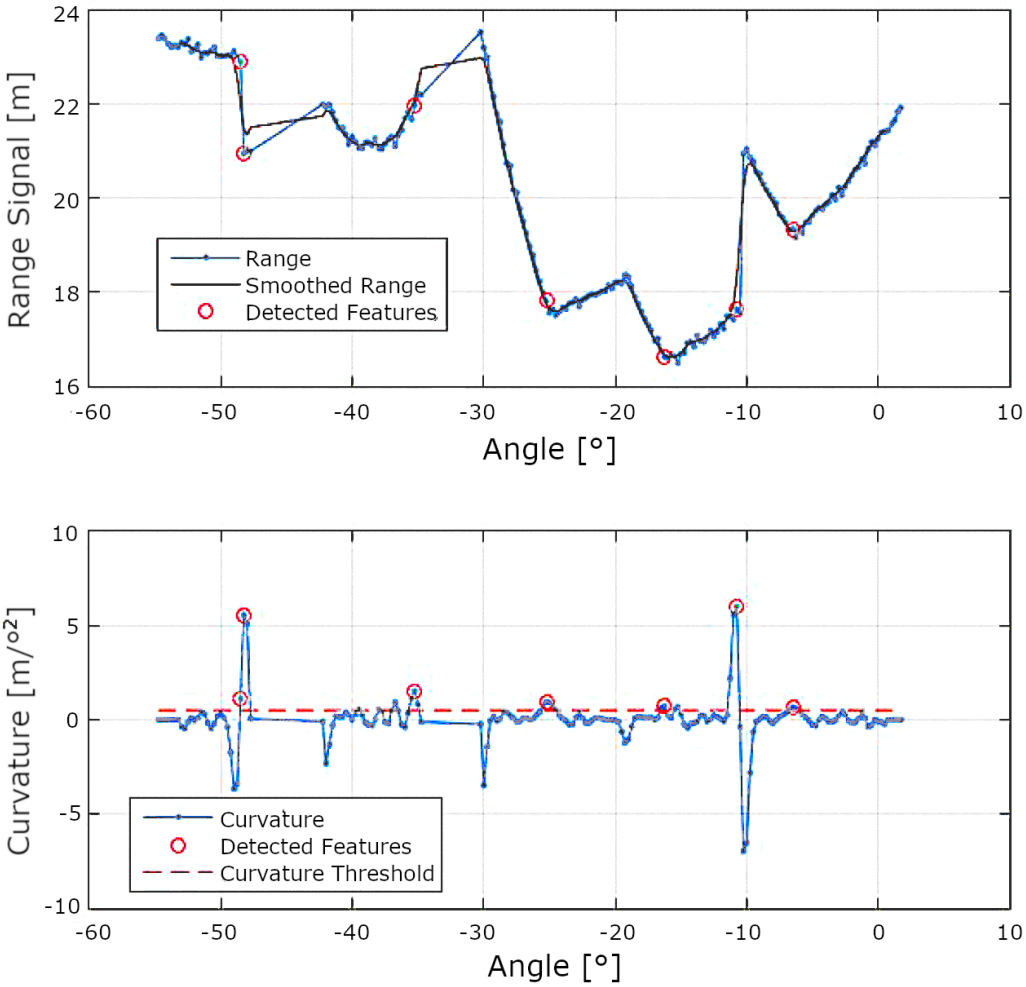
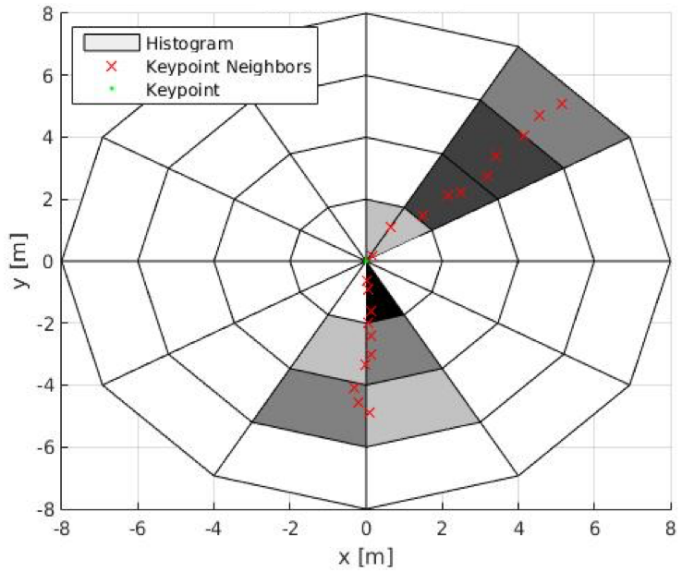
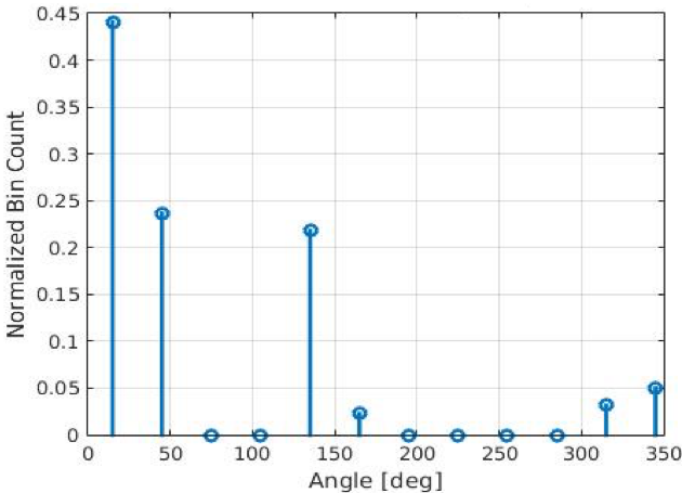



**Figure 5.2:** Curvature based feature detection. The second derivative of the upper image is shown in the lower image.

By fitting a polygon to the smooth LiDAR data, as displayed in the upper part of Figure 5.2, the curvature of the geometry can be obtained by calculating the numerical second derivative from the raw scan, see lower part of Figure 5.2. The approach is to find and select the correct peaks, which correspond to sharp corners or discontinuities. Note that in Figure 5.2 all spikes in curvature are detected, but none of the convex corners. This is done intentionally, because it would create *shadow features* from occlusion, which move while the car is moving.




(a) Polar histogram visualizing radial and angular bins.




(b) Normalized Histogram over angular bins.

**Figure 5.3:** The descriptor of the LiDAR landmarks uses polar histograms around the detected point.

The description of the landmarks follows a standard method [Kri14]. A polar histogram is plotted around every point, see Figure 5.3 (a) and per angular and radial bin, the number of LiDAR points for each bin is stored, as shown in Figure 5.3 (b). The vector is normalized to help with scale invariance, and shifted to the dominant orientation, which refers to the orientation with the highest count to make the descriptor rotation invariant.

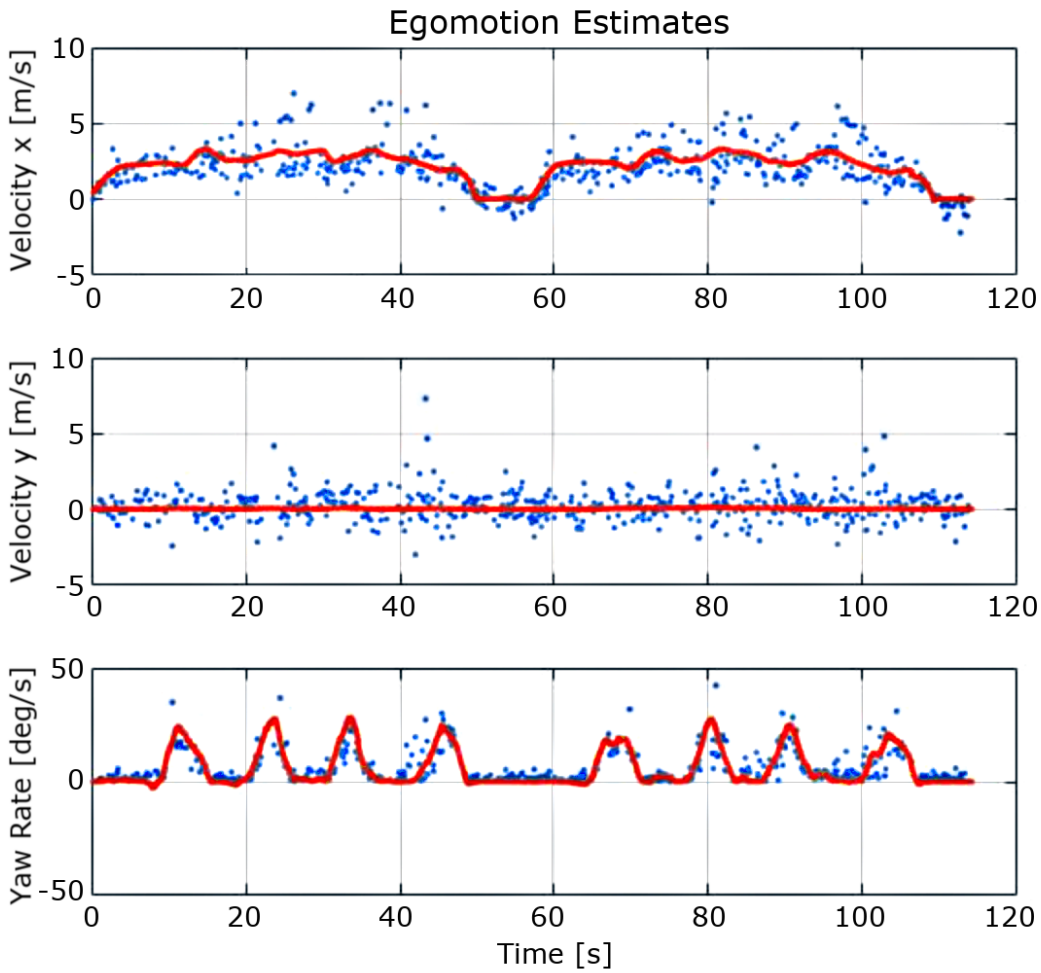



**Figure 5.4:** Ego motion estimation from LiDAR landmarks (blue dots) vs. reference measurement from the iTrace system (red line).

This rather simple approach was evaluated by performing a feature matching between two scans at $t_i$ and $t_{i+1}$ estimating ego motion from all the sequences of the figure-of-eight shaped dataset, as described in Section 3.3. The results for one trajectory are shown in Figure 5.4. The ego motion follows the reference nicely with few outliers. Note that no outlier rejection was performed in order to evaluate the native tracking capability of the feature extractor and descriptor with regards to feature stability and recognizability. While there are still some outliers found in Figure 5.4, the overall performance was deemed sufficient, as there are no systematic shifts in $x$ and $y$ direction and a precise heading estimation. Therefore, the landmarks are stable and recognizable enough to be used in GraphSLAM.

### 5.1.2 GraphSLAM with LiDAR

The LiDAR and RADAR fusion is straight-forward from an implementation point of view. As described in Section 4.3 the landmark data is stored in an R-tree [Gut84]. In order to handle objects from different sensors, two separate data stores are used to ensure that with each incoming scan, only landmarks from the same sensors are matched with each other. Apart from that, there are no changes necessary to the architecture described in Figure 2.14. The timing between RADAR, LiDAR and odometry sensors is handled by the unique timestamps provided by ADTF, which are assigned when the data reaches the vehicle PC. This means that some latency is introduced by the different measurement principles of LiDAR and RADAR[1], but since these latencies are typically small compared to the measurement error, they are not estimated separately in this approach. Instead the data is processed according to the ADTF timestamp.

In Figure 5.5 an unoptimized map of RADAR and LiDAR features combined is displayed. Note that due to different underlying physical measurement principles and a different landmark generation process, the features do oftentimes not describe the same physical object or one sensor is more distorted than the other. Nevertheless, the optimization is able to compensate for these effects and produce a highly accurate fusion map. The computation of such a map is explained in the following section.

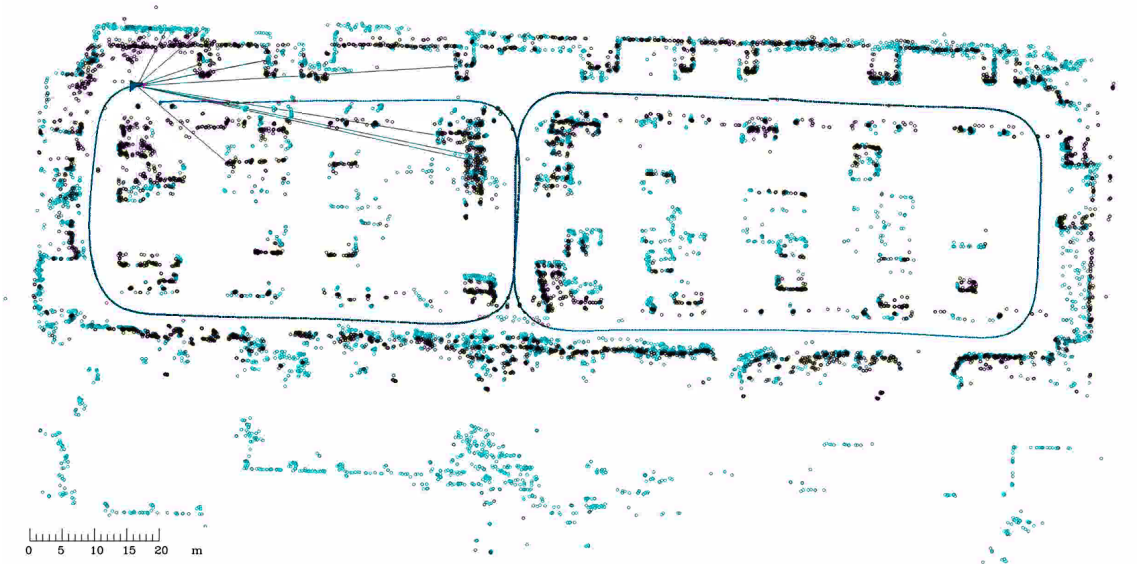



**Figure 5.5:** RADAR and LiDAR landmarks (unoptimized) in the same optimization problem with the LiDAR data in light blue and the RADAR data in black.

[1]LiDAR sensors use mirrors to deflect the beams, while RADAR has no moving parts.

### 5.1.3 Evaluation

In the evaluation of the RADAR and LiDAR fusion, the results are processed on all the datasets described in Section 3.3. In the following sections both the mapping and localization performance is analyzed for the eight shaped dataset and the realistic parking maneuvers. The localization accuracy between RADAR, LiDAR and fusion is compared.

**Initial Mapping** accuracy is displayed in Figure 5.6. The RADAR performance from Section 4.4 is compared to the LiDAR sensor and the fused result. As expected the LiDAR sensor yields better pose estimation during mapping both due to the higher resolution and the absence of clutter. The LiDAR sensor yields an average mapping error of under 30 cm, which is about half of the RADAR sensor. The fused result on the other hand has an average localization error of around 44 cm and thus yields results between the RADAR and the LiDAR sensor which is expected in a fusion algorithm using both sensors simultaneously. Reasons for this can be systematic measurement errors of the individual sensors and small misalignments in calibration. Any more occluded areas might profit more from the 360 ° coverage of the RADAR, which helps in more complex parking space maneuvering scenarios.

**Joint Graph Optimization** was performed on the realistic parking maneuvers described in Section 3.3. The results for the fusion based approach are shown in Figure 5.7. When combining all the data to one unified map, the keypoint con-

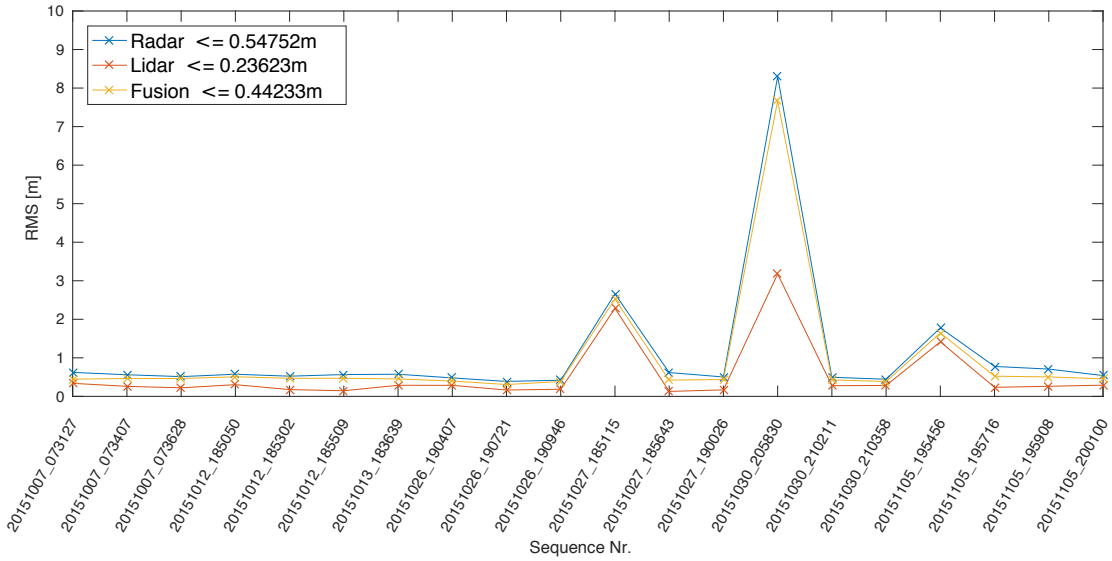



**Figure 5.6:** Mean pose error of the initial mapping for RADAR, LiDAR and fused result per trajectory.

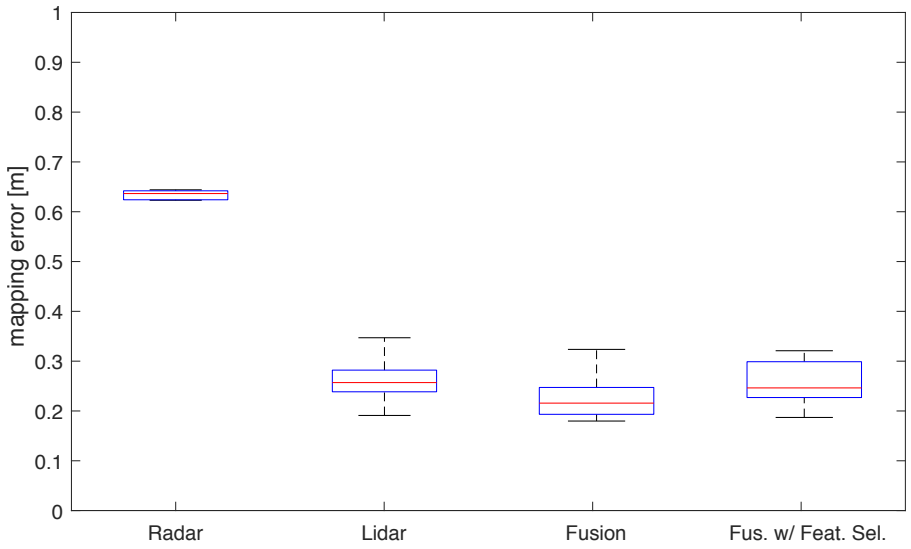



**Figure 5.7:** The Joint Graph Optimization mapping error for RADAR, LiDAR and fusion on the realistic parking maneuvers. The fusion is evaluated both with and without feature selection. The boxes indicate the quantiles and the bars indicate minimum and maximum errors.

straints between the reference map and the local map are most important. In this case, due to the 360 ° vision of the RADAR sensor, it can contribute more to the overall fusion result than the LiDAR with its very limited FOV can. Thus the fused Joint Graph Optimization result improves with respect to the LiDAR when adding the RADAR sensor, despite the singular RADAR performance being significantly worse than the singular performance of the LiDAR. The mean fused mapping accuracy with joint graph optimization is thus only 22 cm, compared to 28 cm with just the LiDAR sensor. Furthermore, the performance of the feature selection is comparable to the fused result with all the landmarks, but with significantly less data. Thus gaining an up-to-date map at all times outweighs the slight deterioration of map accuracy by the localization performance in Figure 5.7, which is comparable to the LiDAR results.

The fused map from Joint Graph Optimization is displayed in Figure 5.8. The map was registered to the ground truth map from the total station by manually selecting equivalent points, such as tree trunks and lamp posts, which could be identified in both maps. The RMS error of the transformation was approximately 35 cm, which is in the order of magnitude of the mapping trajectory error. In Figure 5.8, the color indicates the distance between points from both maps. As can be seen, the static objects are located very precisely in the map (low point distance, blue), while the cars in the center of the parking lot all have no reference point in the reference map. This is due to the removal of all parked cars from the reference map to be able to analyze the static environment. The static points are located on the curbs, bike stands and fences in the parking lot and are all represented in various shades

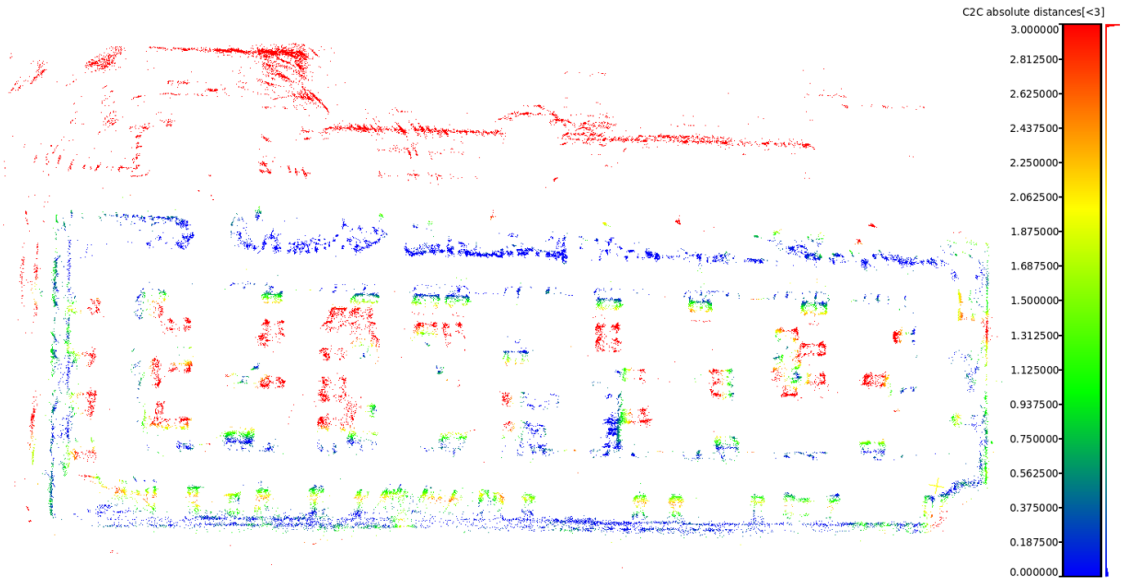



**Figure 5.8:** The finalized jointly optimized map from sensor fusion with respect to the total station.

of blue in the map, representing a landmark error of less than 35 cm. Points that do not correspond to static objects in the feature map are still an important part of the map though, as they represent the dynamic environment and thus support the localization significantly.

**The Localization** performance depends mostly on the feature matching accuracy between the current scan and the map after the RANSAC was carried out. If there are at least ten corresponding feature pairs with an error below 50 cm, the feature matching can be considered a good match and is used for pose estimation. In Table 5.1 the average uncertainty as reported by Algorithm 10 in Section 4.4 is displayed for one sequence of the dataset, which yields an average position estimation error of 35 cm for RADAR only and 50 cm for LiDAR only, while the fused result averages at 24 cm across for this very sequence.

The distance to ground truth is plotted over time in Figure 5.9. Because the trajectory on the parking lot has many sharp turns, the LiDAR sensor has difficulties finding good sets of matches with at least 10 landmarks, thus the position is prone

| | $\sigma_x$ [m] | $\sigma_y$ [m] | $\sigma_\theta$ [rad] |
|---|---|---|---|
| Radar | 0.129 | 0.137 | 0.016 |
| Lidar | 0.141 | 0.207 | 0.009 |

**Table 5.1:** Average uncertainty of the feature matching estimation described in $x, y, \theta$ by the standard deviation $\sigma$ respectively.

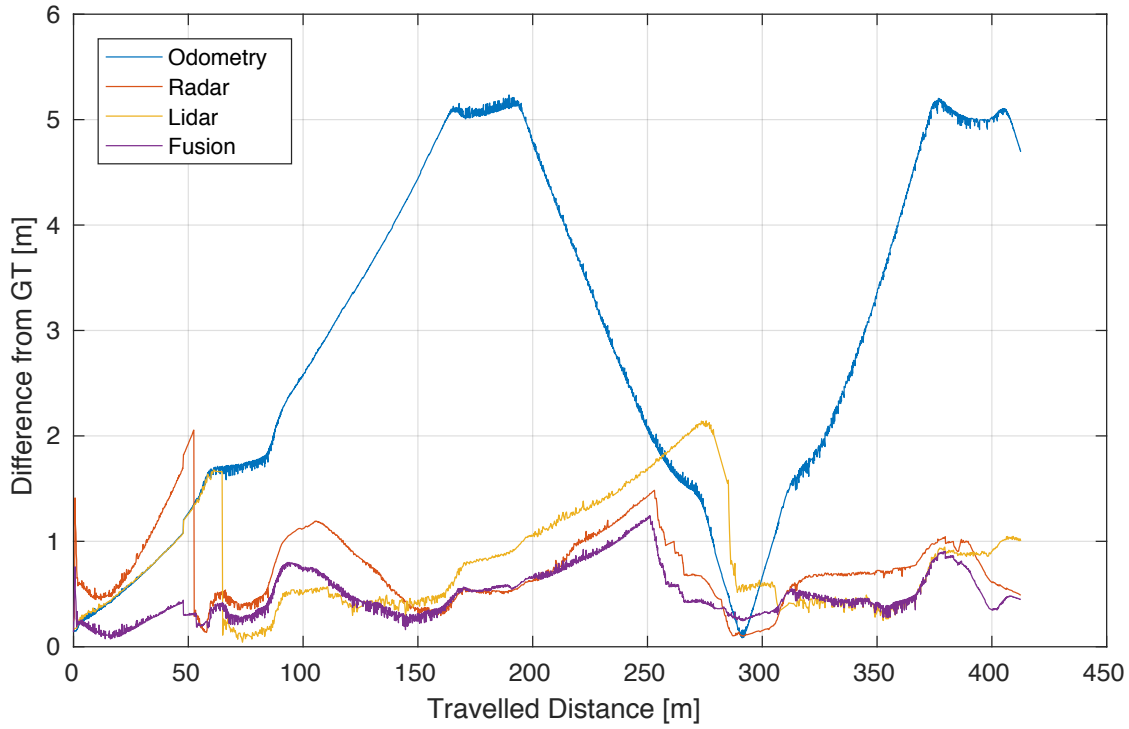



**Figure 5.9:** Position error for a specific trajectory comparing the odometry result to the fusion and the individual sensors.

to more drift than the radar sensor. If a good match from the LiDAR is established, the position error jumps typically to a much lower value than the RADAR can achieve – see around travelled distance from 50 m to 100 m in Figure 5.9. The fused result profits from both the consistent matches from the RADAR sensors and the highly accurate LiDAR matches and achieves overall better performance than the singular results as indicated by the purple line in Figure 5.9.

Fusing LiDAR and RADAR thus improves the overall accuracy and consistency of the localization. While the mapping performed slightly worse when combining the two sensors, the improvement in localization is significant with on average over 30 % better localization performance than the high precision LiDAR and more than 300 % better localization accuracy than RADAR alone. Combined with the availability improvements through Joint Graph Optimization and significant runtime improvements due to Feature Selection, the localization performance is sufficient to enable autonomous driving on the parking lot.

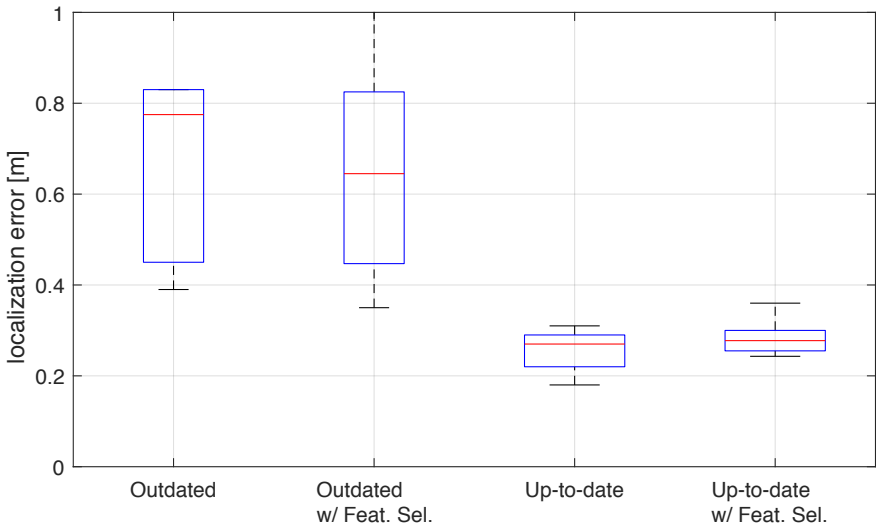
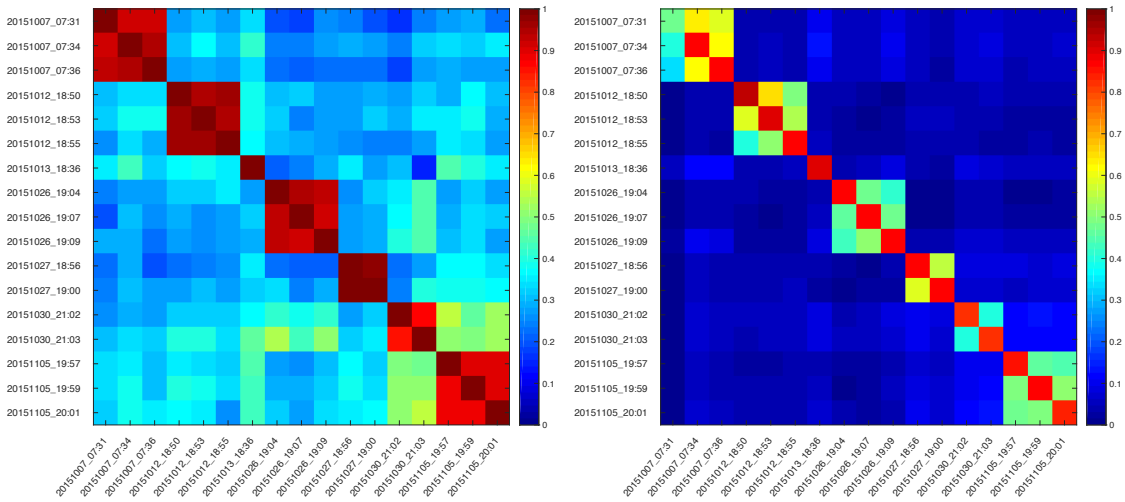



**Figure 5.10:** Boxplot indicating the fused localization accuracy with an outdated map and a current map on the eight shaped datatset.

The importance of the latter part is shown in Figure 5.10. While the fused localization accuracy improves slightly with feature selection on an outdated map, the most important part of the GraphSLAM framework is maintaining an up-to-date map.

In Figure 5.11 the cross validation between all eight shaped data is shown, because the trajectories cover the same area and are thus suitable to serve both as a mapping and localization input. On the $x$ axis, the different mapping sequences and on the




(a) RADAR (b) LiDAR

**Figure 5.11:** Cross validation of success rate of feature matching in eight shaped dataset.

$y$ axis the localization from different sequences is shown. The color indicates the amount of good matches over all matches. As expected, the diagonal has only perfect matches and on the same days of measurements, over 90 % of landmarks are identified again. The maps on other measurement days have roughly around 50 % to 70 % of successful matching. Moreover the RADAR sensor Figure 5.11 (a) with the BASD descriptors has a very high success rate even after several days, while the LiDAR matching performance deteriorates significantly more (Figure 5.11 (b)). This is on the one hand due to the descriptors, but also due to the measurement principle of the RADAR sensor as the occupancy grid map from which the features are extracted takes the reflection amplitude into account. The RADAR only reflects strongly from manmade structures, such as sharp corners and cylinders. Thus, they are also more persistent and recognizable in the landmark data. The LiDAR on the other hand merely performs a surface scan and is thus more dependent on view points, available line of sight and the geometry of parked vehicles.

## 5.2 Stereo Camera Integration Using Stixel

Beside the introduction of LiDAR into the GraphSLAM framework, camera based features are also suitable for GraphSLAM. It can be used to further increase the robustness of the localization, because it has another viewing angle and provides another means of feature extraction. The work provided in this section was performed in collaboration with M. Muffert [Muf18]. For this approach a stereo camera system is used to extract landmarks from disparity point clouds by introducing the Stixel environment representation. These landmarks are furthermore tracked and added as a third feature source to the GraphSLAM framework.

### 5.2.1 The Stixel Representation

In stereo vision, the raw image data is used to triangulate geometric information from each individual pixel, e.g. with the SGM algorithm. For the 1400 px × 400 px camera pair in the test vehicle this typically results to around 550,000 points per timestamp. Classical feature extraction algorithms break down when extracting features from such a high-density point cloud. Thus a data compression is required that does not sacrifice too much useful information.

The Stixel environment representation fulfills these requirements by segmenting the current disparity image into free space and object information [PF, BFP09]. It assumes that most man made structures have either horizontal or vertical planar surfaces and segments objects into vertically oriented adjacent rectangles. Each rectangle is called a Stixel, and has a fixed width in image space and a variable height. The free space is then given in the region from the bottom of the image to the base point of the Stixel. An example image segmented with Stixel is displayed in Figure 5.12. The Stixel representation reduces the amount of information that

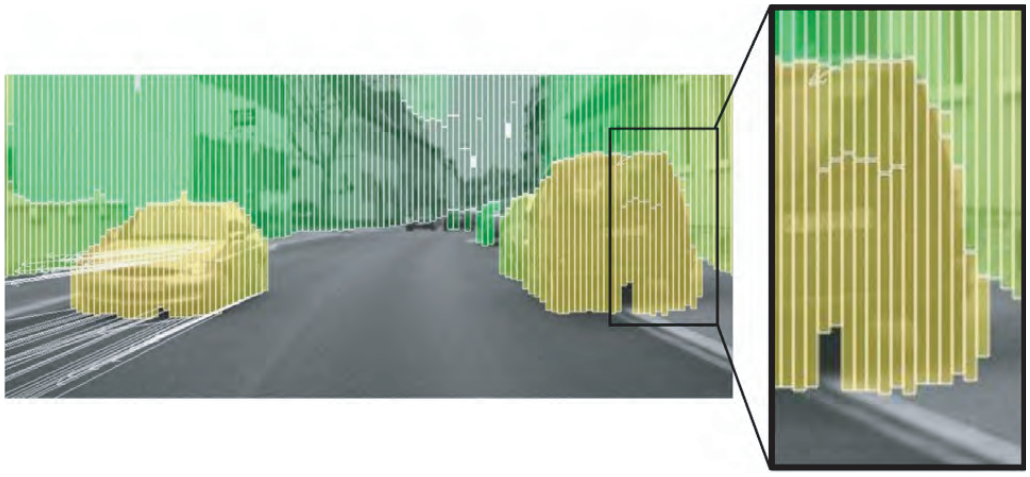


**Figure 5.12:** The Stixel representation on a sample image [Muf18].

needs to be processed by a factor of 1000 and is robust against stereo outliers [ESF12, EHPF12].

**Generation of the Stixel Representation** can be seen as a typical maximum a posteriori estimation problem which is solved by dynamic programming [Bel54]. It results in the most likely segmentation of the disparity image $\mathcal{D}$ into the classes $\mathcal{C}_s \in \{\text{free space, obstacle}\}$ from the possible set $\mathbb{S}$. Then the goal is to find the most probable Stixel labeling $\mathcal{S}^*$ given by

$$\mathcal{S}^* = \arg\max_{\mathcal{S}\in\mathbb{S}} p(\mathcal{S}|\mathcal{D}). \tag{5.2}$$

Because the Stixels are vertical objects in the image, it is more efficient to perform $W$ column-wise segmentations to obtain individual labelings $\mathcal{S}_u$ at image position $u$ into $N_u$ sub-classified Stixel $s_u^n$. Each Stixel $s_u^n$ has the following characteristics:

- $u$: column coordinate
- $v_{(u,bt)}^n$: row coordinates of bottom
- $v_{(u,tp)}^n$: row coordinates of top
- $d_u^n$: disparity
- $\sigma_{d_u^n}^2$: precision
- $c_u^n$: confidence
- $p_u^n$: outlier probability

The disparity $d_u^n$ is typically calculated by the mean over all disparity values that are contained in the Stixel and accordingly the variance $\sigma_{d_u^n}^2$ is calculated as the standard deviation of the disparity distribution within the Stixel. The confidence $c_u^n$ and the outlier probability $p_u^n$ are part of the result of the optimization problem in Equation 5.2. The full solution using dynamic programming is described in [Pfe12].

**Dynamic Stixels** are an extension of the Stixel representation described above to allow for object tracking. Up to this point, the Stixel only describe objects in a single disparity image. To track individual Stixel over time, the tracking algorithm is introduced by [PF10]. This approach requires the individual object velocities. While the RADAR is able to output the radial velocities of objects natively, for stereo based approaches they are estimated via optical flow [LK81]. Additionally, the vehicle's ego motion is required, as described in Section 2.1.1. Accordingly the Stixel definition for a dynamic Stixel is given by

$$s_u^{n,\text{dyn}} = \{s_u^n, \dot{x}_u^n, \dot{z}_u^n\}, \tag{5.3}$$

where $\dot{x}_u^n, \dot{z}_u^n$ are the longitudinal and lateral velocity of the object, respectively. The dynamic Stixels are thus tracked until they leave the FOV and can be referred

to as a landmark in the context of this thesis. They each have a coordinate in space – described by the column and row coordinates – and an identifier from the tracking algorithm. Note that this identifier is not unique as required in Section 4.3 though. Thus the detection of a loop closure is inherently difficult with a Stixel based approach.

### 5.2.2 GraphSLAM with Stixel

If the dynamic Stixels from Section 5.2.1 are treated as a landmark, they can be included into a GraphSLAM approach similar to the ones in previous chapters. The resulting graph can be seen in Figure 5.13. This has been done in collaboration with Maximilian Muffert [Muf18] on the KITTI benchmark dataset [GLSU13]. The Stixel extraction and tracking developed by Maximilian Muffert were combined with the GraphSLAM framework developed in this thesis to show the results presented in the following sections.

**Stixel Graph Construction** uses the dynamic Stixel and the odometry information. The motion model constraints are given as in the previous chapters by the Ackermann steering geometry. The pose-landmark constraints are generated from the static Stixel as determined by the tracking in Equation 5.3. Since the Stixel are tracked over time to determine the velocity, the observations forming a tracklet are used as observations for the graph construction. To improve the optimization, only the most stable features are selected, i.e. those that have been observed in at least in 50 frames. The measurement uncertainty is modeled by the theoretical precision of the triangulated position from image coordinates of the bottom point of the Stixel. Since the triangulation error increases quadratically with distance, only landmarks up to 40 m from the sensor are considered.

As previously pointed out, tracking of the Stixels does not enable a global search for matches like a descriptor matching approach would. However to detect loop closures and formulate global constraints, a global matching is necessary. In [Muf18] the loop closure detection was carried out by hand by assigning Stixel tracklets corresponding to the same physical object the same ID. In practice, it would be possible to generate a descriptor for each Stixel based on the neighboring object's characteristic properties, such as the disparity $d_u^n$, confidence $c_u^n$ and the outlier probability $p_u^n$ to apply the full matching capability of the GraphSLAM framework to the Stixel data.

**Stixel Graph Optimization** was done using the GraphSLAM framework described in Section 4.3 using a Levenberg-Marquardt optimization. Since the trajectory visits some locations multiple times, the optimization quality can be deduced from the overlapping of landmark shapes at these intersections. The result of the optimization for specific regions is shown in Figure 5.14. In the close-ups the landmark

optimization is visualized: before optimization landmarks are spread across a large

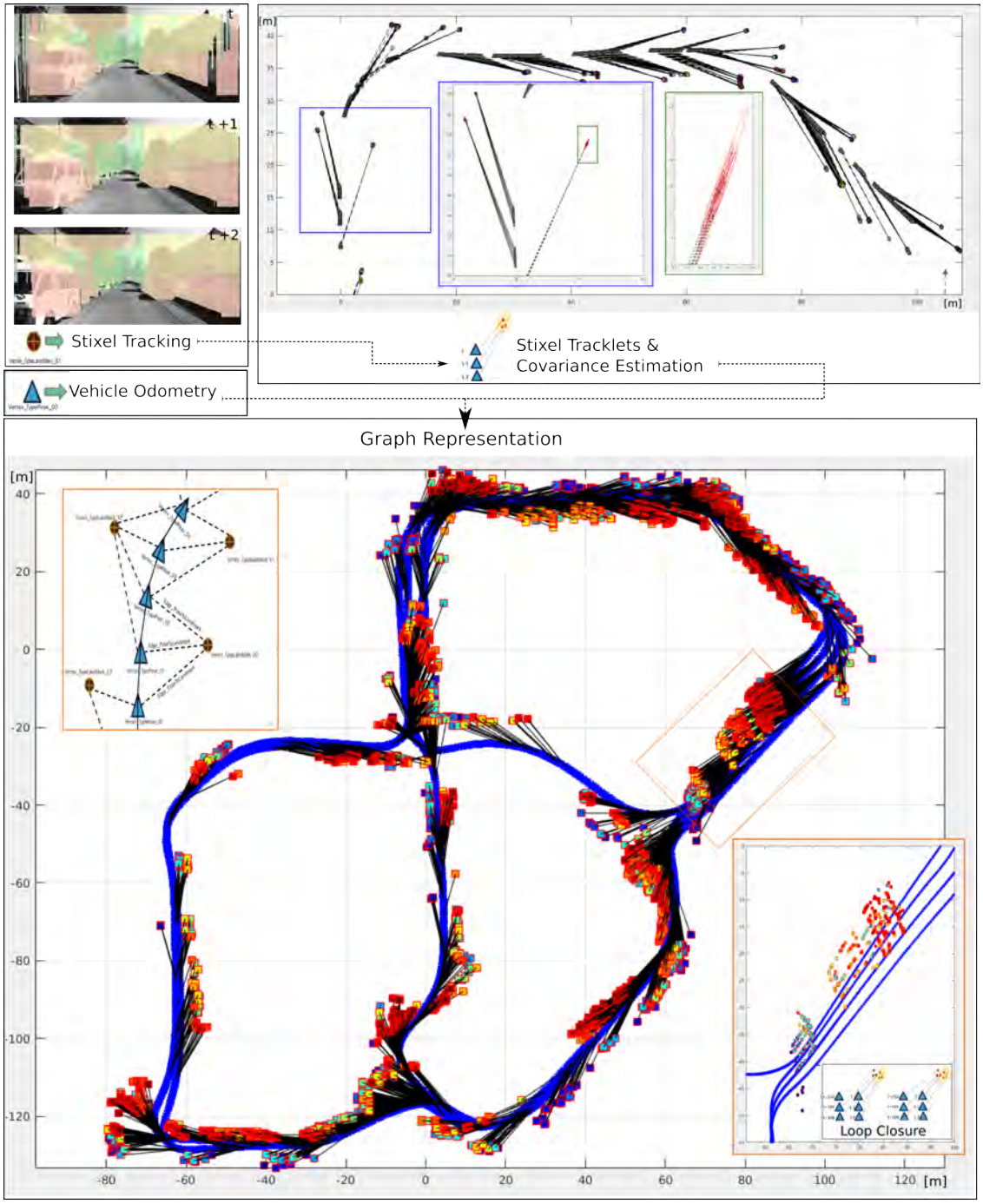



**Figure 5.13:** Construction of the graph from the recorded images. The tracked Stixels (colored boxes) are added to the graph of odometry poses (blue trajectory). The pose-to-landmark constraints are indicated by the black lines. The analysis was performed on the KITTI dataset. Courtesy [Muf18].

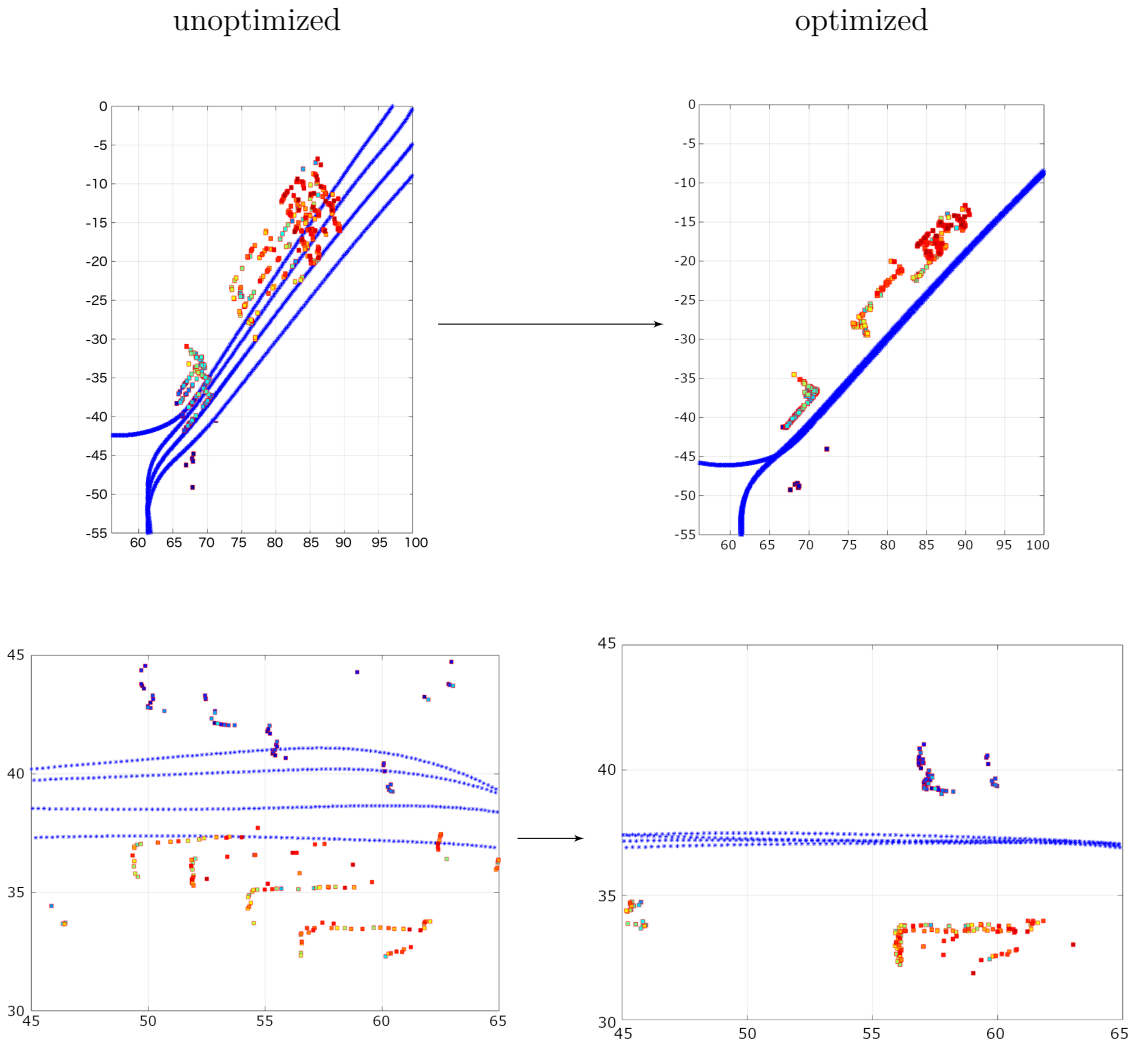



**Figure 5.14:** Graph of multiple rounds across the same location before and after optimization. The driven trajectory is shown in blue and the Stixel landmarks are marked as colored squares [Muf18].

area, after optimization they all fall onto the same physical object. The application of GraphSLAM to the KITTI dataset to a large scale urban scenario did not require many adaptations of the framework and shows the versatility of the framework. In practice, the urban scenario behaves significantly different to the parking lot, e.g. large stretches of the map do not have any landmarks. Still the localization performs reasonably well. The resulting landmark map before and after optimization can be seen in Figure 5.15. For a full quantitative evaluation of the Stixel based GraphSLAM approach, the interested reader is referred to the dissertation of Maximilian Muffert [Muf18] who produced the results as part of his investigations into incremental map building with Markov Random Fields on Stixel data.

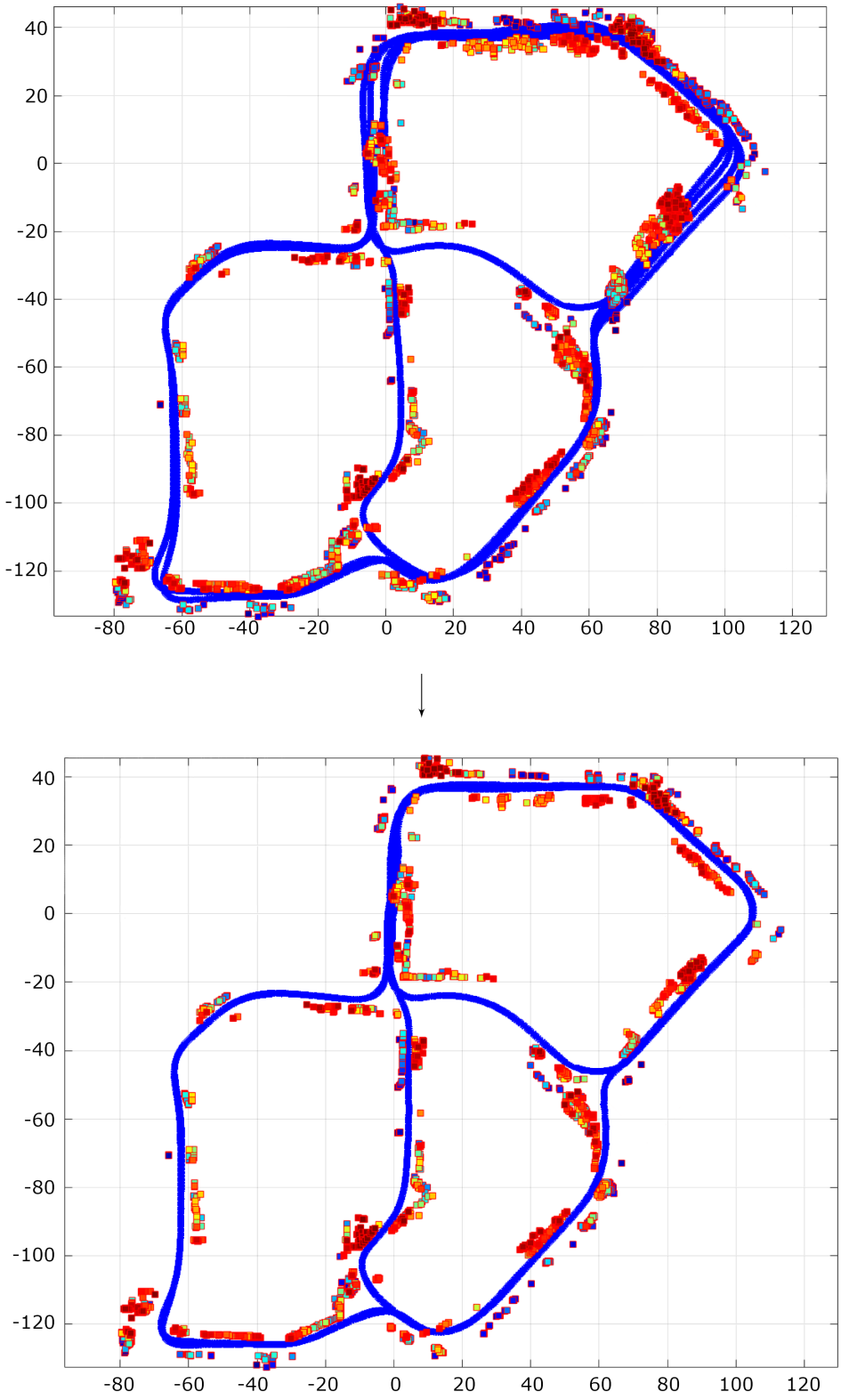



**Figure 5.15:** The entire landmark map (from the KITTI dataset [GLSU13]) before optimization (upper) and after optimization (lower) [Muf18].

## 5.3 Conclusion

In this chapter, the sensor fusion based on the GraphSLAM framework was discussed. By generating landmarks with unique identifiers from different sensors, the results of the GraphSLAM mapping and localization process can be improved with respect to the RADAR sensor. While it could be shown that adding a highly accurate LiDAR sensor improves the localization performance, it could also be demonstrated that the RADAR sensor improves localization performance especially in unknown environments due to its 360° vision and when the map is outdated because of its robust features. Furthermore, Stixel landmarks from a stereo camera system could be integrated into the GraphSLAM framework showing the versatility of the framework and the underlying g2o based optimization engine [KGS+11a].

# 6 Conclusion and Future Work

## 6.1 Conclusion

Vehicle localization is one of the most integral parts of autonomous driving, as it links the surroundings as perceived by the sensors to a map. In most cases and applications, highly precise DGNSS/INS systems are too expensive or impractical to perform adequate localization. As described in Chapter 1, the focus of this thesis lies on the parking use case, where the car also needs to perform well in GPS-denied areas, such as parking garages and thus must solve the localization problem purely with perception sensors. Here, the RADAR sensor is one of the most interesting and challenging sensors, because it is affordable, unobtrusive and robust against changing weather conditions. However it is intrinsically noisy and provides artifacts due to its measurement principle. With this sensor, the parking scenario is broken down into two steps: a manual recording of the parking trajectory, where the customer teaches the car where it should park and an autonomous mode, where the vehicle has to find its position on the initial map and then navigate to the predefined spot. But since this map only represents a past version of the environment, the localization performance on only the initial map will degrade over time. Thus this thesis solves the full SLAM problem in order to maintain the map such that the system can find the parking lot again for an indefinite period of time.

In Chapter 2 the unique characteristics of the RADAR sensor are described and the concept of designing a localization system that is capable of handling the unique point clouds that the RADAR provides. It is especially important to note the low density and high noise floor of the point clouds, as well as RADAR specific outliers such as speckle, multiple scattering and a limited angular resolution.

Yet, since an increasing amount of vehicles are equipped with RADAR sensors by default today to enable the most basic driver assistance systems, the results of this thesis are especially important for an application in a mass produced autonomous vehicle. It was shown in Chapter 4 that the RADAR sensor alone is sufficient for basic autonomous driving scenarios and that when taking the physical properties of the sensor into account, the sensor can produce accurate results even in dynamic scenarios like parking lots while keeping processing power and memory consumption sufficiently low for real applications. With Joint Graph Optimization, a framework is proposed in this thesis that not only enables shared access across multiple cars to map data for different environments from a backend server, but also allows all modern cars to participate in building a globally optimized map of the surroundings,

where each vehicle provides an updated or new piece of the map. Thus, complex parking lots, urban street networks or even highway networks can be mapped iteratively without the need for high cost sensors. Low cost localization systems such as the RADAR GraphSLAM framework are more relevant than ever, as most vehicle manufacturers are working on series autonomous driving systems that are fulfilling autonomous tasks in different scenarios investigated in this thesis.

In order to measure the performance of the localization and to build a prototype of the described system, a large dataset including measurements on a parking lot on multiple days over the course of several months was recorded, as described in Chapter 3. It was especially useful when evaluating the performance of the developed localization methods in highly dynamic environments and measuring the difference between using an outdated or an up-to-date map. The dataset does not only include RADAR data though. It also contains vision and LiDAR sensors, which can be used to analyze the drawbacks and merits of sensor fusion.

The contributions of this thesis have successfully introduced the RADAR sensor as a viable option for accurate localization into the research field of intelligent transportation systems and robotics. It was shown that RADAR sensors are capable of more than the ordinary ACC application and produce highly robust and accurate localization in complex, dynamic environments.

**Radar Based Particle Filters** were investigated both with a grid based map representation and a novel cluster based representation in Section 4.1 and Section 4.2, respectively. In both instances, we were able to solve the full SLAM problem using particle filters. While for grid based approaches, the initial implementation was provided by Werber et. al. [DWR+15], we were able to extend the approach by adding a map update capability to solve the full SLAM problem. The ClusterSLAM approach introduces a map representation based on density-based stream clustering to model the unique physical properties of the RADAR sensor. Solving the SLAM problem using a FastSLAM based approach proved to be robust on the cluster map representation. Furthermore the approach introduced a decay mechanism for clusters to become irrelevant for localization if they were not continuously confirmed by new measurements to handle the dynamic environment in the parking lot. However the multitude of free parameters both in the map representation and the Particle Filter reduced the suitable scenarios significantly and a more stable solution is needed.

**Radar based GraphSLAM** is described in Section 4.3 and provides a much more reliable and accurate solution than the previous approaches. While the ClusterSLAM approach used indistinguishable clusters to perform localization, the GraphSLAM approach utilizes RADAR based landmarks, which are annotated with a unique signature based on the surroundings of the feature. The BASD descriptor specifically designed to be used with RADAR grids was developed in collaboration with Rapp et. al. [RDH+16a]. After a powerful outlier detection using RANSAC, a graph is constructed using odometry information for pose constraints and the landmark observation as measurement constraints. Optimizing the graph yields a highly accurate feature map, which can be used for localization with an accuracy of below 1 m. The technique eliminates systematic, but local errors of the RADAR sensor by finding a globally consistent solution when optimizing the entire map. The described use case of self-parking on the parking lot can still be fulfilled if the optimization takes place after the vehicle has reached the parking destination taking all a priori information into account. The GraphSLAM algorithm serves as the foundation to a framework that encompasses a variety of features, including map update, crowd based mapping and sensor fusion. To the author's knowledge the RADAR GraphSLAM framework developed in this thesis is the first feature based RADAR localization and can show an efficient localization with a small map size and little memory usage. In order to achieve the level of robustness needed though, the map has to be kept up to date.

**Joint Graph Optimization** introduced in Section 4.4 is the first extension of the RADAR GraphSLAM framework to enable crowd based mapping on the parking lot. Instead of recording and optimizing the trajectory of the vehicle on-site, the information can be transmitted to an off-site server solving a global optimization problem over many maps from different sources and at different times of the day. This way, even vehicles that are not equipped with a self-parking system can contribute to the mapping of large areas when using an on-board RADAR sensor. The registration of the maps is achieved via a set of key points that are placed equidistantly along the mapping trajectory. Each key point then provides a small piece of the map, a key frame. The key frames are stored and matched among the maps from other vehicles to produce a set of constraints that interconnect the drives from different vehicles. Then the optimization problem of multiple maps determines the most consistent map given a large number of input maps. Redundant landmarks are then pruned from the map to select the relevant information to be stored, because they are most often used for localization. With an always up-to-date map, the localization result on a jointly optimized map for the RADAR sensor is below 0.6 m with a constant map size of about 571 $\mathrm{Mb/km^2}$. The architecture was designed such that only the recording of the map data and the localization needs to run on the vehicle, while any optimization and aggregation can be performed on a backend server. Due to the compact nature of the BASD landmarks, only small amounts of data have to be transmitted to the server.

**Sensor Fusion** is added to the GraphSLAM framework in Chapter 5 in order to improve the robustness of the system and to have a side-by-side comparison of the RADAR sensor with other sensors. In particular, two other landmark sources are considered: curvature based LiDAR landmarks and Stixel based stereo camera landmarks. Features were extracted from the LiDAR sensor using a curvature based detector and a scale invariant histogram based descriptor to provide stable landmarks. The approach is validated through an ego motion estimation resulting from tracking the landmarks. Since the landmarks proved suitable for localization, they are added into the optimization problem mentioned above alongside the RADAR landmarks and optimized together. The results of the fusion are compared to the performance of the individual sensors. While the mapping performance of the fused result was in between the LiDAR and RADAR mapping performance, localization and map update improved significantly even with respect to the highly accurate LiDAR sensor, because the RADAR landmarks prove to be very robust and the 360 degree RADAR setup ensures long tracking of individual landmarks, because they are considerably longer inside the FOV of the sensors. The total accuracy of the fused approach was below 0.3 m on an up-to-date map and thus proves suitable for the described scenario.

Furthermore, the GraphSLAM algorithm was optimized with Stixel based landmarks. Within a collaboration with [Muf18], it could be shown that slightly adapting the GraphSLAM framework to include tracked Stixel landmarks, which resulted in highly accurate maps. The experiments were executed on the KITTI urban dataset and thus show that the approach can be expanded to the urban use case, rather than the parking lot scenario.

## 6.2 Future Work

The RADAR GraphSLAM framework is a modular software that can be extended into a multitude of directions to further improve the localization performance, robustness and availability.

After a consistent map has been produced by Joint Graph Optimization, the landmarks are selected based on their relevance, which is determined by the number of maps containing a given landmark. An interesting extension could be to find a descriptor based map selection before features are added into the map. Since most landmarks that belong to dynamic objects are located on vehicle fronts, a classifier could be trained to determine whether a certain BASD descriptor represents a dynamic or static object. This should be possible because the descriptor uses the geometry of the surroundings and each vehicle front has a recognizable shape that would have a unique influence in the landmark description.

Another addition would be to enable the Stixel landmarks to detect loop closures with the same reliability as the RADAR and LiDAR landmarks do. This can be done in a similar way, as is done for the descriptors for BASD, by aggregating the properties, e.g. the disparity and the outlier probability of each Stixel in the neighborhood of the landmark, to obtain a specific signature that is invariant under a larger range of view points and distances.

Further research is needed concerning the integrity of the position. While the accuracy and availability are important to make an automotive system work, the integrity, or the trustworthiness of the position estimation make the localization system reliable and ensure a guaranteed remaining error rate [WSD+16, HSS+19]. Since the feature selection algorithm ranks the landmarks by their relevance with respect to the number of observations and how recently they have been observed, it is possible to deduct an integrity level from the quality of the landmarks. This ranking can even be used to dynamically adjust the planned trajectory of the autonomous system to find the most integer route, i.e. with the lowest expected remaining error.

# Acknowledgements

The work of this thesis would not have been possible without the help of the following people, who I would like to thank most sincerely for their support.

Prof. Dr. Cristóbal Curio supported the thesis from the University side and was always able to give unique and interesting remarks, which challenged me to analyze the problem at hand in more detail. Gratitude also goes to Prof. Dr. Andreas Schilling for agreeing to support this thesis as the second assessor.

Dr. Christoph Keller and Dr. Martin Haueis kindly supervised my thesis in the vehicle localization team at Daimler AG's R&D department. Their extensive knowledge in the field of localization and SLAM, as well as the research vehicle architecture were most helpful in many situations when completing this thesis.

Furthermore, I would like to thank Dr. Markus Hahn and Matthias Rapp for their extensive support when working with the RADAR sensors. Thanks to their preliminary work, the work in this thesis progressed as quickly as possible. Thanks also goes to Dr. Max Muffert, who provided deeper insight into the Stixel world.

Appreciation also goes to Dr. Friedrich Bös, Dr. Sabine Hofmann, Isabell Wayand and Stephanie Schöning for proof-reading this thesis and adding significant value to it by leaving constructive comments.

The student Master's theses and internships of Matthias Schmidt, Marcus Wörner, Florentin Schirmer, Davide Liberato, John Dan Pierce, Peter Hurt, Moritz Matti Henning and Wei Zhang helped shape and expand the framework of this thesis into multiple directions.

Finally I would like to thank Daimler AG and especially the entire vehicle localization team, as well as the Universities of Tübingen and Reutlingen for making the work described in this thesis possible.

# Glossary

**ABS** Anti-lock braking system.

**ACC** Adaptive Cruise Control.

**ADTF** Automotive Data and Time-Triggered Framework.

**BASD** Binary Annular Statistics Descriptor.

**CAN** Controller Area Network.

**CW** Continuous Wave.

**DBSCAN** Density-Based Spatial Clustering of Applications with Noise.

**DCS** Dynamic Covariance Scaling.

**DGNSS** Differential Global Navigation Satellite System.

**ECU** Electronic Control Unit.

**EKF** Extended Kalman Filter.

**ESP** Electronic Stability Program.

**FAST** Features from Accelerated Segment Test.

**FMCW** Frequency Modulated Constant Wavelength.

**FOV** Field of View.

**FPGA** Field Programmable Gate Array.

**FREAK** Fast Retina Keypoints.

**FSCD** Fast Scatter Center Detection.

**GNSS** Global Navigation Satellite System.

**GPS** Global Positioning System.

**ICP** Iterative Closest Point.

**IMU** Inertial Measurement Unit.

**KPI** Key Performance Indicator.

**LFSR** Linear Feedback Shift Register.

**LiDAR** Light Detection and Ranging.

**LRR** Long Range Radar.

**MEMS** Micro Electro Mechanical System.

**MISRA** Motor Industry Software Reliability Association.

**NDT** Normal Distribution Transformation.

**ORB** Oriented FAST and rotated BRIEF.

**RADAR** RAdio Detection And Ranging.

**RANSAC** RANdom SAmple Consensus.

**RMSE** Root Mean-Squared Error.

**SGM** Semi-Global Matching.

**SIFT** Scale-Invariant Feature Transform.

**SLAM** Simultaneous Localization and Mapping.

**SNR** Signal to Noise Ratio.

**SRR** Short Range Radar.

**SURF** Speeded Up Robust Features.

**SVD** Singular Value Decomposition.

# List of Figures

1.1 The parking scenario in front of an office building (upper image, Google Maps) is shown as an areal photography. Parking scenarios as shown schematically in the lower image, are both an important and challenging problem for autonomous driving applications. . . . . 7

2.1 Honda's Electro Cyro-Cator from 1981 was said to be the first navigation system using an Inertial Measurement System to determine the driving state of a vehicle (Source: Honda.com). . . . . 11

2.2 Ackermann steering describes the steering around a common centre of rotation, such that there is no wheel slip. This is archived by turning both front wheels with a different turning radius. . . . . 12

2.3 The emitted RADAR chirp wave form is depicted in the upper diagram, while the frequency domain is shown in the middle graph (transmitted $Tx$ and received $Rx$ signal). The lower graph shows the detected signal in the frequency domain, which forms from interference between $Tx$ and $Rx$. . . . . 15

2.4 The pulsed RADAR consists of sequential frequency pulses. The wave form is depicted in the upper diagram, while the frequency domain is shown in the lower graph ($Tx$, $Rx$). . . . . 16

2.5 The RADAR waves are transmitted by the Transmitter $Tx$ with antenna Gain $G$ and power $P_T$ and scattered off an effective area $A_R$ of the target. The reflected signal that is caught by the Receiver $Rx$ then has the remaining power $P_R$. . . . . 18

2.6 Comparison of a RADAR point cloud (lower) with a high precision sensor (LiDAR) point cloud (upper) on parking lot data in front of the parking lot shown in Figure 1.1. . . . . 21

2.7 Increased number of targets on the road as a result of speckle. . . . . 22

2.8 Targets on the road and rain gutters (marked red) as illustration of clutter. They have a high reflectivity and are therefore not filtered. . 23

2.9 Circular artifacts can result from insufficient vehicle velocities. The vehicle is located at the center of the figure and has 360 degree RADAR coverage. The circles originate from low angular resolution while the vehicle is standing still. . . . . 24

2.10 Typical simplified architecture of a multi sensor autonomous driving setup. The sensor data is usually aggregated in a sensor fusion module. The fused sensor data is passed to the localization module, which provides the correct planning information used for path planning. . . 25

2.11 Grid and landmark based approaches are most common among SLAM algorithms [LCW+12]. . . . . 26

2.12 Visual representation of the two-fold Bayes filter process of control update and measurement update [TBF05b]. Light blue distribution: measurement update, dark blue: updated posterior and black dot: observed landmark. . . . . 29

2.13 Top: Proposal distribution (dashed) and target distribution (solid). Middle: Particles sampled from the proposal distribution. Bottom: Sampled particles weighted according to the temporary set [TBF05b]. 32

2.14 Fundamental architecture of the GraphSLAM algorithm. . . . . 34

2.15 Model of a feature-based graph using the odometry poses and landmark observations as nodes. . . . . 35

2.16 The robust kernels plotted alongside the unaltered quadratic cost function. . . . . 40

3.1 Schematic representation of the vehicle equipped with additional sensors. The LiDAR opening angle can be seen in green and the four RADAR opening angles are displayed in blue. . . . . 47

3.2 Representations of the sensors used in the prototype vehicle. . . . . 49

3.3 The sensor coordinate systems for all relevant components (RADARs $R_1$ through $R_4$ and LiDAR L). All sensor inputs are calibrated to the middle of the rear axis of the vehicle. The red and blue arrows represent the $x$ and $y$ axis, respectively and the green dots/crosses represent the $z$ axis. . . . . 50

3.4 A Google Maps plot of the parking lot, alongside the 3D reference point cloud and the available datasets. . . . . 52

4.1 Sensor models showing the grid update based on the measurement uncertainty of different sensors. . . . . 57

4.2 Resulting occupancy grid map. . . . . 59

4.3 Update of the grid map on realistic parking maneuvers. The upper image (previous map) holds three parked cars in the red box, while in the lower image (current map) there are only two cars parked in the lower edge of the parking space. . . . . 61

4.4 Overview of the ClusterSLAM localization system. The system inputs raw RADAR data, as well as the odometry and outputs an estimated pose. . . . . 62

4.5 Different types of micro clusters. (a) represents outlier micro clusters with insufficient density. (b) potential micro cluster. The RADAR detections are shown in blue, the standard deviation is shown in yellow. The dashed line indicates the maximum size of each cluster before it is separated into two. . . . . 64
4.6 The cluster weight for a given point $p_i$ is calculated using the normal distance $d_{ij}$ of the center points $C_j$ from the line of sight originating from the sensor $S$ for each $p_i$. The radius $R_j$ indicates the $\sigma_j$ environment. . . . . 65
4.7 Map representation using $p$-micro-clusters in three levels of detail. As before, the applied color scheme represents the mean amplitude of the clusters. The ground truth trajectory is shown in green, odometry in red and the best pose estimate in orange. The cars parked in the parking lot environment are schematically overlayed in blue. . . . . 67
4.8 Illustration of scan matching. The scan $\mathcal{S}$ (dark blue) is overlaid onto the map $\mathcal{M}$. The vehicle pose is estimated based on the matching of $\mathcal{S}$ to $\mathcal{M}$. . . . . 70
4.9 Resampling of differently weighed particles. A random number $r$ is selected once. Then $M$ particles are equidistantly sampled from the particles. The color and length of the sticks indicates the weight. Darker colors and larger segments correspond to higher weights. Thus it is ensured, that higher weighted particles get redrawn more often. Based on [TBF05b]. . . . . 71
4.10 Deviation from ground truth of the pose error during the map creation phase of the best particle (dark blue) and the odometry (light blue) at 16:00 when the parking lot is highly occupied. The loop closure has been detected with an error of about 1 m at the end point. . . . . 73
4.11 The mean error of the mapping process for all sequences in the data set sorted by time of the day, which is an indicator for parking lot occupancy. The error bars indicate a $1\sigma$ environment. . . . . 73
4.12 The vehicle (blue drop shape) passes a region that changed since the initial mapping. Between the time steps $t_1$ and $t_2$, the map update removes the two indicated cars (black boxes), while preserving the immobile structures. . . . . 74
4.13 Localization results after initial mapping at 16:00. The respective vehicle configurations are illustrated as ground truth maps at the top. Each localization error with respect to the traveled distance compared to the odometry result is shown directly below. . . . . 76
4.14 Overview of the GraphSLAM localization system. . . . . 78
4.15 The radial scheme utilized in FSCD. The features are marked in white (boxes in left image) and the relevant radial points are numbered from 1 to 16 (right) [RDH+16b]. . . . . 81

4.16 The same scenario as in Figure 4.15 is shown with the full descriptor rings (right). To binarize the feature vector, the sign of the difference between the statistical values of 4.22 to 4.26 are used [RDH+16b]. . . 82
4.17 Graph of two portions of the parking lot before (left) and after (right) optimization. The graph consists of odometry and landmark measurements (vertices, green) and their observations (edges, gray). Before the optimization, the map contains drift error, as depicted in the upper left. The red line indicates the trajectory of the vehicle during mapping, landmarks are depicted in green. . . . . 87
4.18 Fully optimized map using GraphSLAM. . . . . 88
4.19 The trajectory of the vehicle as measured by odometry (upper) and calculated by the optimization (lower). Both plots include error ellipses, describing 95% confidence of the pose. . . . . 89
4.20 Overlay of the ground truth (red) and an optimized landmark map (blue), which have been aligned by matching a set of points (green) by hand between the ground truth and optimized landmark map. . . 91
4.21 Mean point error of a selection of 32 hand labeled features. The blue horizontal line at $\mu = 6.8\,\text{cm}$ marks the mean error. . . . . 91
4.22 Estimated vehicle position (blue) vs. ground truth (red). The optimized maps of the mapping (black) and at the time of localization (gray) are depicted. Mapping was performed on an almost fully occupied parking lot, while localization was performed when it was half empty. . . . . 92
4.23 Overview of the extended GraphSLAM localization system. Gray elements depict the static localization system from [SKR+16], while the white ones show the crowd based addition to the system. . . . . 94
4.24 Local match based registration showing equidistant sampling of the base map (red). The key points are matched to the increment map (blue) and constraints between the two maps are determined. . . . . 95
4.25 Joint optimization of six test drives. From (a) through (e) the graph is optimized before another drive is added. The map gets more consistent incorporating new data during each additional parking drive. In (f) all six drives have been optimized together to form a globally optimized map. . . . . 96
4.26 Clustering of landmarks in a map of the parking area. Different colors indicate different clusters. . . . . 99
4.27 Histogram of deviations from the ground truth. The dark blue color indicates the individually processed maps, the light blue the joint optimization. . . . . 101
4.28 Localization accuracy both with and without joint graph registration. On average, localization accuracy with single optimization is 0.79 m. The localization accuracy with joint graph registration is 0.55 m. . . . 102

4.29 Comparison of the localization performance and availability for the updated map (a) and the outdated map (b). The grey dots represent the map that was used for reference, while the black dots indicate landmarks measured during localization. The triangles indicate the localization result. . . . . 103
4.30 Feature map size on the *eight shaped dataset*, where all trajectories overlap both with feature selection and without. . . . . 104

5.1 Overview of the GraphSLAM localization system. The gray elements depict the static localization system from [SKR+16], while the white ones show the crowd based addition to the system. . . . . 107
5.2 Curvature based feature detection. The second derivative of the upper image is shown in the lower image. . . . . 110
5.3 The descriptor of the LiDAR landmarks uses polar histograms around the detected point. . . . . 111
5.4 Ego motion estimation from LiDAR landmarks (blue dots) vs. reference measurement from the iTrace system (red line). . . . . 112
5.5 RADAR and LiDAR landmarks (unoptimized) in the same optimization problem with the LiDAR data in light blue and the RADAR data in black. . . . . 113
5.6 Mean pose error of the initial mapping for RADAR, LiDAR and fused result per trajectory. . . . . 114
5.7 The Joint Graph Optimization mapping error for RADAR, LiDAR and fusion on the realistic parking maneuvers. The fusion is evaluated both with and without feature selection. The boxes indicate the quantiles and the bars indicate minimum and maximum errors. . . . . 115
5.8 The finalized jointly optimized map from sensor fusion with respect to the total station. . . . . 116
5.9 Position error for a specific trajectory comparing the odometry result to the fusion and the individual sensors. . . . . 117
5.10 Boxplot indicating the fused localization accuracy with an outdated map and a current map on the eight shaped datatset. . . . . 118
5.11 Cross validation of success rate of feature matching in eight shaped dataset. . . . . 118
5.12 The Stixel representation on a sample image [Muf18]. . . . . 120
5.13 Construction of the graph from the recorded images. The tracked Stixels (colored boxes) are added to the graph of odometry poses (blue trajectory). The pose-to-landmark constraints are indicated by the black lines. The analysis was performed on the KITTI dataset. Courtesy [Muf18]. . . . . 123
5.14 Graph of multiple rounds across the same location before and after optimization. The driven trajectory is shown in blue and the Stixel landmarks are marked as colored squares [Muf18]. . . . . 124

5.15 The entire landmark map (from the KITTI dataset [GLSU13]) before optimization (upper) and after optimization (lower) [Muf18]. . . . . 125

# List of Algorithms


1 Particle Filter . . . . . 33
2 Gauss-Newton Algorithm . . . . . 38
3 Levenberg-Marquardt Algorithm . . . . . 39

4 Radar Grid Mapping . . . . . 59
5 Odometry Motion Model . . . . . 69
6 Low Variance Resampling . . . . . 71
7 FSCD algorithm on an occupancy grid map . . . . . 80
8 Binary feature matching . . . . . 84
9 RANSAC . . . . . 85
10 Modified RANSAC . . . . . 97
11 Mean Shift Clustering . . . . . 98

# Bibliography


[ABS14] P. Agarwal, W. Burgard, and C. Stachniss. A Survey of Geodetic Approaches to Mapping and the Relationship to Graph-Based SLAM. *IEEE Robotics & Automation Magazine*, pages 63–80, sep 2014.

[ABY+13] J. Almazan, L. M. Bergasa, J. J. Yebes, R. Barea, and R. Arroyo. Full auto-calibration of a smartphone on board a vehicle using IMU and GPS embedded sensors. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 1374–1380. IEEE, jun 2013.

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5): 698–700, sep 1987.

[AJ12] M. Adams and E. Jose. *Robotic Navigation and Mapping with Radar*. Artech House, 2012.

[AOV12] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517, 2012.

[ATS+13] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust map optimization using dynamic covariance scaling. In *2013 {IEEE} International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 62–69, 2013.

[AWS14] A. Amini, T. Wah, and H. Saboohi. On Density-Based Data Streams Clustering Algorithms: A Survey. *Journal of Computer Science and Technology*, 29(1): 116–141, 2014.

[Bel54] R. Bellman. THE THEORY OF DYNAMIC PROGRAMMING, 1954.

[BFP09] H. Badino, U. Franke, and D. Pfeiffer. The stixel world - A compact medium level representation of the 3d-world. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5748 LNCS, pages 51–60, 2009.

[BM92] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2): 239–256, feb 1992.

[BSG+09] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kuemmerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tardos. A Comparison of SLAM Algorithms Based on a Graph of Relations. In *Proceedings of IEEE/RSJ Conference on Robots and Systems (IROS)*, St.Louis, MO, USA, 2009.

[BTV06] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. pages 404–417. Springer, Berlin, Heidelberg, 2006.

[BY06] M. Bühren and B. Yang. Simulation of Automotive Radar Target Lists using a Novel Approach of Object Representation. In *IEEE Intelligent Vehicles Symposium. Proceedings.*, pages 314–319, 2006.

[BZ09] M. Bosse and R. Zlot. Keypoint design and evaluation for place recognition in 2D lidar maps. *Robotics and Autonomous Systems*, 57(12): 1211–1224, dec 2009.

[CCR07] T. Collins, J. J. Collins, and C. Ryan. Occupancy grid mapping: An Empirical Evaluation. In *IEEE Mediterranean Conference on Control and Automation. Proceedings.*, pages 1–6, 2007.

[CGB+10] P. Checchin, F. Gerossier, C. Blanc, R. Chapuis, and L. Trassoudaine. Radar Scan Matching SLAM Using the Fourier-Mellin Transform. In Andrew Howard, Karl Iagnemma, and Alonzo Kelly, editors, *Field and Service Robotics*, volume 62 of *Springer Tracts in Advanced Robotics*, pages 151–161. Springer Berlin Heidelberg, 2010.

[Che13] Y. Chen. Algorithms for Simultaneous Localization and Mapping. Technical report, 2013.

[con19] Continental Radar Sensor Overview. Technical report, Continental AG, 2019.

[CTG+11] J. Callmer, D. Törnqvist, F. Gustafsson, H. Svensson, and P. Carlbom. Radar SLAM Using Visual Features. *EURASIP Journal on Advances in Signal Processing*, 2011(1), 2011.

[DAB+14] J. Dickmann, N. Appenrodt, H. L. Bloecher, C. Brenk, T. Hackbarth, M. Hahn, J. Klappstein, M. Muntzinger, and A. Sailer. Radar contribution to highly automated driving. In *European Radar Conference (EuRAD), 2014 11th*, pages 412–415, 2014.

[DHS+14] R. Dube, M. Hahn, M. Schutz, J. Dickmann, and D. Gingras. Detection of Parked Vehicles from a Radar Based Occupancy Grid. In *IEEE Intelligent Vehicles Symposium. Proceedings.*, pages 1415–1420, 2014.

[DKB+12] J. Dickmann, J. Klappstein, H. Bloecher, M. Muntzinger, and H. Meinel. Automotive radar — "quo vadis?". In *2012 9th European Radar Conference*, pages 18–21, Oct 2012.

[DWR+15] J. Dickmann, K. Werber, M. Rapp, J. Klappstein, M. Hahn, J. Dickmann, K. Dietmayer, and C. Waldschmidt. Automotive radar gridmap

representations. In *IEEE International Conference on Microwaves for Intelligent Mobility (ICMIM)*, 2015.

[ECQZ06] M. Estert, F. Cao, W. Qian, and A. Zhou. Density-Based Clustering over an Evolving Data Stream with Noise. In *SIAM International Conference on Data Mining. Proceedings.*, pages 328–339, 2006.

[EG13] E. Einhorn and H.-M. Gross. Generic 2D/3D SLAM with NDT Maps for Lifelong Application. In *European Conference on Mobile Robots. Proceedings.*, pages 240–247, sep 2013.

[EHPF12] Markus Enzweiler, Matthias Hummel, David Pfeiffer, and Uwe Franke. Efficient stixel-based object recognition. In *2012 IEEE Intelligent Vehicles Symposium*, pages 1066–1071. IEEE, 2012.

[EKSX96] M. Ester, H. P. Kriegel, J S, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *AAAI Conference on Artificial Intelligence. Proceedings*, pages 226–231. AAAI Press, 1996.

[EP03] A. Eliazar and R. Parr. DP-SLAM: Fast, Robust Simultaneous Localization and Mapping without Predetermined Landmarks. In *International Joint Conference on Artificial Intelligence. Proceedings.*, pages 1135–1142. Morgan Kaufmann, 2003.

[ESF12] F. Erbs, B. Schwarz, and U. Franke. Stixmentation-probabilistic stixel based traffic scene labeling. In *Bmvc*, pages 1–12, 2012.

[FH75] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21: 32–40, 1975.

[FKW07] N. Fairfield, G. A. Kantor, and D. Wettergreen. Real-Time SLAM with Octree Evidence Grids for Exploration in Underwater Tunnels. *Journal of Field Robotics*, 2007.

[GLSU13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11): 1231–1237, sep 2013.

[GSB05] G. Grisetti, C. Stachniss, and W. Burgard. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In *IEEE International Conference on Robotics and Automation. Proceedings.*, pages 2432–2437, apr 2005.

[Gut84] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD '84*, volume 14, page 47, New York, New York, USA, 1984. ACM Press.

[Hir06] Heiko Hirschmuller. Stereo vision in structured environments by consistent semi-global matching. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2386–2393. IEEE, 2006.

[HSS+19] Isabell Hofstetter, Michael Sprunk, Frank Schuster, Florian Ries, and Martin Haueis. On Ambiguities in Vehicle Localization and their A Priori Detection in Maps. In *IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[Hub73] P. J. Huber. Robust regression: Asymptotics, conjectures and monte carlo. *The Annals of Statistics*, 1: 799–821, 1973.

[ibe19] DATA SHEET ibeo ScaLa B3.0. Technical report, Ibeo Automotive, 2019.

[iMA14] iMAR Navigation GmbH. iTraceRT-F400-E Accurate Real-Time Surveying, Vehicle Trajectory and Dynamics Estimation with deeply coupled INS/GNSS Filtering. Technical report, iMAR Navigation GmbH, Im Reihersbruch 3, 66386 St. Ingbert, Germany, 2014.

[iso11] Road vehicles-Vehicle dynamics and road-holding ability-Vocabulary. Iso 8855, International Organization for Standardization, 2011.

[JH13] V. Jain and P. Heydari. *Automotive radar sensors in silicon technologies*, volume 9781441967. 2013.

[KBD+15] D. Kellner, M. Barjenbruch, K. Dietmayer, J. Klappstein, and J. Dickmann. Joint Radar Alignment and Odometry Calibration. Technical report, 2015.

[KGS+11a] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, Shanghai, China, 2011.

[KGS+11b] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613, 2011.

[KHD+09] R. Kümmerle, D. Hahnel, D. Dolgov, S. Thrun, and W. Burgard. Autonomous driving in a multi-level parking structure. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3395–3400, 2009.

[KMN+] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An Efficient k-Means Clustering Algorithm: Analysis and Implementation. Technical report.

[Kri14] S. Krig. Interest Point Detector and Feature Descriptor Survey. *Computer Vision Metrics*, pages 217–282, 2014.

[Küm13] R. Kümmerle. *State Estimation and Optimization for Mobile Robot Navigation*. PhD thesis, Albert-Ludwigs-University of Freiburg, Department of Computer Science, apr 2013.

[LCW+12] J. Li, L Cheng, H. Wu, L. Xiong, and D. Wang. An Overview of the Simultaneous Localization and Mapping on Mobile Robot. In *IEEE International Conference on Modelling, Identification Control. Proceedings.*, pages 358–364, 2012.

[Lin12] T. Lindeberg. Scale Invariant Feature Transform. *Scholarpedia*, 7(5): 10491, 2012.

[LK81] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. 1981.

[LMV+17] A. Ly, M. Marsman, J. Verhagen, R. Grasman, and E. Wagenmakers. A Tutorial on Fisher Information *. Technical report, 2017.

[Mal10] H. Malepati. *Digital media processing : DSP algorithms using C*. Newnes/Elsevier, 2010.

[Mar63] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11: 413–441, 1963.

[Mat19] M. Matousek. Waymo CEO John Krafcik explains big challenge for self-driving cars, 2019.

[MDBB12] D. Meyer-Delius, M. Beinhofer, and W. Burgard. Occupancy Grid Models for Robot Mapping in Changing Environments. 2012.

[MDHGB10] D. Meyer-Delius, J. M. Hess, G. Grisetti, and W. Burgard. Temporary maps for robust localization in semi-static environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5750–5755, 2010.

[ME85] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar, 1985.

[MG56] E. L. Murphy and R. H. Good, Jr. Thermionic Emission, Field Emission, and the Transition Region. *Phys. Rev.*, 102: 1464–1473, 1956, http://dx.doi.org/10.1103/PhysRev.102.1464.

[MMPF] M. Muffert, T. Milbich, D. Pfeiffer, and U. Franke. May I Enter the Roundabout? A Time-To-Contact Computation Based on Stereo-Vision. Technical report.

[MSS06] W. C. Mitchell, A. Staniforth, and I. Scott. Analysis of Ackermann Steering Geometry. *SAE Technical Papers U6*, 2006.

[Muf18] M. Muffert. Incremental Map Building with Markov Random Fields and its Evaluation. 2018.

[MVAV11] J. Mullane, B. Vo, M. D. Adams, and B. Vo. A Random-Finite-Set Approach to Bayesian SLAM. *IEEE Transactions in Robotics*, 27(2): 268–282, apr 2011.

[PF] D. Pfeiffer and U. Franke. Towards a Global Optimal Multi-Layer Stixel Representation of Dense 3D Data. *researchgate.net*.

[PF10] D. Pfeiffer and U. Franke. Efficient representation of traffic scenes by means of dynamic stixels. In *2010 IEEE Intelligent Vehicles Symposium*, pages 217–224, 2010.

[Pfe12] D. Pfeiffer. *The Stixel World*. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, 2012.

[PRLM+09] L. Pedraza, D. Rodriguez-Losada, F. Matia, G. Dissanayake, and J. V. Miro. Extending the Limits of Feature-Based SLAM With B-Splines. *IEEE Transactions in Robotics*, 25(2): 353–366, apr 2009.

[PT05] M. A. Paskin and S. Thrun. Robotic Mapping with Polygonal Random Fields. In Faheim Bacchus and Tommi Jaakkola, editors, *Conference on Uncertainty in Artificial Intelligence. Proceedings.* AUAU Press, Arlington, Virginia, 2005.

[RDH+16a] M. Rapp, K. Dietmayer, M. Hahn, B. Duraisamy, and J. Dickmann. Hidden Markov model-based occupancy grid maps of dynamic environments. *FUSION 2016 - 19th International Conference on Information Fusion, Proceedings*, 2016.

[RDH+16b] M. Rapp, K. Dietmayer, M. Hahn, F. Schuster, J. Lombacher, and J. Dickmann. FSCD and BASD: Robust landmark detection and description on radar-based grids. In *2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. IEEE, 2016.

[RFM10] R. Rouveure, P. Faure, and M. O. Monod. Radar-based SLAM Without Odometric Sensor. In *Actes du colloque*, page 9 p., Clermont Ferrand, France, 2010.

[RGH+13] M. Rapp, T. Giese, M. Hahn, J. Dickmann, and K. Dietmeyer. A Feature-Based Approach For Group-Wise Grid Map Registration. pages 5198–5203, 2013.

[RGH+14] M. Rapp, T. Giese, M. Hahn, M. Muntzinger, J. Dickmann, and K. Dietmayer. Merkmalsbasiertes Map-Matching von radarbasierten Belegungskarten. *Fahrerassistenzsysteme Workshop*, pages 17–26, 2014.

[RHT+15] M. Rapp, M. Hahn, M. Thom, J. Dickmann, and K. Dietmayer. Semi-Markov Process Based Localization Using Radar in Dynamic Environments. In *IEEE International Conference on Intelligent Transportation Systems*, pages 423–429, sep 2015.

[RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571. IEEE, nov 2011.

[RSH+16] M. Rapp, F. Schuster, M. Hahn, J. Dickmann, and K. Dietmayer. ScatFAST and BASD: Robust Landmark Detection and Description on Radar-Based Grids. In *International Conference on Microwaves for Intelligent Mobility (ICMIM)*, page to appear. IEEE, 2016.

[Sch13] F. Schuster. *Development of New Methods of Data Transmission for the Upgraded ATLAS Pixel Detector at the HL-LHC*. PhD thesis, Georg-August Universität Göttingen, 2013.

[SHR17] O. Sorkine-Hornung and M. Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1), 2017.

[SK16] F. Schuster and C. G. Keller. DE102015011358: Verfahren zum Betrieb eines Fahrzeugs, 2016.

[SKH16] F. Schuster, C. G. Keller, and M. Haueis. Measuring the World: Designing Robust Vehicle Localization for Autonomous Driving. In *2016 IEEE Intelligent Vehicles Symposium (IV) - Workshops*, 2016.

[Sko85] M. I. Skolnik. Fifty years of radar. *Proceedings of the IEEE*, 73(2): 182–197, 1985, http://ieeexplore.ieee.org/document/1457400/.

[Sko08] M. I. Skolnik. *Radar Handbook, Third Edition*. Electronics electrical engineering. McGraw-Hill Education, 2008.

[SKR+16] F. Schuster, C. G. Keller, M. Rapp, M. Haueis, and C. Curio. Landmark based Radar SLAM Using Graph Optimization. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

[SML12] T. Stoyanov, M. Magnusson, and A. J. Lilienthal. Point Set Registration Through Minimization of the L2 Distance Between 3D-NDT Models. In *IEEE International Conference on Robotics and Automation. Proceedings.*, pages 5196–5201, 2012.

[SSF15] U. Schwesinger, R. Siegwart, and P. Furgale. Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners, may 2015.

[SWK+16] F. Schuster, M. Wörner, C.G. Keller, M. Haueis, and C. Curio. Robust localization based on radar signal clustering. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016.

[SWKH16a] F. Schuster, M. Wörner, C. G. Keller, and M. Haueis. DE102015003666: Verfahren zur Verarbeitung von erfassten Messdaten eines Sensors, 2016.

[SWKH16b] F. Schuster, M. Wörner, C. G. Keller, and M. Haueis. DE102015011467: Verfahren zur Erstellung einer digitalen Karte eines Parkraums, 2016.

[SZK+17] F. Schuster, W. Zhang, C.G. Keller, M. Haueis, and C. Curio. Joint graph optimization towards crowd based mapping. In *International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017.

[TA10a] G. D. Tipaldi and K O Arras. FLIRT - Interest regions for 2D range data. In *2010 IEEE International Conference on Robotics and Automation*, pages 3616–3622. IEEE, may 2010.

[TA10b] G. D. Tipaldi and K. O. Arras. Flirt-interest regions for 2d range data. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3616–3622. IEEE, 2010.

[TBF05a] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. the MIT Press, Cambridge (Mass.) (London), 2005.

[TBF05b] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[TM05] S. Thrun and M. Montemerlo. The GraphSLAM Algorithm With Applications to Large-Scale Mapping of Urban Structures. *International Journal on Robotics Research*, 25(5/6): 403–430, 2005.

[TSW16] J. Timpner, D. Schurmann, and L. Wolf. Trustworthy Parking Communities: Helping Your Neighbor to Find a Space. *IEEE Transactions on Dependable and Secure Computing*, 13(1): 120–132, jan 2016, http://ieeexplore.ieee.org/document/7097675/.

[Van08] S. A. Van De Geer. Least-Squares Estimation. In *Encyclopedia of Statistics in Quality and Reliability*. John Wiley & Sons, Ltd, Chichester, UK, mar 2008.

[WBK+14] K. Werber, M. Barjenbruch, J. Klappstein, J. Dickmann, and C. Waldschmidt. How do traffic signs look like in radar? In *2014 44th European Microwave Conference*, pages 135–138. IEEE, oct 2014.

[WHB+10] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In *IEEE International Conference on Robotics and Automation. Proceedings.*, 2010.

[WSD+16] M. Wörner, F. Schuster, F. Dolitzscher, C. G. Keller, M. Haueis, and K. Dietmayer. Integrity for autonomous driving: A survey. In *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE, 2016.

[YO10] Yangming L. and E. B. Olson. Extracting general-purpose features from LIDAR data. In *2010 IEEE International Conference on Robotics and Automation*, pages 1388–1393. IEEE, may 2010.

[ZBS+14] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knöppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb. Making Bertha Drive–An Autonomous Journey on a Historic Route. *{IEEE} Intell. Transport. Syst. Mag.*, 6: 8–20, 2014.

[ZCS+08] H. Zhao, M. Chiba, R. Shibasaki, X. Shao, J. Cui, and H. Zha. SLAM in a Dynamic Large Outdoor Environment Using a Laser Scanner. pages 1455–1462, 2008.

[ZLN09] H.-M. Zogg, W. Lienhart, and D. Nindl. Leica TS30 White Paper, 2009.

[ZLS+14] J. Ziegler, H. Lategahn, M. Schreiber, C.G. Keller, C. Knoppel, J. Hipp, M. Haueis, and C. Stiller. Video Based Localization for Bertha. In *IEEE Intelligent Vehicles Symposium. Proceedings.*, pages 1231–1238, 2014.